Security Operation Center Lab

Bachelorarbeit

Studiengang Informatik
OST – Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Frühjahrsemester 2021

Autorin:	Alexandra Diener
Betreuer:	Ivan Bütler
Projektpartner:	OST
Experte:	Thomas Röthlisberger
Gegenleser:	Prof. Dr. Farhad D. Mehta

Erstellt am: **14. Juni 2021**

Letzte Änderung am: **15. Juni 2021**

Abstract

Problemstellung

Für das kommende Semester bietet die OST ein neues Cyber Defense Modul an. Da die meisten Unternehmen auf Microsoft Windows und Active Directory basieren, soll mit dieser Arbeit eine On-Demand Lernplattform für Studierende entwickelt werden, damit diese den Aufbau eines Security Operation Center (SOC) erlernen und praktisch anwenden können.

Ziel der Arbeit

Diese Arbeit soll ein Basis SOC Framework bereitstellen, welches im zukünftigen Cyber Defense Modul verwendet wird. Der Fokus ist hierbei ein SOC für Ausbildungszwecke bereit zu stellen, welches in ein virtuelles On-Demand Windows Active Directory eingebunden ist. Die Infrastruktur wird dabei über das Hacking-Lab der OST pro Student oder Studentengruppe in der Azure Cloud deployed. Sekundäres Ziel dieser Arbeit ist es, diverse Übungsszenarien zu konzipieren, welche die Studenten darin befähigen ein SOC einzurichten, zu nutzen und entsprechende Erfahrungen damit zu sammeln.

Ergebnis

Primäres Resultat dieser Arbeit ist die erfolgreiche Erweiterung des On-Demand Active Directory Netzwerkes um ein SOC und einem virtuellen Attack Launcher Service. Als SOC Lösung wurde die Open Source Software Wazuh evaluiert. Für die Simulation der Hacker Angriffe wurde auf Basis von Docker ein eigener «Attack-Launcher» Service entwickelt. Diesen nutzen die Studenten, um vordefinierte Attacken auf die Infrastruktur zu lancieren und entsprechende Alerts im SOC auszulösen. Ausserdem kann das virtuelle SOC Framework mit ihren diversen Services pro Student oder Studentengruppe über das Hacking-Lab der OST in der Azure Cloud deployed werden.

Schlussendlich sind auch diverse Übungsszenarien entstanden, wie beispielsweise das Einrichten von Log Forwardern, die Verbesserung der Log Qualität mittels Active Directory, Group Policy und Verbesserung der Erkennungsrate mittels externen Indicators of Compromise (IOCs).

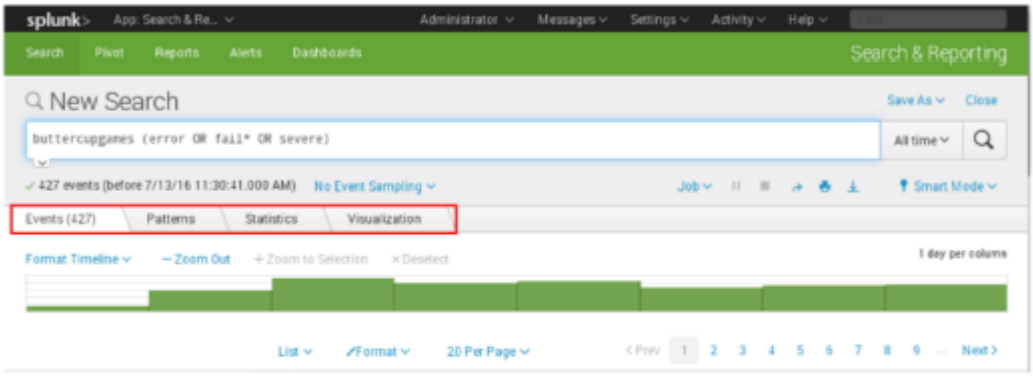
Inhalt

1 Aufgabenstellung im Original

Security Operation Center Lab

Einführung

Im kommenden Herbstsemester 2021 findet an der FH OST zum ersten Mal das Cyber Defense Modul statt. Im Fokus stehen präventive Massnahmen um Cyber Angriffe und den Abfluss von Daten (Exfiltration) zu erkennen. Dazu werden Log Daten von DNS, Firewalls, HTTP Proxies, Endpoint Security etc. an eine zentrale Plattform (Cyber Attack Handling Plattform) geschickt, welche dann als Werkzeug für ein SOC (Security Operation Center) bereitsteht.



Aufgabenstellung

Bei dieser Arbeit geht es darum im Hacking-Lab entsprechende Cyber Defense Übungen und die dazu notwendige Technik zu entwickeln, damit Studenten einen praktischen Bezug zum Thema erhalten. Die exakte Aufgabenstellung und der genaue Ablauf der Übungen kann und soll durch die Studierenden mitdefiniert werden. In diesem Sinn ist der folgende Absatz eher als Idee gemeint, so dass Sie sich etwas darunter vorstellen können.

Idee eines solchen Cyber Defense Szenario

Die Studierenden müssen zuerst auf virtuellen Maschinen im Lab (oder Docker) sogenannte Splunk Forwarder einrichten, um Logfiles von virtuellen Maschine an eine zentrale Splunk oder Elasticsearch Installation zu schicken. Anschliessend müssen die Studenten innerhalb von Splunk oder Elasticsearch im Dashboard entsprechende Triggers konfigurieren, damit man die Logs systematisch auswerten und Attacken erkennen kann. Dann sollen die Studenten das Dashboard nutzen um einen Incident verarbeiten (Tickets) zu können.

Sample Scenario

Damit Sie sich ein mögliches Szenario besser vorstellen können, finden Sie nun ein Beispiel von einer konkreten Umsetzung

- Entwicklung Hacking Szenario das man letztlich während der Übung erkennen kann
  - CEO Fraud
  - Phishing Kampagne
  - Ransomware
  - E-Mail Trojaner
  - Active Directory Enterprise Admin Attack
- Bereitstellung Basic Cyber Challenge Plattform mit Sample Daten (so dass Daten im Portal vorhanden sind, bevor man startet)
- Einrichtung Log Forwarder Konzept
- Entwicklung Import Filter
- Entwicklung Dashboard
- Entwicklung Ticket-System um Alerts SOC Mitarbeitern zuzuordnen

Anforderungen an das Team

- Interesse an Docker
- Interesse an Security
- Interesse an Incident Handling
- Interesse an Entwicklung eines Angriffs, der dann via Cyber Attack Handling Plattform erkannt und bearbeitet werden kann

2 Management Summary

2.1 Ausgangslage

Informatikwesen und Sicherheit

Ein immer gegenwärtiger Begleiter der Informatik ist die Computersicherheit. Diese wird oft aus diversen Gründen ungenügend stark berücksichtigt und endet nicht selten in Hackerangriffe und bösartiger Software, welche die Geräte Infizieren.

Ein gutes Überwachungssystem zu haben und umso mehr das Wissen wie man eines bedient, ist leider jedoch relativ selten. Dem entgegenzuwirken ist auch nicht ganz billig, wenn man sich diverse Computersicherheitskurse und kommerzielle Angebote anschaut. Die Alternative sind sogenannte Open-Source Sicherheits-Plattformen, bzw. öffentlich zugängliche Projekte, welche auch Einblicke in die Codebasis geben und einem eben die Möglichkeit geben, eine Übersicht der Infrastruktur zu geben und mögliche Hackerangriffe abzuwehren. Der Haken hier ist, wie bereits schon erwähnt, das Know-How.

Situation an der OST

In den kommenden Semestern wird an der Ostschweizer Fachhochschule Campus Rapperswil-Jona verstärkt in Computersicherheitsmodule investiert bzw. unterrichtet. Eines von diesen Modulen heisst Cyber Defense und wird zum ersten Mal im Herbstsemester 2021 durchgeführt. Dieses Modul wird sich verstärkt mit der Verteidigung einer Infrastruktur beschäftigen und wie man Hackerangriffe erkennt. Für so ein Modul braucht es im Idealfall auch eine Trainingsumgebung für die Studenten, welche in Übungen das theoretische Wissen dann anwenden können.

Aus diesem Grund besteht Bedarf nach einer Lab Infrastruktur, wo das Erlernte möglichst unter Einfluss eines Angriffsszenarios, welches auch in der Industrie so vorkommt, angewendet werden kann.

Ziel des Cyber Defense Moduls ist den Studenten zu zeigen, wie man aktiv eine Infrastruktur überwachen kann und Gegenmassnahmen bei Hackerangriffen einleiten kann.

Lösungsansatz

In einer zu Beginn der Arbeit erhaltenen Infrastruktur soll eine neue Komponente eingebaut werden, welche eine Open-Source Plattform Lösung für die erwähnten Übungen enthalten soll. Diese neue Komponente mitsamt Plattform soll anhand einer prototypischen Implementierung einer möglichen Lösung untersucht werden. Im Idealfall soll die Infrastruktur nach der Implementierung stabil bleiben, gute Möglichkeiten für Erweiterungen geben und natürlich eine Lernumgebung den Studenten bieten.

2.2 Vorgehen

Ansatz

Im Rahmen dieser Arbeit soll eine neue Komponente zur bereits bestehenden Infrastruktur WinattackLab hinzugefügt werden und eine Lab Umgebung für die Studenten darauf kreiert werden. Prinzipiell soll eine Open-Source Plattformlösung fokussiert werden, welche mit der Infrastruktur zusammen stabil erstellt werden kann und in einem zweiten Schritt Übungsszenarien konzipiert werden, welche von der gewählten Plattform unterstützt werden.

Umfeld

Zum oben erwähnten Ansatz gehört einerseits ein Weg, die Infrastruktur bzw. das Lab aufzusetzen und anderseits eine Umgebung, um Aufgaben für die Studenten bereitstellen zu können. Beides wird über das Hacking-Lab der OST von Ivan Bütler, dem Betreuer dieser Arbeit, zur Verfügung gestellt. Während der Arbeit wird jedoch ohne direkte Interaktion des Hacking-Labs mit der Infrastrukturlösung interagiert, sondern getrennt davon an der Implementation und den Übungsszenarien gearbeitet.

Technologien

Die gewählte Open-Source Plattformlösung für das kommende Cyber Defense Modul ist Wazuh [1]. Wazuh ist eine Plattform, welche sich vor allem auf die Analyse der Geschehnisse in der Infrastruktur spezialisiert, jedoch weniger in ein aktives Eingreifen bei einem Hackerangriff.

Wazuh wird auf der neuen Komponente in der Infrastruktur, einem Linuxbetriebsystem, erstellt. Die Infrastruktur selbst wird über Terraform [2], ein Open-Source Infrastruktur als Code Software Tool, erstellt.

Neben Wazuh und Terraform wurden während des gesamten Projekts diverse weitere Technologien evaluiert, welche im Kapitel 3.2.5 festgehalten werden.

2.3 Ergebnisse

Erreichte Ziele

Das primäre Ziel dieser Arbeit, eine stabile Labumgebung für die Studenten des zukünftiges Moduls Cyber Defense zu kreieren, wurde erreicht. Diese kann über das Hacking-Lab der OST kreiert werden und mit den dort generierten Zugangsdaten können die Studenten die Infrastruktur erreichen.

Das sekundäre Ziel, Übungsszenarien zu konzipieren, wurde auch erreicht. Insbesondere wurden acht Aufgaben auf das Hacking-Lab der OST übernommen, welche sich mit der Einführung der Plattform Wazuh beschäftigen und erste Konfigurationsaufgaben zur Ereignisanalyse in der Infrastruktur bereitstellen.

Nicht erreichte Ziele

Ursprüngliches Ziel der Arbeit war eine Labumgebung zur Verfügung zu stellen, welche zum einen eine Übersicht der Geschehnisse in der Infrastruktur geben kann und zum anderen die Möglichkeit bietet, automatisierte Gegenmassnahmen bei einem Hackerangriff durchzuführen. Letzteres wurde innerhalb dieser Arbeit nicht

erreicht, da die Ziele der Arbeit zu Beginn des Semesters an ein Zweierteamprojekt adressiert waren.

Da der Fokus dementsprechend auf die Analyse, das Konfigurieren und das Filtern von Ereignissen verschoben wurde, lässt sich dies auch bei den Übungsszenarien feststellen. Es wurden dementsprechend keine Übungsszenarien zu automatisierten Gegenmassnahmen bei Hackerangriffen konzipiert.

Ausblick

Einer der massgeblichen Faktoren beim Auswahlverfahren der Plattform Wazuh war, dass diese Plattform relativ einfach mit einer anderen Plattform zusammenarbeiten kann, welche sich mehr mit automatisierten Gegenmassnahmen bei Hackerangriffen beschäftigt. Die Rede ist hier von der Plattform TheHive [3]. Als Weiterentwicklung könnte dementsprechend die Infrastruktur um die Plattform TheHive erweitert werden, wenn Wazuh im Bereich der automatisierten Gegenmassnahmen unzureichend wäre.

Es steht ausserdem nichts im Wege weitere Übungsszenarien zu konzipieren. Vor allem nicht, wenn die Infrastruktur um eine weitere Plattform erweitert werden würde.

3 Technischer Bericht

3.1 Einleitung

Problemstellung, Vision

In den aufkommenden Semestern werden an der OST, Campus Rapperswil-Jona, neue Cyber Security Module unterrichtet und eines dieser Module wird auf Cyber Defense fokussiert sein. Diese Arbeit soll ein Vorwerk legen, welches in diesem Cyber Defense Modul weiterverwendet werden kann. Fokus ist hierbei ein Security Operation Center, kurz SOC, für Ausbildungszwecke bereit zu stellen, welches auf dem OST Hacking-Lab gehostet wird.

Die Schwierigkeit liegt hierbei eine gute SOC Umgebung, unterteilt in SIEM und SOAR, aufzubauen, die den Benutzern die Möglichkeit gibt, sich in das Umfeld einzuarbeiten und den Erstellern die Möglichkeit gibt, eine Aufgabenstellung zu generieren. SIEM steht hierbei für “Security Information and Event Management” und SOAR für “Security Orchestration, Automation and Response”.

Um dieses Ziel zu erreichen, sollen diverse Cyber Security Plattformen analysiert werden, die möglichst einfach in Microsoft Azure Cloud aufgesetzt werden können. Eine zusätzliche Schwelle ist hier die Zusammenarbeit mit Terraform und das Inkludieren der SOC-Lösung innerhalb einer gegebenen Infrastruktur.

Bei einer erfolgreichen Arbeit wird dieses Projekt in den neuen Cyber Defense Modulen verwendet werden können und die Dozierende beim Unterrichten der Materialien unterstützen.

Ziele

Funktionalitäten des SOCs sollen sein:

- Ein Dashboard im Sinne eines SIEMs [4] zu erstellen und Schwellenwerte zu setzen
- Die Möglichkeit bei einem Security Incident einen Alarm zu senden und im Sinne eines SOARs [5] eine automatisierte Gegenmassnahme auszuführen.
- Ein Benutzer soll imstande sein, das SOC so zu editieren, dass die in der Aufgabenstellung vorgegebenen Security Incidents registriert und automatisiert gemeldet werden können.

Umgebung

Die schlussendliche Umgebung für die Studenten ist das Hacking-Lab der OST von Ivan Bütler. Innerhalb dieser Arbeit wird jedoch vor allem mit Terraform, verknüpft mit Microsoft Azure Cloud, und den zwei OS Systemen Windows 10 und Ubuntu 18.04 gearbeitet. Terraform selbst wird auf der virtuellen Maschine «Hackling-Lab LiveCD» [6] von der Firma Compass Security von Ivan Bütler ausgeführt.

Zusätzlich wird diese Arbeit sich mit Docker befassen und ein Grossteil der verschiedenen Ressourcen in Docker Container halten.

Um den Aufbau des SOCs möglichst zu vereinfachen, wird hier eine bereits bestehende Open Source Plattform Lösung gewählt.

Vorgehen

Die Prozedur beinhaltet eine längere Analyse- und Auswertungsphase der Technologien und eine relative frühe Besprechung der Black-Box Umgebungen, wie zum Beispiel das Hacking-Lab von Ivan Bütler. Dadurch soll ein besseres Bild erschaffen werden, was es genau an Aufwand für diese Arbeit braucht und wie viel Einarbeitungszeit es brauchen wird.

Nach der Auswertungsphase soll eine Open Source Lösung für ein Security Operation Center gewählt werden. Die nächsten Schritte beinhalten die Einbindung des SOCs in die übergebene Infrastruktur, welche durch Terraform aufgesetzt wird und die Erarbeitung diverser Übungsszenarien. Hierfür soll auch wieder genug Zeit eingeplant werden, da der Umgang mit Terraform neu ist.

Sobald die SOC-Plattform von anderen Clients erreichbar ist und auf das Dashboard geblickt werden kann, wird der Fokus stark auf Übungsszenarien verlegt. Diese sollen weiter ausgearbeitet werden und Wazuh und die bestehende Infrastruktur soll hierbei, wenn nötig, für die Übungen angepasst werden.
Gegen den Schluss der eingeplanten Projektzeit wird der Fokus auf das Polieren der erarbeitenden Übungsszenarien und Dokumentation verlegt.

3.2 Stand der Technik

3.2.1 Analyse SIEM und SOAR

Überblick

Wenn man von einer SOC-Plattform spricht, sollte man diese in einen SIEM (Security Information and Event Management) und einem SOAR (Security Orchestration, Automation and Response) Bereich grob aufspalten können. Die Trennung ist hierbei nicht immer klar definiert, da die Aufgaben von einem SIEM und SOAR übergreifend sein können. Innerhalb dieser Arbeit wurde jedoch eine grobe Trennung zwischen diesen zwei Bereichen vollzogen und diese wird in diesem Kapitel erläutert.

Übersicht SIEM & SOAR

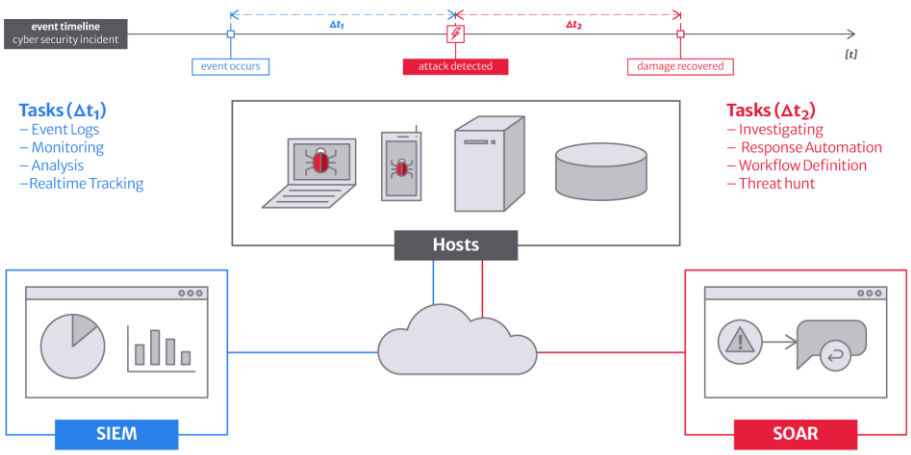


Abbildung 1: Übersicht SIEM & SOAR Seite

SIEM

«Security Information and Event Management» Lösungen sind an erster Stelle dafür zuständig Log- und Ereignisdaten aus verschiedenen Quellen wie Netzwerk, Server und Anwendungen zu sammeln und diese in Echtzeit zu aggregieren, identifizieren, kategorisieren und analysieren. Dabei kann und wird auch auf externe Informationen von Sicherheitssoftware-Anbietern zurückgegriffen, damit ein besseres Gesamtbild der Sicherheitslandschaft gebildet werden kann.

Zusätzlich dient die erwähnte Datenanalyse zur relativ raschen Erkennung, ob Sicherheitsprobleme vorhanden sind und dementsprechend die Möglichkeit einen Alarm zu senden. Insgesamt fördert eine gute SIEM-Lösung den Prozess Incident Response. [4] [7]

Verantwortungen von einem SIEM können grob wie folgt zusammengefasst werden:

- Macht Patternsuche in den Log-Daten nach Indikatoren für eine Cyberattacke
- Erstellt Korrelation der Event-Informationen zwischen Netzwerkgeräten und identifiziert abnormale Aktivität
- Alarmiert entsprechend der definierten Regeln des Security Analysten

SOAR

«Security Orchestration, Automation and Response» besteht im groben aus drei Bereichen der Welt des Cyber Defenses:

- Threat and Vulnerability Management
- Security Incident Response
- Security Operations Automation

SOAR sammelt ähnlich wie SIEM auch Daten von diversen Quellen, jedoch integriert SOAR mehr interne und externe Applikationen, die ein Automatisiertes eingreifen bei einem Sicherheitsproblem ermöglichen. Kurz gesagt spezialisiert sich SOAR vor allem in das automatisierte eingreifen, wenn ein Security Incident stattfindet und SIEM auf eine Aufbereitung und Analyse der diversen Daten, welche erst das automatisierte eingreifen von SOAR ermöglichen. [5]

Verantwortungen von einem SOAR können grob wie folgt zusammengefasst werden:

- Alarm-Meldungen der integrierten Plattformen sammeln und an einer zentralen Stelle speichern
- Gibt die Möglichkeit einzelne Vorfälle genau zu untersuchen, einzustufen und nachzuvollziehen
- Ermöglicht die Definition von automatisierten Workflows als Abwehrreaktion auf mögliche zukünftige Bedrohungen
- Stellt Playbooks zur Verfügung zur Abwehr von spezifischen Bedrohungen

3.2.2 Bestehende Open-Source Cyber Security Plattformen

Überblick

Im Rahmen dieser Arbeit wurden diverse Cyber Security Plattformen analysiert, welche sich für die oben erwähnten SIEM- und SOAR-Prozesse eignen. Dabei wurden jedoch nur zwei dieser Plattformen nach einer rein Web- und Dokumentations-Analyse per «Hands-On» weiter untersucht. Grund für diese Entscheidung ist vor allem die relativ knappe Zeitspanne und die Vielfalt der zu analysierenden Open

Source Plattformen. Zusätzlich wurden auch viele der Plattformen während der Dokumentations-Analyse wegen schlechter Dokumentation und unpassenden Projekzustandes von der für diese Arbeit möglichen Kandidatenliste eliminiert.

3.2.2.1 Reine Dokumentation- & Web-Analyse von Plattformen

GRR

GRR Rapid Response [8] ist ein Incident Response Framework und ist vor allem auf Remote Live Forensics fokussiert. Es besteht aus zwei Teilen: GRR Client und GRR Server, beide basieren auf Python. Der Status von GRR hatte jedoch breaking builds bei der Untersuchung und machte allgemein einen eher minderwertigen Eindruck als mögliche SOAR Plattform oder Tool für diese Arbeit.

HELK

HELK [9] ist eine Open Source Plattform, die sich vor allem darin spezialisiert hohe analytische Fähigkeiten zur Verfügung zu stellen und in einem grösseren Environment gut skalliert. In erster Linie wurde es jedoch nicht als SIEM oder SOAR entwickelt, sondern diente primär zur Recherchen-Hilfe. Die gelisteten Tools sind unter anderem Kibana, KSQL, GraphFrames und Jupyter. Jedoch ist das Projekt zum Zeitpunkt dieser Arbeit noch im Alpha-Status und damit erübrigte sich die weitere Analyse, da das Risiko von möglichen «breaking changes» auf der Plattform minimal gehalten werden soll.

OpenEDR

Open Source Endpoint Detection and Response [10], kurz OpenEDR, ist eine Open Source Cyber Security Plattform die sich in erster Linie als SIEM versteht. Da jedoch bei der Analyse früh erkannt wurde, dass man auf jeder überwachten Maschine Filebeat (Teil von Elastic Stack, siehe Kapitel 3.2.5) installieren muss und jede Filebeat-Instanz angepasste Einstellungen gebrauchen würde, wurde diese Plattform als unpassend für diese Arbeit eingestuft.

MISP

MISP [11] ist eine Open Source Lösung, die zum einen das Sammeln, Speichern und Teilen von Cyber Security Indikatoren unterstützt, bzw. Bedrohungen von Security Incident Analysen und Malware Analysen im Auge behält. Zusammengefasst ist MISP eine Art «Sharing-Plattform» über Sicherheitszwischenfälle. Die Analysen sind dabei vom aktiven Informationenaustausch abhängig, womit sich diese Plattform alleinstehend für diese Arbeit nicht eignet.

DetectionLab

DetectionLab [12] ist spezialisert auf die Schnelle Bereitstellung einer Windows-Domain, die diverse Security-Tools und best practices zu System Logging Konfigurationen enthält. Es wird vermerkt, dass man das Lab nie an wichtige Netzwerke verbinden sollte, da es nicht hardened [13] ist bzw. insecure designed wurde und spezifisch zu Übungszwecken existiert.

Die Voraussetzungen sind [14]:

- Auf Azure per 4 separaten Instanzen
- Azure Subscription
- Terraform v12
- *az* die Azure Kommandozeile [15]

- Ansible

Um einen Überblick zu verschaffen, werden nachfolgend die wichtigsten Features und Tools gemäss der DetectionLab GitHub Seite [12] aufgeführt:

- Microsoft Advanced Threat Analytics
- Splunk Forwarder
- Custom Windows Auditing Configuration via GPO
- Palantir's Windows Event Forwarding [16]
- Powershell Transcript Logging
- Osquery konfiguriert um zu einem Fleet [17] Server via TLS zu verbinden
- Sysmon installiert & konfiguriert [18]
- Autostart Items werden zu Windows Event Logs [19] geloggt.
- Zeek & Suricata zur Netzwerküberwachung und Alarmierung
- Apache Guacamole damit Hosts einfach über den lokalen Browser erreichbar sind

DetectionLab wurde als möglicher Kandidat genauer betrachtet, jedoch schlussendlich verworfen: Es ermöglicht zwar eine einfache Einbindung von zusätzlichen Tools oder individuellen Einrichtungen, aber diese Abhängigkeit von extern gehaltenen Tools kann auch ein gewisses Risiko mit sich bringen. Eine gute Dokumentation ist hierbei auch nicht zu finden, was sich bei einem möglichen Problem mit der Plattform rächen würde.

Security Onion

Security Onion [20] ist ein auf Linux-Distribution basierte Intrusion Detection und Security Monitoring System. Wichtige Elemente der Infrastruktur sind Docker, Elastic Stack, Redis, ELK Stack, Grafana. Security Onion inkludiert ausserdem die SOC Plattformen Wazuh und TheHive und es gibt diverse Architektur-Optionen für simple oder komplexere Infrastrukturen. Dokumentation ist in Ordnung bis gut, jedoch geht diese Plattform Richtung Commercial bzw. Enterprise.

Diese Plattform wurde immer wieder als mögliche Kandidatin für diese Arbeit betrachtet, jedoch sprachen zwei Tatsachen schlussendlich dagegen. Erstens ist bereits jetzt ersichtlich, dass ein Teil der Dokumentation und Features Richtung Commercial Software geht und weg von Open Source, was auf lange Dauer potentiell mehr sein wird. Zweitens wurde nach dem Wechsel auf eine Einzelbachelorarbeit der Fokus mehr auf die SIEM-Funktionalitäten von einem SOC verschoben. Da SecurityOnion, wegen der Einbindung von zwei SOC Plattformen und diversen anderen Technologien, komplexer ist und die inkludierte Plattform TheHive kaum zum Einsatz kommen würde, wurde diese Plattform als möglicher Kandidat verworfen.

3.2.2.2 Zusätzliche Plattform Hands-On Analyse

Überblick

Zusätzlich zur theoretischen Analyse der Security Plattformen wurde eine Hands-On Analyse der Plattformen TheHive und Wazuh durchgeführt. Zu diesem Zweck wurden die Plattformen aufgesetzt und in Betrieb genommen. An den laufenden Plattformen wurden Versuche durchgeführt, die sich mit den später entstandenen Übungsszenarien einigermassen decken.

TheHive

TheHive [3] Open-Source Security Incident Response Platform benutzt Cortex [21] zur Datenanalyse und Incident Response. Cortex ist eine auf künstliche Intelligenz basierte Plattform für Continuous-Security. TheHive benutzte diese Plattform, um sogenannte Observables zu analysieren. Observables sind beispielsweise IP oder E-Mail-Adressen. Ausserdem benutzt TheHive MISP [11], den Standard für Threat Information Sharing.

Im Grunde genommen, eignet sich TheHive sehr für einen SOAR Einsatz, jedoch ist das automatisierte Aufsetzen für 'nur' einen SIEM Einsatzes zu aufwendig. Grund dafür sind die diversen Komponenten, die auf den Systemen jeweils installiert werden müssen.

Benutzte Hauptsprachen:

- Scala
- Python
- Frontend benutzt AngularJS

Erwähnenswertes:

TheHive hat eine gute Dokumentation, jedoch ist sie sehr verteilt und nicht an einem Ort zusammengefasst. Die Dokumentation enthält jedoch einen Installation-Guide für RPM-, DEB- und binary-package und stellt auch ein Docker Image zur Verfügung. Ausserdem gibt es eine up-to-date Training VM, welche auf Ubuntu 20.04 läuft. Es benutzt Elasticsearch und unterstützt diverse Authentisierungsmöglichkeiten. Interessant ist, dass die Community bzw. der Support über Gitter erreichbar ist. [22]

Wazuh

Wazuh [23] ist eine Open Source Security Plattform, die im Sinne eines SIEMs und EDRs (Endpoint Detection & Response) ihr Schwergewicht in die Überwachung der Infrastruktur, Erkennung von Sicherheitsrisiken und Incident Response stellt.

Es existiert eine kostenpflichtige «Wazuh as a Service» Cloud-Version, jedoch gibt es auch ein «Wazuh Lab Environment» [24], welches aber nur auf Amazon Cloud getestet wurde. Es soll aber auch möglich sein, mit Azure Cloud dieses Environment aufzusetzen.

Ein sogenannter Wazuh agent wird auf die zu überwachenden Maschinen installiert und kommuniziert mit dem Wazuh manager, der auf einem Server installiert ist. Damit die Daten übersichtlich repräsentiert sind, werden sie weiter an Komponenten von Elastic Stack, siehe Kapitel 2.2.4, gesendet und dank Kibana schlussendlich in einem Dashboard dargestellt.

Die untenstehende Abbildung aus der Wazuh-Dokumentation stellt eine mögliche Architektur dar:

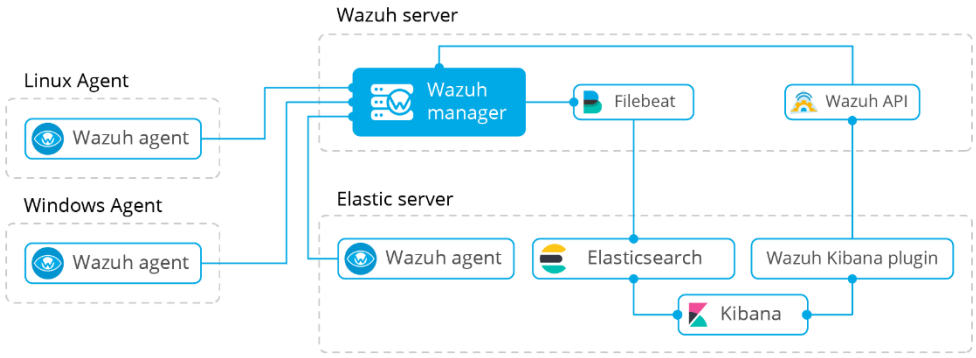


Abbildung 2: Beispiel Architektur von Wazuh Lab Environment

Nachfolgend die wichtigsten Features, Tools und Begrenzungen von Wazuh:

- Gute Dokumentation
- «Nur» SIEM fokussiert
- «Nur» x86-64 Architektur (Standard Intel/AMD 64-bit Prozessoren)
- Integriert mit Elastic Stack [25]
- Mit Ansible, Puppet [26] oder Virtual Machine deploybar
- Kann Microsoft Azure Infrastrukturen überwachen
- Hat Windows, Linux und diverse andere Agents
- Open-Source Community ist aktiv [1]
- Deployment vom Wazuh-Server kann All-in-one oder Distributed sein [27]
- Hat kein Ticket-System

3.2.3 Wazuh und TheHive Features

Überblick

Die zwei Hauptkandidaten für diese Arbeit sind schlussendlich Wazuh und TheHive. Da Wazuh den Fokus auf SIEM und TheHive auf SOAR setzt und der Schwerpunkt dieser Arbeit auch auf SIEM verlagert wird, ist Wazuh die für diese Arbeit benutzte Plattform. Die Gewichtung wird dabei durchgehend auf eine gute Dokumentation und die Möglichkeit über Docker die Plattform out-of-the-box benutzen zu können gelegt.

Feature Gegenüberstellung

Um ein besseres Bild der diversen Features zu bekommen, werden diese in der folgenden Tabelle gegenübergestellt. Insbesondere in was für Features diese Plattformen sich unterscheiden und dementsprechend was für Aufgaben genau schlussendlich mit der jeweiligen Plattform kreiert werden können. Dabei ist noch zu vermerken, dass die hier gelisteten Features jeweils direkt in den jeweiligen Dokumentationen gesucht wurden. Wenn kein Eintrag gefunden wurde oder nur grob etwas in einem Blog oder Gitter beschrieben wurde, wurde das Feature mit '-' gekennzeichnet.

Feature	Wazuh	TheHive
Analyzed Feedback	-	Cortex
Ticketsystem	-	Dashboard
Agents	Wazuh Agents	-
Rules Classification	16 vordefinierte, die modifizierbar sind	-
Traffic Generator	Custom Python Script	Custom Python Script
File Integrity Monitoring	FIM module	-
AD Event Forwarding	Eventlog/-channel	-
Daten forwarder	Splunk	-
Proxy Log Forwarder	Logstash	Logstash
E-Mail Alerts	Postfix	TheHive4py
E-Mail Log Forwarder	Logstash	Logstash
Notification	Slack	TheHive4py
Tool Login via AD	-	Cortex
Network Intrusion Detection System	Suricata	Suricata
Web Server Log Forwarding	Logstash	Logstash

Tabelle 1: Feature-Gegenüberstellung von Wazuh & TheHive

3.2.4 Gewählte SOC-Plattform Wazuh

Überblick

Hier wird ein Überblick der gewählten SOC-Plattform Wazuh gehalten und warum diese gewählt wurde.

Warum Wazuh?

Wazuh war schlussendlich der ideale Kandidat, da es eine SIEM fokussierte Plattform ist, eine gute Dokumentation hat, eine aktive Community, diverse Use Cases im Security-Sinne bietet und einfach über Docker aufsetzbar ist. Ausserdem sind unter den Security Use Cases ein paar wenige SOAR fokussierte Use Cases zu finden, welche eventuell für den SOAR Stretch Goal zum Einsatz kommen könnten.

Security Use Cases

Wazuh [28] bietet die folgenden Use Cases im Security-Sinn, welche sich dementsprechend auch zur Verwendung in den Übungsszenarien anbieten:

Log Data Analysis: Automatisierte Log Management und Analyse zur schnelleren Bedrohungsentdeckung.

Rootkits Detection: Wazuh scannt auf Kernel- und Benutzerbereichsebene regelmässig nach Rootkits.

Configuration Assessment: Geht um Automated Security Configuration Assessment (SCA) welches zentral für die Verbesserung der Sicherheitslage des Unternehmens und Verkleinerung der Angriffsfläche ist. Das Wazuh SCA Modul scannt das überwachte System regelmässig und berichtet Fehlkonfigurationen.

Vulnerability Detection: Wazuh unterhaltet eine up-to-date Liste von allen auf dem Endpoint installierten Applikationen, auf welchen ein Wazuh Agent installiert ist, und korreliert diese Informationen mit der National Vulnerability Database (NVD).

Containers Security Monitoring: Wazuh kann, auf Infrastruktur oder Container Level, Anzeichen von Sicherheitsvorfällen bei Docker Anwendungen erkennen und melden.

File Integrity Monitoring (FIM): Überwacht Dateien auf System- und Anwendungsebene und meldet gegebenenfalls Veränderungen.

Active Respone: Wazuh kann eine automatisierte Reaktion auf eine Bedrohung hin ausführen, wenn diese erkannt wird. Die möglichen Reaktionen sind im folgenden Bereich anwendbar: Blockieren von Netzwerkverbindungen, laufende Prozesse anhalten und bösartige Dateien löschen. Ausserdem können massgeschneiderte Python, Bash oder PowerShell Skripts ausgeführt werden.

System Inventory: Sammelt Hardware- und Softwareinformationen vom überwachten System und hilft beim Patch-Management.

Cloud Security Monitoring: Wazuh stellt Bedrohungserkennung, Konfigurationskonformität und kontinuierliche Überwachung für Multicloud und Hybride Umgebungen zur Verfügung. Dabei werden zwei Level überwacht: Endpoint-Level und Cloud-Infrastruktur-Level.

Regulatory Compliance: Die sogenannten Wazuh-Rules werden gegen Compliance-Anforderung abgebildet, damit diese möglichst up-to-date sind. Die unterstützten Standards sind:

- Payment Card Industry Data Security Standard (PCI DSS)
- General Data Protection Regulation (GDPR)
- NIST Special Publication 800-53 (NIST 800-53)
- Good Practice Guide 13 (GPG13)
- Trust Services Criteria (TSC SOC2)
- Health Insurance Portability and Accountability Act (HIPAA)

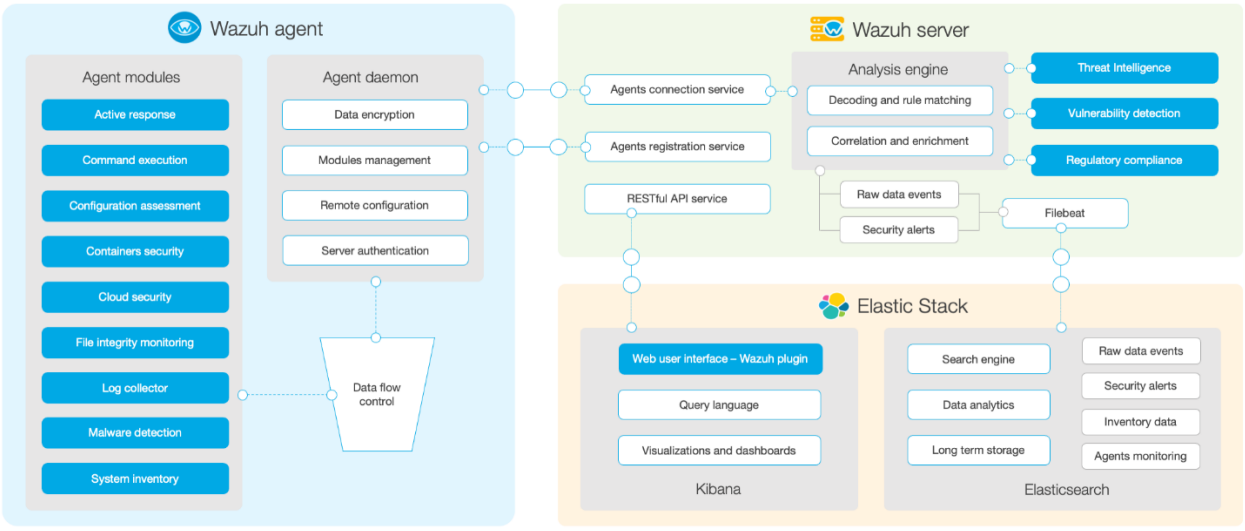


Abbildung 3: Wazuh Komponenten und Datenfluss von Wazuh Dokumentation

3.2.5 Analyse von Tools

Überblick

Neben den Plattformen wurden während des gesamten Projektes diverse andere Tools untersucht, welche im Verlauf der Arbeit angefallene Aufgaben übernehmen. Beispielsweise handelt es sich dabei um Mail-Server, NIDS, Elastic Stack und Logging-Tools.

Aus Zeitgründen konnte hierbei nicht für jede Aufgabe eine Evaluationsliste geführt werden. Deshalb beschränkt sich dieses Kapitel auf eine kurze Übersicht und die tatsächlich ausgewählte Technologie.

Arkime

Arkime [29], früher bekannt als Moloch, ist ein Open Source Tool welches sich in Paket- bzw. Traffic-Analyse spezialisert. Es benutzt im Grunde drei Komponenten:

- Capture: Untersucht Netzwerk, schreibt/liest PCAP Dateien und sendet Metadaten an Elasticsearch
- Viewer: Auf jeder erfassenden Maschine läuft eine Node.js Applikation. Diese dient als Webinterface und verarbeitet die PCAP Dateien.
- Elasticsearch: Die Datenbank von Arkime

Dieses Tool macht an sich einen guten Eindruck und hat auch eine aktive Community hinter sich stehen zum Zeitpunkt dieses Projektes.

Suricata

Suricata [30] ist ein Open-Source, rasantes und robustes Network Intrusion Detection System (NIDS). Neben NIDS bietet es auch ein Network Intrusion Prevention System (NIPS). Damit kann es neben Analysen am Datenverkehr auch direkt Pakete blockieren. Suricata würde innerhalb der Übungen zum Einsatz kommen, wenn es darum geht bösartige Netzwerkaktivitäten zu erkennen.

Zeek Zeek [31] ist ein Open-Source Network Security Monitor (NSM) Framework. Netzwerk Pakete werden mit pcap gefangen und weiter zu einem Event System transferiert, welche diese akzeptiert oder ablehnt. Ehemals unter Bro bekannt, entwickelt in 1994 dank Vern Paxson. Zeek ist eine Alternative zu Suricata.

Postfix Postfix [32] ist ein für UNIX-Systeme gedachte Mail-Server, welcher bei IBM-Research von Wietse Venema erschaffen wurde. Zurzeit wird er bei Google weiterhin von Venema unterstützt.

MailCatcher MailCatcher [33] ist ein sehr simpler SMTP-Server, welcher jegliche Nachricht fängt und in einem User Interface anzeigt, solange die Nachricht an ihn gesendet wird. Innerhalb dieser Arbeit wird MailCatcher zu Übungszwecken verwendet.

MailHog MailHog [34] ist ein einfach aufzusetztender SMTP-Server welcher als Alternative zu MailCatcher dienen könnte.

Splunk Splunk [35] ist zwar eine Plattform wird jedoch in dieser Arbeit unter den Tools gelistet, da sie nicht eine Security basierte Plattform ist, sondern eine Log-, Monitoring- und Reporting Plattform. Ergo wird diese Plattform höchstens als Tool für die ausgewählte Cyber Security Plattform benutzt.

The Elastic Stack The Elastic Stack [25], oder auch ELK Stack benannt, enthält die Kernprodukte Elasticsearch, Kibana, Beats und Logstash. Elasticsearch ist eine Suchmaschine und Analytics Engine, Kibana ermöglicht die Visualisierung von Daten, mit Beats können die Daten tailliert werden und Logstash dient als serverseitige Datenverarbeitungspipeline.

terraform graph Über *terraform graph* [36], ein Befehl für die Terraform-Kommandozeile, kann eine visuelle Repräsentation von einer Konfiguration oder einem execution plan von einem Terraform-Deployment gemacht werden. Der DOT-Output wird mit GraphViz [37] zu Diagrammen umformatiert. An sich ist dieses Tool sehr nützlich, solange nicht zu viele Komponenten Teil der Infrastruktur sind. Das erstellte Diagramm wird nämlich dementsprechend grösser und komplexer und ab einem gewissen Punkt keinen guten Überblick der Infrastruktur mehr geben kann.

TheHive4py TheHive4py [38] ist ein Python API-Client für die Plattform TheHive. Es erlaubt es Analysten Alarme von verschiedenen Quellen zu TheHive zu schicken. Diese Alarme können dann in sogenannte Cases importiert werden, indem vordefinierte Templates benutzt werden. Bei SOAR Funktionalitäten bei der TheHive Plattform ist dieser API-Client unabdinglich.

Ansible Ansible [39] ist ein Open Source Automatisierungs-Tool. Es kann unter anderem Systeme konfigurieren, Software deployen und Continuous Deployment orchestrieren. Anders ausgedrückt, kann es Ad-hoc-Kommando-Ausführungen und Softwarekonfigurationsmanagement kombinieren. Innerhalb dieser Arbeit ist Ansible

für die Playbooks relevant, die im Zusammenhang mit Incident Response in den Einsatz kommen.

Erwähnenswertes:

- Benutzt YAML zur Formulierung
- Basiert auf Python
- Playbooks werden damit definiert

Sysmon

Sysmon [40] wird als Schweizer Sackmesser für System-Monitoring bezeichnet. Es wird so bezeichnet, weil es diverse Monitoring-Tasks sehr präzis ausführen kann. Die Schwierigkeit ist hier nur das richtige Utensil zu benutzen, bzw. die richtige Konfiguration vorgenommen zu haben. Sysmon ist sehr mächtig und wird innerhalb dieser Arbeit in den Übungen verwendet.

Mimikatz

Mimikatz [41] ist ein Open-Source-Programm und Hackertool, mit dem Benutzer sich Authentifizierungsdaten wie Kerberos-Tickets anzeigen lassen und diese speichern können. Mimikatz wird innerhalb von Übungen verwendet, um die nötigen Konfigurationen zu verstehen, welche gemacht werden müssen, damit dessen Aktivität erkannt wird.

3.2.6 Analyse von Infrastruktur-Technologien

Überblick

Im Rahmen dieser Arbeit wurden zusätzliche folgende Technologien untersucht. Diese sind nicht direkt für ein Security Operation Center vonnöten, jedoch für das Deployment der Infrastruktur und deren Interaktion unabdinglich. Sie kommen also in dem Teil der Arbeit zum Zug, wo es um das Aufsetzen einer funktionstüchtigen Lab-Umgebung geht.

Terraform

Terraform [2] ist eine Open-Source-Infrastruktur von HashiCorp die per Command Line Interface (CLI) hunderte von Cloud-Services-Workflows leitet bzw. leiten kann. Sie codiert Cloud-APIs in Konfigurations-Dateien um und diese werden wiederum benutzt, um Infrastrukturen zu erstellen oder zu zerstören. Terraform benutzt dafür eine eigene Sprache [42], die aus einer simplen Syntax besteht. Darunter definiert sind «Blocks», «Arguments» und «Expressions». Die Sprache ist deklarativ und zieht nur die impliziten und expliziten Beziehungen zwischen den Ressourcen in Betracht, wenn es darum geht die Reihenfolge der Operationen zu determinieren. Dies bedeutet, dass die Reihenfolge der «Blocks» und der Dateien in der Regel irrelevant ist. Die Syntax wiederrum ist das A und O von Terraform. Wenn diese nicht nach der jeweiligen Ressourcen Syntax Konfiguration übereinstimmt, funktioniert das Deployment nicht.Erwähnenswertes:

Mit Terraform wird in dieser Arbeit die Infrastruktur deployed, d.h. dass das Security Operation Center auf einer Maschine gehosted wird, welche durch Terraform aufgesetzt wird. Terraform hat zwar eine sehr gute Dokumentation, jedoch ist diese auch sehr gross und, wenn man neu mit dieser Technologie interagiert, nicht ganz einfach spezifische Konfigurationen nachzuvollziehen. Neben der Dokumentation von

Terraform, bietet die Azure Cloud einen Installations-Guide zur Terraform-Verknüpfung und auch weitere Konfigurationsdokumentation. [43]

Microsoft Azure

Microsoft Azure [44], ursprünglich «Project Red Dog» genannt, ist eine Cloud von Microsoft. Die jeweiligen Ressourcen, die mit Terraform deployed werden, können nur dank der Verknüpfung zu einem Azure Cloud Konto erstellt werden.

Innerhalb dieser Arbeit wird ausserdem mit der Cloud nur über das az-Command-Line-Interface-Tool interagiert. Dies ist vom Betreuer dieser Arbeit so gewünscht worden.

Docker

Docker [45] dient zur Isolierung von Anwendungen und ist eine PaaS (Platform as a Service). Die Isolierung erreicht es dank Container Virtualisierung.

Innerhalb dieser Arbeit wird stark mit Docker gearbeitet, wobei das am häufigsten verwendete Tool Docker Compose ist. Zum Beispiel werden das SOC Wazuh und die Elastic Stack Komponenten zu Services über Docker Compose erstellt.

traefik

Traefik [46] wird vor allem im Zusammenhang mit Docker oder Kubernetes gebraucht. Es wird als ein Open-Source Edge Router bezeichnet, i.e. es bekommt Anfragen vom System und findet heraus, an welche Komponenten es diese weiterleiten soll.

In dieser Arbeit wird traefik beispielsweise für Anfragen an die Wazuh und Attack-Launcher Services verwendet, damit diese Services nichtmehr über eine IP-Adresse und Port Nummer aufgerufen werden müssen im Browser.

3.3 Lösungsansatz

Konzept für die Umsetzung

Anhand eines Prototyps soll untersucht werden, ob es möglich ist, ein Security Operation Center zu Lernzwecken im Hochschulumfeld zu bedienen. Dazu sollen Übungsszenarien konzipiert werden, die einer Lernförderung der Studenten im aufkommenden Fach Cyber Defense dient.

Für die Umsetzung werden zu Beginn ein Arbeitsplan und diverse Kriterien an Plattformen und Tools definiert, wobei die Analyse-Prozedur von den Plattformen eine stärkere Gewichtung auf deren Dashboard-Übersichts-Funktionalitäten hat und der Qualität deren Dokumentation gewichtet ist. Ein zusätzliches starkes Kriterium ist die Möglichkeit die Plattform über Docker oder Ansible einzuschleusen und zu konfigurieren.

Nach der Analyseprozedur wird der Fokus auf die Integration der gewählten Plattform in die gegebene Infrastruktur verschoben. Hierfür wird ein Proof of Concept (PoC) erstellt. Das Ziel beim PoC ist das erfolgreiche Aufsetzen eines Security Operation Centers innerhalb der WinattackLab-Infrastruktur. Ausserdem sollen mögliche Angriffsszenarien erdacht werden, um der Konzipierung der Übungsaufgaben zu unterstützen.

Bei einem erfolgreichem PoC wird die Infrastruktur für die diversen Übungsszenarien erweitert. Hierbei wird beachtet, dass mit simplen Aufgabenszenarien begonnen wird, damit mehr Erfahrung mit der bestehenden Infrastruktur gesammelt werden kann und gleichzeitig relativ simple Einstiegsübungen für die Studenten entstehen.

Gegen Ende des Arbeitsplans wird der Fokus auf das Polieren der Implementation und Dokumentation verstärkt und keine neue Komponente zur Infrastruktur hinzugefügt.

Umfeld

Die Studenten interagieren über das von der OST und Ivan Bütler zur Verfügung gestellte Hacking-Lab mit dem Security Operation Center. Die Implementationslösung dieser Arbeit wird jedoch im Umfeld von Terraform, Docker und der Azure Cloud sein, wobei Terraform selber auf einer Linux-VM ausgeführt wird.

Da das Projekt in Zukunft nur im Rahmen der Fachhochschule OST eingesetzt werden soll, waren deren Richtlinien massgebend für das Projekt.

Technologien

Basierend auf der Analyse wird mit der Plattform Wazuh gearbeitet. Mit der Verwendung von Wazuh kommen diverse damit zusammenhängende Tools auch zum Einsatz. Der Elastic Stack spielt hierbei eine grössere Rolle.

Neben der SOC-Plattform wird stark mit dem Infrastruktur-als-Code-Tool Terraform gearbeitet, welche auf einer Hacking-Lab LiveCD ausgeführt wird. Die zu Beginn der Arbeit erhaltenen Infrastruktur, enthält Linux- wie auch Windows-Clients und Server. Hier ist neben Netzwerkkonfigurationswissen zusätzlich ein breites Wissen in PowerShell- und Bash-Scripting nötig.

Da die diversen Tools, wie auch die Plattform, über Docker laufen, wird auch ein breites Wissen in der Konfiguration von Docker-Dateien verlangt.

Alles in allem ist eine breite Palette an Know-How in diversen Bereichen der Netzwerk-, Security-, Skripting-, Docker- und Cloud-Welt nötig.

3.4 Resultate

Zusammenfassung

Die nach der Gruppentrennung neu definierten Ziele wurden zum Grossteil erreicht. Primäres Resultat des Projekts ist das SIEM-fokussierte SOC Wazuh, welches in der zu Beginn der Arbeit erhaltenen Infrastruktur, auf einer neuen Ressource, einem Ubuntu Server, mitsamt der Infrastruktur deployed werden kann. Das Deployment funktioniert hierbei auch über das Hacking-Lab der OST.

Ausserdem wurden fünf grössere Übungsszenarien konzipiert, wobei die ersten drei Szenarien bereits zu acht Aufgaben auf das Hacking-Lab übertragen wurden.

Eine genauere Analyse davon, welche Ziele erreicht und nicht erreicht wurden, sowie einen Ausblick auf mögliche Weiterentwicklung, wird im Kapitel 4.5 beschrieben.

4 Projektdokumentation

4.1 Anforderungsspezifikation

Definition

In der Elaborationsphase wurde insbesondere Analysen von bestehenden SIEM- und SOAR-Plattformen vollzogen. Dadurch konnte die erste Iteration der Anforderungsspezifikation erfolgen. Die Ergebnisse sind insbesondere die Use Cases, nichtfunktionalen Anforderungen und Diagramme zur Architektur und der Prozedur.

4.1.1 Anforderungen an die Bachelorarbeit

Anforderungen

Zu Beginn des Projekts wurden folgende Anforderungen übergeben:

- Die Arbeit bzw. das SOC Lab soll auf dem Hacking-Lab der OST benutzt werden können
- Die benutzte Cloud soll Azure sein
- Das Deployment der Infrastruktur wird mit Terraform erzielt
- Eine bereits vorhandene Open Source SIEM- oder SOAR-Plattform soll als Basis dienen

4.1.2 Use Cases (Software-Engineering)

Überblick

Im Rahmen dieser Arbeit sind fünf Use Cases und fünf Rollen bzw. Actors entstanden. Der Begriff Use Case und Actor wird hierbei innerhalb des Software-Engineering-Umfeldes behandelt und nicht innerhalb des Netzwerksicherheits-Umfeldes.

Der Fokus innerhalb dieser Arbeit wird zudem ausserordentlich auf die Use Cases und Rollen verlegt, die nicht SOAR-bezogen sind.

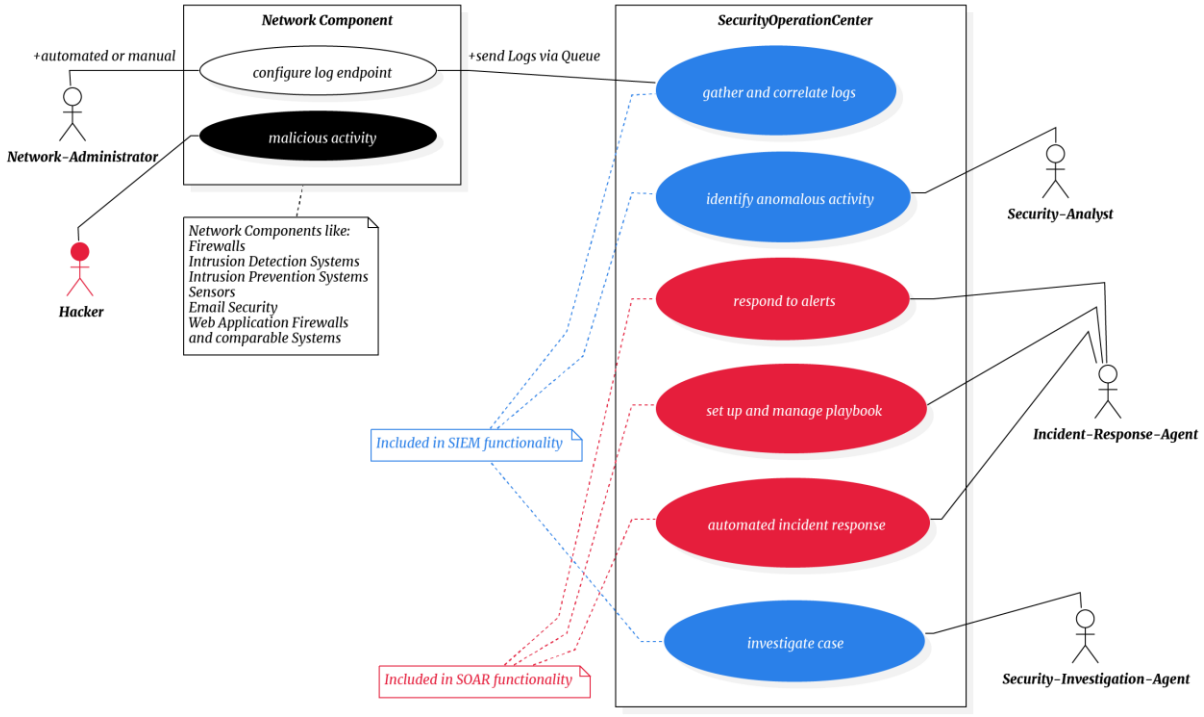


Abbildung 4: Use Cases Diagramm

Actors

Die Use-Case-Analyse erbrachte einen Actor Student, der diverse Rollen vertritt. Die fünf Rollen sind: Security-Analyst, Incident-Response-Agent, Security-Investigation-Agent, Hacker und Netzwerk-Administrator

UC 1: Abrufen Datenübersicht

Ein Student kann in der Rolle Security-Analyst eine Übersicht der aggregierten und analysierten Daten in Form eines Dashboards bekommen.

UC 2: Alert erhalten

Ein Student soll in der Rolle Security-Investigation-Agent auf einem Dashboard eine Notifikation eines Sicherheitsvorfalls erhalten können, welches er weiter untersuchen kann.

UC 3: Incident Response definieren

Ein Student soll in der Rolle Incident-Response-Agent in der Lage sein, über ein Dashboard ein Incident-Response zu einem Sicherheitsvorfall definieren zu können.

UC 4: Malicious Activity starten

Ein Student soll in der Rolle Hacker eine Datei manipulieren können, die im AD zu finden ist oder eine böswillige Aktivität starten können, wie zum Beispiel eine SSH-Brute-Force-Attacke.

UC 5: Endpoints Konfigurieren

Ein Student soll in der Rolle Netzwerk-Administrator End-Points konfigurieren können, um Logs und Events an das Dashboard senden zu können.

4.1.3 User Stories

Überblick

In der Elaboration-Phase wurden acht User Stories definiert. Diese sind von der folgenden Tabelle zu entnehmen. Fokus wird dabei vor allem auf die ersten vier User Stories gelegt, da diese SIEM-fokussiert sind und eine hohe Wichtigkeit für dieses Projekt haben. Die restlichen vier User Stories werden als Stretch Goals definiert, da sie, mit Ausnahme der Nummer fünf, stark SOAR-fokussiert sind.

Nr.	User Story	Akzeptanzkriterium	Wichtigkeit
1	Als Netzwerk-Administrator möchte ich einen Agent auf einem Host im AD hinzufügen können, welcher mit der SOC-Plattform kommuniziert.	Das Hinzufügen des Agents ist abgeschlossen, wenn die Kommunikation mit dem SOC Server ersichtlich ist.	Sehr hoch
2	Als Netzwerk-Administrator und Hacker möchte ich den Kommunikationsfluss steuern können.	In Form von z.B. einem Tomahawk-Skript kann Netzwerkverkehr stimuliert werden, um einen echten Kommunikationsfluss zu simulieren	Sehr hoch
3	Als Hacker möchte ich auf eine Datei zugreifen oder diese modifizieren können, die im AD zu finden ist, um möglichen Schaden dem System hinzuzufügen.	Sobald mindestens eine Datei manipuliert wurde und dies, wenn diese Datei untersucht werden würde, erkannt wird.	Hoch
4	Als Security-Analyst möchte ich eine Übersicht der Events gewinnen können, um die Aktivitäten auf dem Netzwerksystem zu erkennen.	Auf einem Dashboard ist ersichtlich, welche Events in Real-Time wo geschehen.	Sehr hoch
5	Als Security-Investigation-Agent möchte ich einen Vorfall untersuchen können.	Es ist möglich einen Vorfall als false- oder true-positive Kategorisieren zu können.	Tief
6	Als Incident-Response-Agent möchte ich Playbooks definieren können, um bei einem Sicherheitsvorfall diese verwenden zu können.	Es ist möglich Playbooks zu definieren, die bei einem Vorfall eingesetzt werden können.	Mittel
7	Als Incident-Response-Agent möchte ich einen Vorfall behandeln können, um das System zu schützen.	Es ist möglich bei einem Vorfall manuell ein Playbook oder eine Gegenmassnahme zu starten.	Mittel
8	Als Incident-Response-Agent möchte ich bei einem Sicherheitsvorfall die Benutzer des Systems notifizieren können.	Es ist möglich eine Mail den Benutzern des Systems zu senden, in dem der Vorfall gemeldet wird.	Tief

Tabelle 2: User Stories

4.1.4 Mögliche Hacking Szenarien

Überblick

Im Rahmen dieser Arbeit werden generell diverse Hacking-Szenarien grob definiert, aus welchen zu einem späteren Verlauf Aufgabenstellungen konzipiert werden können. Diese werden hier festgehalten, jedoch innerhalb der Arbeit so nicht umgesetzt und sollen mehr als mögliche Inspiration zur Übungskreierung beitragen.

4.1.4.1 Cyber-Defense-Szenarien

Überblick

Hier werden fünf Attacken beschrieben die in einem Cyber-Defense-Szenario als Übung eingebaut werden könnten. Zur Inspiration wurde vor allem auf die Dokumentation von Wazuh zurückgegriffen. [47] [48]

SSH Brute Force

In diesem Szenario werden fehlerhafte Logins untersucht.

1. Die Lab-Umgebung aufstarten
2. Login Versuche unter einem nicht registrierten Namen auf einen Host, der über das SOC verbunden ist
3. Die erzeugten Logs nach dem entsprechenden Namen durchsuchen
4. Dashboard aufmachen und die Logs, welche den Namen enthalten durchsuchen
5. Die entsprechende Regel suchen in den Konfigurationen
6. Die entsprechende Regel ausschalten und testen, ob die Login Events immer noch rapportiert werden

Fazit: Die Sichtbarkeit von Logs ist durch Regeln beeinflussbar.

Rootkit Detektion

In diesem Szenario werden mögliche installierte Rootkits gesucht.

1. Die Lab-Umgebung aufstarten
2. Auf einen Linux Agent einloggen als Admin
3. Das Rootkit installieren
4. Checken, ob das Rootkit funktioniert
5. Rootcheck-Scan konfigurieren
6. Im Dashboard nachschauen, ob es aufgeführt wird als Rootkit

Fazit: Rootkits können auf Netzwerkhosts per Logfileeintrag festgestellt werden.

Filesystem-manipulation

In diesem Szenario werden alle möglichen Manipulationen auf dem Filesystem dargestellt und entsprechend kritische Strings geloggt.

1. Die Lab-Umgebung aufstarten
2. Auf einen Windows Agent einloggen als Admin
3. Filesystemlogging konfigurieren, damit Verzeichnisse überwacht werden
4. Dateien hinzufügen, löschen mit bösartigem Inhalt füllen
5. Die Log-Events anschauen
6. Das Dashboard betrachten

Fazit: Filesysteme und ihre Verzeichnisse können zentral überwacht werden

DDOS

In diesem Szenario wird ein DDOS-Angriff simuliert.

1. Die Lab-Umgebung aufstarten
2. Den Agenten konfigurieren, so dass mindestens 50 Events pro Sekunde geloggt werden können
3. Den Alert-Level in der Konfiguration senken
4. Mit dem Netcat-Script "makeflood" eine Event-Flut erzeugen
5. Das Dashboard betrachten
6. Den Alert-Level wieder erhöhen

Fazit: Der Alert-Level entscheidet darüber ob Events im Dashboard erscheinen oder nicht.

Port Scan

In diesem Szenario werden abnormale Netzwerkaktivitäten aufgezeigt und analysiert.

1. Die Lab-Umgebung aufstarten diesmal mit einem NIDS zusätzlich
2. Das NIDS triggern mit einem Script, welches Requests sendet
3. Auf dem Dashboard die Events auslesen
4. Den Kommunikationsvorgang nachvollziehen zwischen NIDS und SOC

Fazit: Portscan und andere abnormale Aktivitäten können mit dem SOC dargestellt und analysiert werden.

4.1.4.2 Incident-Response-Szenarien

Überblick

Hier werden fünf Attacken beschrieben die in einem Incident-Response-Szenario als Übung eingebaut werden könnten. Zur Inspiration dieser Szenarien wurde auf diverse Quellen zurückgegriffen und mit der Dokumentation von Wazuh abgeglichen. [49] [50] [51]

Dateienbeschädigung

In diesem Szenario wird dank Logdaten ein bestimmter Bereich im System als beschädigt/infiziert geflaggt. Es soll ein Playbook erstellt werden, die diese Dateien in ein Sandbox-Environment kopiert und auf dem Originalsystem löscht.

1. Die Lab-Umgebung aufstarten
2. Bekommt Datenschaden-Alert-Meldung, diese untersuchen
3. Playbook erstellen für copy/paste der beschädigten Dateien und Löschfunktion der originalen Dateien
4. Playbook soll bei erneutem Alarm dieser Kategorie gewünschten Effekt erzielen

Fazit: Erfahrung sammeln, wie man ein Playbook kreiert, dabei wird der Einfachheit halber ein Template zur Verfügung gestellt.

Firewall Breach

In diesem Szenario wird insbesondere die Firewall untersucht, die einen Incident besser identifizieren hätte können, wenn mehr Daten geloggt worden wären.

1. Die Lab-Umgebung aufstarten
2. Bekommt Firewall-Breached-Alert-Meldung, diese untersuchen
3. Student soll untersuchen was genau betroffen ist
4. Erkenntnis zu wenig Daten vorhanden
5. Firewall updaten, damit mehr Daten geloggt werden

Fazit: Erkenntnis, dass auch für Incident Response und weitere Forensics spezifische Logdaten sehr wichtig sind.

Preserve Breach

In diesem Szenario geht es darum jegliche betroffenen Daten eines Breaches zu konservieren für die weitere Analyse.

1. Die Lab-Umgebung aufstarten
2. Bekommt Alert-Meldung, dass Dateien beschädigt wurden
3. Playbook existiert, welches alle betroffene Dateien in ein Sandbox Environment konserviert
4. Logdaten fehlen jedoch bei Playbookdefinition und weitere Erkenntnisse für das Incident sind unmöglich
5. Playbook modifizieren, dass auch Logdaten mitkopiert werden

Fazit: In erster Linie geht es darum ein bereits bestehendes Playbook zu modifizieren und dann wieder die Erkenntnis, dass ohne (gute) Logs Forensics unmöglich werden.

Systemisolation

In diesem Szenario soll ein sich ausbreitender Breach simuliert werden, der so schnell wie möglich isoliert werden muss (Im Sinne WinAttackLab).

1. Die Lab-Umgebung aufstarten
2. Ongoing-Breach-Alert
3. Manuell ist man zu langsam den Breach einzudämmen
4. Automatisierten Isolationsplan erstellen/vervollständigen

Fazit: Es soll erkannt werden, dass manuell einen laufenden Breach einzudämmen nicht möglich ist und man im Vorfeld einen Systemisolationsplan erstellen sollte, zum Beispiel über Ansible.

4.1.5 Übungsszenarien

Überblick

Innerhalb dieser Arbeit werden diverse Übungsszenarien erstellt. Mit Übungsszenario wird insbesondere der Startpunkt, Ablauf und hoffentlich Erlerntes einer für die Studenten vorgegebene Übung beschrieben. Die diversen Szenarien werden auf Englisch festgehalten, an Ivan Bütler übergeben und von ihm auf dem Hacking-Lab verfügbar gemacht. Generell wird vorausgesetzt, dass die Infrastruktur durch den Studenten über das Hacking-Lab zuvor erstellt wurde, mit Ausnahme des Übungsszenarios 0, da es sich dabei um das Tutorial für genau diesen Vorgang handelt.

Folgend werden jedoch die erarbeiteten Übungsszenarien nur grob beschrieben und Erklärungen gegeben, warum gewisse Entscheidungen getroffen wurden. Die ausführliche Version ist im Anhang zu finden.

Übungsszenario 0: SOC Wazuh

Das Übungsszenario 0 dient als Einführung in Aufbau und Deployment der Labumgebung, weswegen es auch Einführungsszenario bezeichnet wird. Das Einführungsszenario besteht aus zwei Teilen. Im ersten Teil geht es um das

Deployment der Infrastruktur und im zweiten Teil um erste Interaktionen mit dem Security Operation Center.

Das Deployment wird hier durch den Studenten selbst über das Hacking-Lab ausgelöst. Die Dauer des Deployments dauert zwischen 50 und 60 Minuten. Der Rest des Übungsszenarios ist ausschliesslich dafür gedacht, dass die Studenten erste Einblicke in das SIEM-fokussierte Security Operation Center Wazuh erhalten. Der Fokus wird hier auf die Security Events im Dashboard gelegt und wie gefiltert werden kann. Ausserdem soll der Zusammenhang von einem Wazuh-Manager und einem Wazuh-Agent verstanden werden.

Neben der Einarbeitung kann der Student über den Attack-Launcher wiederholt eine SSH-Brute-Force Attacke auslösen. Diese Attacke wird registriert und soll schlussendlich im Wazuh-Dashboard ersichtlich sein. Die Entscheidung bei Bedarf den Angriff wiederholt auszuführen, wurde gefasst, um die unterschiedliche «Rule» bzw. Regel-Kategorisierung von Wazuh zu vertiefen.

Übungsszenario 1: Wazuh Agents über AD und GPO

Dieses Szenario wird benutzt, um den Studenten zu zeigen, wie ein Wazuh-Agent in einem Active Directory verteilt werden kann. Hier wird speziell darauf geachtet, dass gezeigt wird, wie neue Items in der Group Policy und dem Active Directory erstellt werden können. Es soll dem Studenten einerseits zeigen, dass ein Wazuh-Agent relativ einfach in einer Domain verteilt werden kann, andererseits soll die Aufgabe das Wissen über der Interaktion mit dem Active Directory und Group Policy erhöhen.

Übungsszenario 2: Sysmon, mimikatz und Log-Konfiguration mit alert

In diesem Szenario wird der Umgang mit Sysmon und das Darstellen von durch Sysmon registrierten Events in Wazuh untersucht. Die Generierung von bösartiger Aktivität wird durch das Tool mimikatz ausgelöst. Diese Aktivität soll schlussendlich von Wazuh registriert werden, weswegen auch der Umgang mit Wazuh-Rules in dieser Aufgabe einbezogen wird. Ausserdem soll ein Log-Forwarder innerhalb dieser Aufgabe erstellt werden, der Sysmon-Events zu Wazuh von einem System ohne Wazuh-Agent sendet. Am Schluss soll ein Alert in Wazuh erstellt werden, der bei einem mimikatz Anwendungsvorfall eine E-Mail sendet.

Übungsszenario 3: CDB und bösartige IP-Aufrufe

In diesem Szenario geht es darum eine CDB (Constant Database) Liste in Wazuh zu erstellen, welche mit bösartigen gelisteten IP-Adressen gefüllt ist. Diese Liste wird frisch in der Übung von der Seite URLHaus bezogen, welche sich auf das Sammeln von bösartigen Adressen spezialisiert hat. Der Student soll hier zum ersten Mal mit dem Wazuh-Manager direkt auf dem Ubuntu-Server interagieren und die nötigen Konfigurationen dort erledigen. Da in diesem Szenario schlussendlich Aufrufe auf echte bösartige IP-Adressen gemacht werden, werden auch die nötigen Konfigurationen für das bereits installierte NIDS Suricata erledigt.

Die CDB-Liste wird im Endeffekt als Vergleichsmittel eingesetzt, welche jeweils von Wazuh bei einem Aufruf auf eine externe Adresse bezogen und verglichen wird. Hierfür wurde beim Attack-Launcher Service die Funktionalität hinzugefügt, dass wiederholt 10 aus diesen bösartigen IP-Adressen dynamisch ausgewählt und aufgerufen werden können.

Ziel der Aufgabe ist schlussendlich einen neuen Alert in Wazuh zu erstellen, welche diese bösartige Aktivität registriert und den neuen Alert sendet.

Übunggszenario 4: Dashboard Kreation

Dieses Szenario dient ausschliesslich dazu, einen Guide zu geben, wie man in Wazuh mit Kibana ein neues Dashboard erstellt und diesem neue Visualisierungen hinzufügen kann. Es dient sozusagen zur visuellen und praktischen Vertiefung mit dem Dashboard und soll dem Studenten zeigen, wie alternative Darstellungen die Daten bzw. die Alerts klarer darstellen.

4.1.6 Nichtfunktionale Anforderungen

Überblick

Die nichtfunktionalen Anforderungen wurden auf Basis der in ISO/IEC 9126 [52] definierten Kriterien erfasst.

Terraform Deployment

Zuverlässigkeit:

- Fehlertoleranz
  Die Infrastruktur soll stabil laufen, bzw. ein Deployment soll eine Instanz vom SOC und Umgebung mit hoher Verfügbarkeit machen.

Übertragbarkeit:

- Installierbarkeit
  Die gesamte Infrastruktur wird über Terraform deployed und die SOC-Ressourcen werden unter der Verwendung von Docker einbezogen. Terraform selbst ist auf einer Linux-VM installiert. Die «Hacking-Lab LiveCD» von Compass Security, der Firma von Ivan Bütler, wird hier verwendet. Grund für diese Wahl ist, dass Auftraggeber und -nehmer dieser Arbeit bereits Erfahrung mit dieser VM haben und sie eine gute Übertragbarkeit leistet.

Funktionalität:

- Sicherheit
  Das Zugreifen auf die Ressourcen wird durch SSH-Key-Exchange oder Username & Passwort geschützt.
  Die Azure Cloud Zugangsdaten, welche für das Terraform Deploymen benötigt werden, werden nur über einen sicheren Channel per FileBox [53] übertragen.

4.1.7 Weitere Anforderung

Zugang Infrastruktur über Hacking-Lab

Die Infrastruktur soll über das Hacking-Lab von Ivan Bütler deploy-/destroybar sein. Grund hierfür ist, dass die Übungen im geplanten Modul im Hacking-Lab stattfinden werden. Die Einbindung der Ressourcen und das Übertragen der Übungsszenarien wird jedoch von Ivan Bütler übernommen. Damit das Übertragen der Übungsszenarien möglichst einfach ist, werden diese in einem Markdown-File pro Szenario auf dem BA-EduSOC Repository gehalten. Es wird ausserdem darauf

geachtet, dass der Text nicht zu verschachtelt und auf Englisch geschrieben ist, um die Übertragbarkeit aufs Hacking-Lab weiter zu verbessern.

4.2 Microsoft Azure Cloud Ressourcen & Kosten

Überblick

In diesem Kapitel werden die Ressourcen und Kosten, welche durch Terraform in der Microsoft Azure Cloud erstellt und verursacht werden, ausführlich beschrieben und analysiert.

4.2.1 Azure Ressourcen

Überblick

Die Azure Cloud stellt unter anderem über Terraform ihre Ressourcen zur Verfügung. Diese Ressourcen werden in einer spezifischen Terraform-Syntax schlussendlich erstellt und über Azure bezogen. Folgend wird beschrieben wie darauf zugegriffen wurde.

Azure Provider

Der Azure Provider wird benutzt, um Infrastruktur-Konfigurationen in Microsoft Azure über das «Azure Resource Manager API» durchzuführen. Die spezifischen Ressourcen, die unterstützt werden, können über das az-CLI abgefragt werden. Generelle Dokumentation über die Syntax und Gebrauch der Ressource ist unter der Terraform azurerm Dokumentation zu finden. [54]

az-CLI

Mit Microsoft Azure wurde innerhalb dieser Arbeit nur über das Command Line Interface Tool az interagiert. Darüber kann man sich bei Microsoft Azure einloggen und innerhalb der Arbeit wurden die zur Verfügung stehenden virtuellen Maschinen abgefragt, die gute Kandidaten für den SOC-Host sind. Da die gewählte SOC-Plattform einen Linux Host braucht, wurde generell nach Ubuntu-Maschinen gesucht. Grund für Ubuntu war die bestehende Erfahrung mit diesem Linux-OS-System.

Das Login kann wie folgt, unter der Benutzung der Microsoft Azure Zugangsdaten, erledigt werden:

az login --service-principal --username someuser --password somepassword --tenant tenantid

Um nach Ubuntu-Maschinen zu suchen, wurde folgender Befehl benutzt:

az vm image list --offer UbuntuServer --publisher Canonical --all --output table

Um sicher zu stellen, dass beim Herunterfahren eines Deployments die VM-Ressourcen wirklich heruntergefahren beziehungsweise zerstört wurden, wurde jeweils der folgende Befehl benutzt:

az vm list -d -o table

Generell kann az-CLI natürlich um einiges mehr, es werden hier lediglich die Befehle erwähnt, mit denen interagiert wurde. Bei Interesse wird auf die offizielle Azure Command-Line-Interface-Dokumentation verwiesen. [55]

4.2.2 Azure-Ressourcen Kostenanalyse

Überblick

Die diversen Azure-Ressourcen, die über Terraform erstellt werden, kosten alle jeweils einen spezifischen Betrag. In dieser Sektion werden diese bezüglich den VMs festgehalten und erklärt, warum an gewissen Orten trotz erhöhter Kosten Entscheidungen zu einem Upgrade getroffen wurden.

In der folgenden Tabelle wird dementsprechend auf die jeweils ausgewählten virtuellen Maschinen und die gewählte Ressourcen-Stärke, welche unter *size* bei Terraform in einer Virtuellen-Maschinen-Ressource gesetzt ist, eingegangen. Die definierten Abkürzungen und Kosten wiederrum findet man auf Azure unter «Pricing Calculator». [56]

Ressource	Size	Instanz	Kosten in $/Stunde
Ubuntu1-Server	Standard_DS3_v2	4 vCPUs, 14 GB RAM, 28 GB Temp storage	0.272
Kali1-Client	Standard_DS1_v2	1 vCPU, 3.5 GB RAM, 7 GB Temp storage	0.068
Windows-Client	**Standard_B2ms**	2 vCPUs, 8 GB RAM, 16 GB Temp storage	0.104
DC1-Server	Standard_B2s	2 vCPUs, 4 GB RAM, 8 GB Temp storage	0.056
FS1-Server	Standard_B1ms	1 vCPU, 2 GB RAM, 4 GB Temp storage	0.028
FS2-Server	Standard_B1ms	1 vCPU, 2 GB RAM, 4 GB Temp storage	0.028
WS1- Server	Standard_B1ms	1 vCPU, 2 GB RAM, 4 GB Temp storage	0.028
Management -Client	Standard_B2s	2 vCPUs, 4 GB RAM, 8 GB Temp storage	0.056
Total			**0.612**

Tabelle 3: VM-Ressourcen-Übersicht

Begründung

Wie ersichtlich ist, entstehen die grössten Kosten bezüglich der virtuellen Maschinen beim Ubuntu-Server und beim Windows-Client.

Ubuntu-Server:

Die Entscheidung, der virtuellen Ubuntu-Maschine mehr Ressourcen zur Verfügung zu stellen, hat diverse Gründe. Der wohl wichtigste Grund ist, dass das SOC auf dieser Maschine gehostet wird und zusammen mit Elastic Stack bereits eine grössere Menge an Ressourcen beansprucht. Neben dem SOC werden auch andere Services gehostet, wie zum Beispiel der Attack-Launcher. Da diese VM mit dem Security Operation Center Wazuh im Endeffekt Kern dieser Arbeit ist, soll auch garantiert werden, dass diese

Maschine eine schnelle Reaktionszeit bietet und nicht durch Ressourcenmangel eingeschränkt wird.

Windows-Client:

Die Entscheidung beim Windows-Client von der Standard Ressource B2s auf B2ms umzusteigen, wurde wegen der erhöhten Interaktion der Studenten mit dem Windows 10 Client gefasst. Das Interagieren war vor diesem Upgrade um einiges langsamer und es war nicht angenehm, damit zu arbeiten. Dies würde insbesondere einen reibungslosen Übungsablauf gefährden.

4.3 Design

Überblick

In diesem Kapitel wird auf die Netzwerk-Architektur und auf die benutzten Technologien eingegangen, die für dieses Projekt verwendet wurden. Es werden jedoch aus Platzgründen nicht alle technischen Facetten der verwendeten Technologien im Rahmen dieser Arbeit beschrieben, sondern die Art und Weise wie die verschiedenen Technologien in dieser Arbeit eingebunden wurden.

4.3.1 Netzwerk-Architektur

WinattackLab PoC

In der Elaboration-Phase wurde ein erster Prototyp entwickelt, der darin besteht, eine zusätzliche Ubuntu-VM mit Docker und diversen Containern in der Azure Cloud in eine vom Betreuer zur Verfügung gestellten WinattackLab-Architektur zu erstellen. In der Construction-Phase wurde dieser Prototyp dann weiter ausgebaut und die Container spezifiziert.

In der folgenden Abbildung ist ersichtlich, wie der Stand am Ende der Elaboration-Phase aussah. Der Umgebung wurde eine Ubuntu 18.04 virtuelle Maschine hinzugefügt, auf welcher Docker bereits mit der Plattform Wazuh, hier mit SOC System Container abstrahiert, läuft.

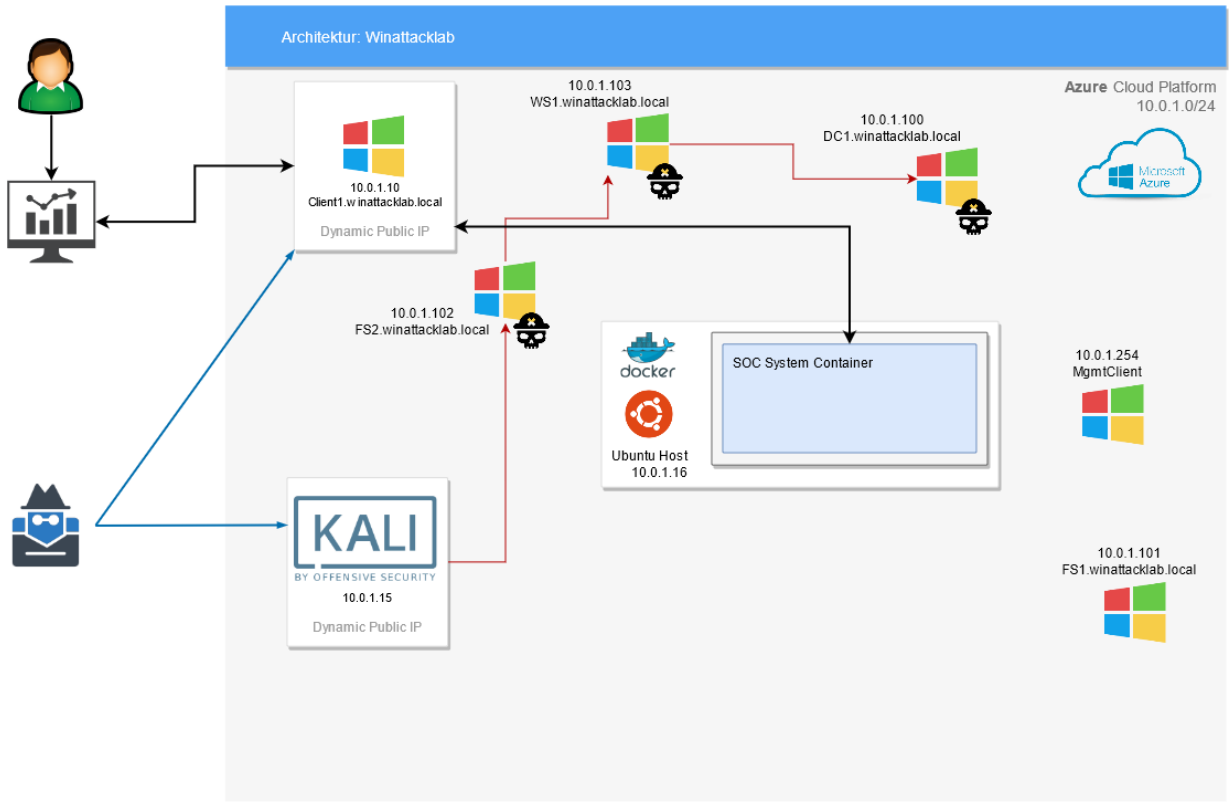


Abbildung 5: Proof of Concept Netzwerk-Architektur

Wazuh-Kommunikations-fluss

Der Kommunikationsfluss von Wazuh funktioniert in der Regel so, dass jede Maschine, welche einen Wazuh-Agent installiert und konfiguriert hat, mit dem Wazuh-Manager kommuniziert. In der Abbildung 4 liegt der Wazuh-Manager auf einem Docker-Container auf dem Ubuntu-Host und ein Wazuh-Agent auf dem Windows Client1.

Es ist nun aber nicht nötig auf jeder Maschine einen Wazuh-Agent zu installieren. Es braucht lediglich ein System mit einem Wazuh-Agent, auf den Events anderer Systeme weitergeleitet werden können. Diese Weiterleitung wird auch innerhalb eines Übungsszenarios weiter analysiert.

WinattackLab Finale Version

Vom Proof of Concept zur finalen Version sind folgende weitere Komponenten einbezogen worden: Auf dem Ubuntu-Host wurde über Docker noch ein traefik-Service erstellt und es wurde die Entscheidung getroffen einen Attack-Launcher-Service hier auch über Docker zu hosten, anstatt, wie ursprünglich gedacht, diesen auf der Kali-Linux-Maschine zu halten. Grund für diese Entscheidung ist die damit vereinfachte Haltung der Infrastruktur und dass der Student schlussendlich direkt über den Browser, ähnlich dem Wazuh-Dashboard, auf den Attack-Launcher zugreifen kann. Neu ist auch, dass der Zugriff auf die Dienste über den Browser nicht mehr direkt über den IP-Adressen-Aufruf stattfindet, sondern über die durch traefik gesetzten Domain-Namen.

Für das HTTPS-Schema wird ein selbst-signiertes Zertifikat verwendet, welches beim Deployment der Infrastruktur auf dem Ubuntu Server erstellt wird und in den Trusted Root CA bei Windows importiert wird.

Die durch traefik weitergeleiteten FQDN sind:

- https://wazuh.winattacklab.local
- https://attack-launcher.winattacklab.local
- https://mailcatcher.winattacklab.local
- http://traefik.winattacklab.local:8080

Finale Netzwerk-Architektur

Bezüglich der finalen Netzwerk-Architektur wurde zur Übersicht folgendes Diagramm erstellt.

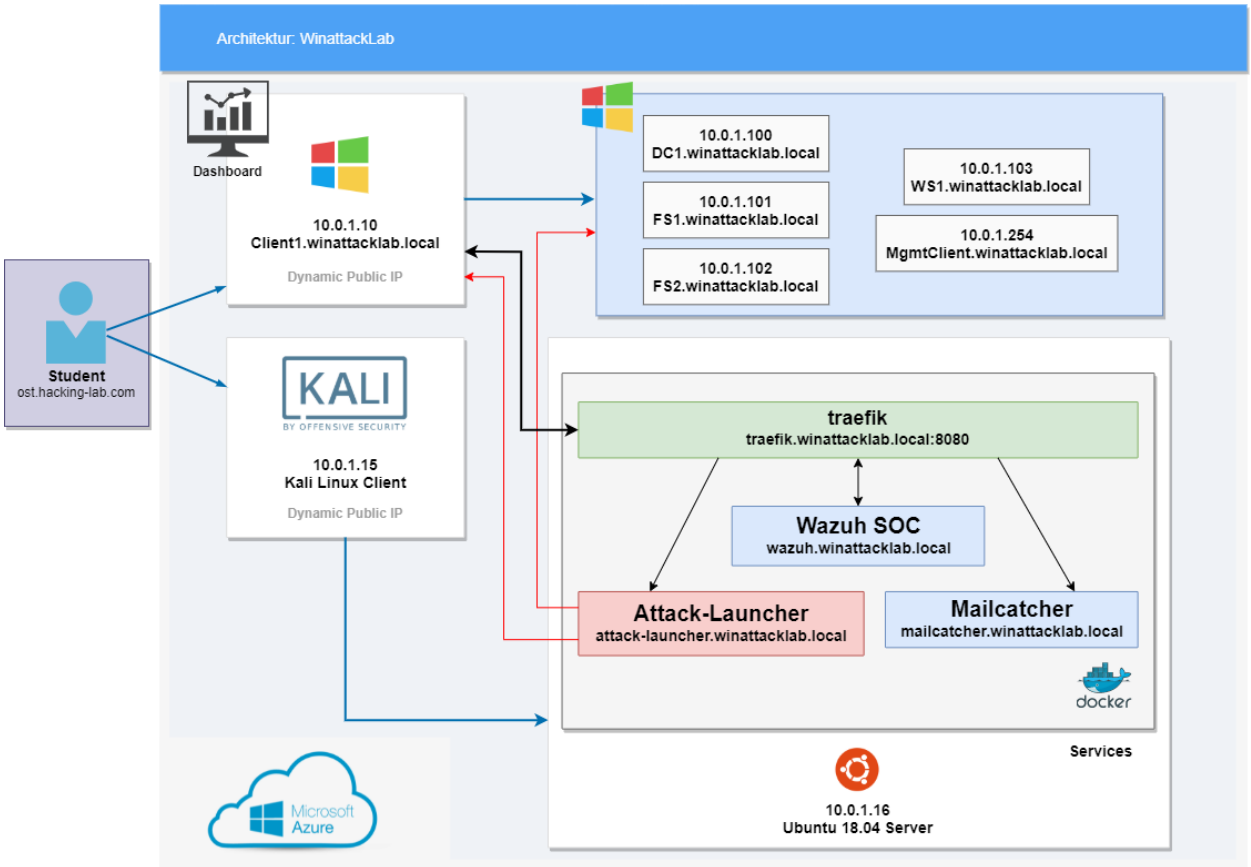


Abbildung 6: Finale Netzwerk-Architektur Übersicht

4.4 Terraform-Ressourcen & Komponenten-Implementation

Überblick

Eine Terraform-Ressource besteht aus einzelnen Komponenten und bildet zusammen mit allen für die VM relevanten Ressourcen zusammen wieder eine Ressource. Damit die Unterscheidung der Ressourcenbenamsung klar ist, wird für die gesammelten Ressourcen und Komponenten im Terraformcode der Begriff Modul bzw. module anstatt Ressource verwendet. In dieser Dokumentation wird jedoch weiterhin der Begriff Ressource verwendet.

Innerhalb dieser Arbeit wurde insbesondere der Infrastruktur WinattackLab eine neue VM-Ressource hinzugefügt. Diese wird in diesem Kapitel ausführlich beschrieben. Ausnahmen gelten bei den erhaltenen Ressourcen, welche mit wenigen Anpassungen für die Erweiterung mit den neuen Ressourcen und den SOC Funktionalitäten erweitert wurden. Diese Ressourcen werden nur am Rande erwähnt und deren Veränderungen kurz beschrieben, da sich diese nicht gross verändert haben und dieser Terraform-Komponentencode generell nicht Open-Source ist.

4.4.1 Terraform Implementation

4.4.1.1 Ubuntu1-Server

Überblick

Die komplett neue Ressource im WinattackLab ist ein Ubuntu 18.04 Image. Auf dieser Ressource wird das Security Operation Center Wazuh inklusive Elastic Stack auf Docker gehostet. Dies gilt auch für die Docker-Services Attack-Launcher, MailCatcher und traefik. Die Docker-Services werden explizit im Handbuch beschrieben.

Der Ubuntu1-Server wird in der Terraform-Implementation in sechs Terraform *.tf* Dateien unterteilt und einem Bash *setup.sh* Skript, welches bei der Extension-Terraform-Datei in den Einsatz kommt. Das Format dieser Terraform-Code-Struktur-Unterteilung wurde generell dem bestehenden Schema, welches in den anderen Komponenten der übergebenen Infrastruktur so aufgefunden wurde, nachempfunden:

0-init.tf

In dieser Datei wird nur eine lokale Variable für die Ubuntu-Maschine erstellt.

```
locals {
  virtual_machine_name = "ubuntu1"
}
```

1-network-interface.tf

In dieser Datei wird das Netzwerk-Interface konfiguriert. Wichtige Konfiguration hier ist die Inbound-Kommunikation-Regel, welche auf *SSH erlaubt* gesetzt ist und dass der DNS-Server auf den Domain Controller der Infrastruktur gesetzt ist. Ausserdem wird hier die sonst in der Infrastruktur verwendete Netzwerk-Interface-Sicherheits-Gruppen-Assoziation eingesetzt, welche die IP-Konfiguration der Ressource durchführt.

```
# Create Network Security Group and rule
resource "azurerm_network_security_group" "ubuntu1nsg" {
    name                = "${var.lab_id_ubuntu1}_ubuntu1_NetworkSecurityGroup"
    location            = var.location
    resource_group_name = var.resource_group_name

    security_rule {
        name                       = "SSH"
        priority                   = 1001
        direction                  = "Inbound"
        access                     = "Allow"
        protocol                   = "Tcp"
        source_port_range          = "*"
        destination_port_range     = "22"
        source_address_prefix      = "*"
        destination_address_prefix = "*"
    }

    depends_on = [var.main_depends_on]

    tags = {
        environment = var.environment_tag
    }
}


# Create network interface
resource "azurerm_network_interface" "ubuntu1nic" {
    name                      = "${var.lab_id_ubuntu1}_ubuntu1_NIC"
    location                  = var.location
    resource_group_name       = var.resource_group_name

    dns_servers               = [var.dc1_server_ip]

    ip_configuration {
        name                          = "${var.lab_id_ubuntu1}_ubuntu1_NICConfiguration"
        subnet_id                     = var.subnet1_id
        private_ip_address_allocation = "Static"
        private_ip_address            = var.ubuntu1_server_ip
    }

    tags = {
        environment = var.environment_tag
    }
}


# Create association between nic and network security group
resource "azurerm_network_interface_security_group_association" "ubuntu1nicnsg" {
  network_interface_id      = azurerm_network_interface.ubuntu1nic.id
  network_security_group_id = azurerm_network_security_group.ubuntu1nsg.id
}
```

2-virtual-machine.tf In dieser Datei wird die Ubuntu-VM unter *source_image_reference* konfiguriert bzw. gesetzt. Hier ist auch der VM-Plan, welcher in der Tabelle von 4.2.2 vorkommt, definiert. Ausserdem wichtig ist die Konfiguration des SSH-Public-Key-Pfades für die zuvor definierten Inbound-Regeln. Die SSH-Keys werden dabei jeweils beim Deployment der Infrastruktur neu generiert.

```
resource "azurerm_linux_virtual_machine" "ubuntu1vm" {
  name                            = "${var.lab_id_ubuntu1}_Ubuntu1_vm"
  location                        = var.location
  resource_group_name             = var.resource_group_name
  network_interface_ids           = [azurerm_network_interface.ubuntu1nic.id]
  size                            = "Standard_DS3_v2"
  admin_username                  = var.admin_username
```

```
  computer_name                 = local.virtual_machine_name

  os_disk {
    name                 = "${var.lab_id_ubuntu1}_Ubuntu1_OSDisk"
    caching              = "ReadWrite"
    storage_account_type = "Premium_LRS"
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "18.04-LTS"
    version   = "latest"
  }

  disable_password_authentication = true

  admin_ssh_key {
    username   = var.admin_username
    public_key = file("${path.module}/kali_public_key")
  }

  tags = {
    environment = var.environment_tag
    type        = "ubuntu1"
  }
}
```

3-execute-extension.tf

Die Datei wird im Zusammenhang mit dem Skript *setup.sh* benutzt. Damit konfiguriert Terraform die Ubuntu-Ressource, damit sie dieses Skript nach der Ressourcenerstellung ausführt. Dabei wurde interessanterweise noch ein Bug von Terraform entdeckt, welcher es verhindert, dass die Skriptdatei in einem anderen Pfad als die Datei mit der Ressource *azurerm_virtual_machine_extension* liegt. Sehr wichtig ist in dieser Ressource das Setzen von *depends_on*. Diese Abhängigkeitsdefinition verhindert, dass das Skript zu früh ausgeführt wird, und wartet dementsprechend auf die Fertigstellung der Ubuntu-VM und der DC-VM-Ressourcen.

```
resource "azurerm_virtual_machine_extension" "ubuntu1vm-execute-custom" {
  name                 = azurerm_linux_virtual_machine.ubuntu1vm.name
  virtual_machine_id   = azurerm_linux_virtual_machine.ubuntu1vm.id
  publisher            = "Microsoft.Azure.Extensions"
  type                 = "CustomScript"
  type_handler_version = "2.0"

  # script needs to be in same directory, triggers current Terraform bug otherwise
  settings = <<SETTINGS
    {
        "script": "${base64encode(templatefile("${path.module}/setup.sh", {
          admin_username=var.admin_username,
          admin_password=var.admin_password,
          hostname=var.hostname,
          ubuntu1_server_ip=var.ubuntu1_server_ip,
          windows_client_ip=var.windows_client_ip
        }))}"
    }
SETTINGS


  depends_on = [azurerm_linux_virtual_machine.ubuntu1vm, var.dc_depends_on]
}
```

4-post-build-commands.tf

In dieser Datei wird eine sogenannte Provisioning-Zeit eingebaut, welche der VM nach deren Erstellung ein wenig Zeit gibt, sich zu etablieren.

```
resource "time_sleep" "ubuntu1-wait-provision" {
  depends_on = [azurerm_linux_virtual_machine.ubuntu1vm]

  create_duration = "2m"
}
```

variables.tf

Diese Datei wird benutzt, um globale Variablen zu definieren. Sie wird im *main.tf* File einbezogen, welches alle Ressourcen, welche deployed werden sollen, beinhaltet. Da diese Datei lediglich diversen Variablen den Typ setzt und den Namen definiert, wird sie hier nicht ausführlich aufgeführt. Stattdessen hier die letzten drei Einträge als Beispiel:

```
variable "ubuntu1_server_ip" {
  type = string
}

variable "dc1_server_ip" {
  type = string
}

variable "windows_client_ip" {
  type = string
}
```

setup.sh

Diese Datei enthält die Logik, welche zuständig für das gesamte SOC-Deployment und Konfiguration, benötigte Tools und weitere Docker-Services ist. Zu Beginn des Skripts werden generell Pfade kreiert und diverse benötigte Tools und GitHub-Projekte heruntergeladen und installiert.

Am Schluss des Skripts wird ein Loop ausgeführt, welche die Verbindung zum Windows 10 Client testet. Ist dieser erreichbar, wird das zuvor erstellte selbstsigniere Zertifikat, welches beim traefik-Service einbezogen wird, auf den Windows 10 Client gesendet.

Aus Platzgründen wurden die Inhalte der Datei nicht hier, sondern im Anhang aufgeführt.

Kali1-Client

Der Kali-Linux-Client wurde um die Ubuntu-Server-IP-Variable erweitert und dessen DNS-Konfiguration zu der IP des DCs geändert. Ausserdem wurde wegen dieser Veränderung die Abhängigkeit zum DC hinzugefügt.

Die Art und Weise wie das Skript *setup.sh* ausgeführt wird, wurde angepasst. Dieses Skript wird nun per Virtual-Machine-Extension-Ressource ausgeführt und nicht zuerst auf das erstellte System kopiert und ausgeführt. Diese Veränderung geht darauf zurück, dass ursprünglich die Attacken von dieser Maschine gestartet wurden und die einzelnen Skripts der Attacken auf das System hochgeladen wurden. Da die Entscheidung getroffen wurde, den Attack-Launcher-Service auf dem Ubuntu-Server zu kreieren, wurde diese Funktionalität nicht mehr gebraucht.

4.4.1.2 DC1-Server

Überblick

Der Windows Domain Controller (DC) der Infrastruktur. Dieser ist im Active Directory für die zentrale Authentifizierung der virtuellen Maschinen und deren Benutzern zuständig.

Die Erweiterung hier liegt im Zusammenhang mit dem SOC. Es wurde ein zusätzliches PowerShell-Skript-Template, mit dem Titel *soc_relevant_template.ps1*, hinzugefügt, welches nur dafür zuständig ist, dass ein neuer Wildcard-DNS-Eintrag hinzugefügt wird. Es wurde jedoch explizit ein neues Skript hinzugefügt, damit die neue Konfiguration und der Zusammenhang mit der Infrastruktur klar ersichtlich sind.

Der neue DNS-Eintrag für die SOC-Funktionalitäten konnte mit einem Einzeiler erreicht werden:

```
Add-DnsServerResourceRecordA -Name * -ZoneName winattacklab.local -IPv4Address ${ubuntu1_server_ip}
```

4.4.1.3 Windows-Client

Überblick

Dem Windows 10 Client wurde ein neues PowerShell-Template hinzugefügt, welches den Namen *connect_wazuh_agent_template.ps1* trägt. Dieses Skript wird während dem Terraform-Deployment ausgeführt. Dank der übergebenen und gesetzten statischen Ubuntu-Server-IP-Variable wird der Wazuh-Agent im Skript nach seiner Installation auch direkt mit dem Wazuh-Manager verknüpft. Die Installation des Wazuh-Agents findet innerhalb des PowerShell-Skripts *installTools.ps1* statt, welches auch neu den Chrome-Browser installiert. Der Chrome-Browser wird in den Übungen für die diversen Service-Ansichten benutzt.

Neben dem neuen Skript wurde nur eine neue Variable für die Ubuntu-Server-IP-Adresse hinzugefügt.

Hier der Inhalt von *connect_wazuh_agent_template.ps1*:

```
[string]$UbuntuClientIpString    = "${ubuntu1_server_ip}"

printInfo("++++++++++ New Script $PSCommandPath ++++++++++")

C:\Users\LAB_ADMIN\wazuh-agent-4.1.4-1.msi /q WAZUH_MANAGER="${ubuntu1_server_ip}" \
WAZUH_REGISTRATION_SERVER="${ubuntu1_server_ip}"

&'C:\Program Files (x86)\ossec-agent\agent-auth.exe' -m ${ubuntu1_server_ip} \
((Get-Content -path 'C:\Program Files (x86)\ossec-agent\ossec.conf' -Raw) -replace '0.0.0.0', \
$UbuntuClientIpString) | Set-Content -Path 'C:\Program Files (x86)\ossec-agent\ossec.conf'

Get-Content -path 'C:\Program Files (x86)\ossec-agent\ossec.conf'

Restart-Service -Name wazuh
```

4.4.1.4 Kali1-Client

Überblick

Der Kali-Linux-Client wurde um die Ubuntu-Server-IP-Variable erweitert und dessen DNS-Konfiguration zu der IP des DCs geändert. Ausserdem wurde wegen dieser Veränderung die Abhängigkeit zum DC hinzugefügt.

Die Art und Weise wie das Skript *setup.sh* ausgeführt wird, wurde angepasst. Dieses Skript wird nun per Virtual-Machine-Extension-Ressource ausgeführt und nicht zuerst auf das erstellte System kopiert und ausgeführt. Diese Veränderung geht darauf zurück, dass ursprünglich die Attacken von dieser Maschine gestartet wurden und die einzelnen Skripts der Attacken auf das System hochgeladen wurden. Da die Entscheidung getroffen wurde, den Attack-Launcher-Service auf dem Ubuntu-Server zu kreieren, wurde diese Funktionalität nicht mehr gebraucht.

4.4.1.5 FS1-, FS2- und WS1-Server und Management-Client

Überblick

Am Code der FS-Server, dem WS1-Server und dem Management-Client wurde nichts verändert.

4.4.2 Manuelle Testverfahren

Überblick

Das Haupttestverfahren dieser Arbeit bestand darin das Deployment der Infrastruktur manuell durchzuführen und das korrekte Aufsetzen der Infrastruktur zu begutachten.

Deployment

Ein Deployment der gesamten Infrastruktur dauert zwischen 50 und 60 Minuten. Dementsprechend ist es relativ zeitaufwendig zu testen, ob gewisse Einstellungen oder Änderungen an den Komponenten etwas an der Logik der Infrastruktur zerstört haben.

Hier muss zwischen Syntax Fehlern der «Infrastructure as Code» Terraform-Umgebung, Unstimmigkeiten beim Timing der Erstellung der Ressourcen und Fehlern in den nachträglichen Veränderungen innerhalb einer Ressource, erledigt über sogenannte Terraform «extensions», unterschieden werden.

Syntax Fehler werden innerhalb wenigen Sekunden bei einem Deployment ersichtlich und verhindern die Durchführung des Deployments. Somit sind diese relativ leicht zu entdecken und zu reparieren.

Zeitlich problematischer sind die anderen zwei Kandidaten: Timing und Extensions. Um diese Fehler zu erkennen, muss das gesamte Deployment ausgeführt werden, damit ersichtlich wird, ob die Konfigurationen funktionstauglich sind. Diese 50-60 Minuten wurden jedoch nicht jeweils verschwendet, da in dieser Zeit jeweils an einem anderen Task gearbeitet wurde, der keine Interaktion mit der Infrastruktur benötigte.

Deployment-Tests wurden generell oft gemacht, da das stabile Aufsetzen der Infrastruktur und des Labs ein Hauptfokuspunkt dieser Arbeit ist. Es wurden auch viele Tests hier durchgeführt, da Abhängigkeitsprobleme zwischen den Ressourcen Windows 10 Client, Ubuntu-Server und dem Domain Controller aufgetaucht sind und das Deployment nicht durchgehend durch die Arbeit hinweg stabil lief. Anhand dieser Tests konnte schlussendlich ein stabiles Deployment konfiguriert werden.

Übungsszenarien Die Übungsszenarien werden in einem iterativen Prozess ausgearbeitet und wiederholt ausgeführt. Gegen den Schluss der Ausarbeitung wird das entstandene dokumentierte Übungsszenario nochmals anhand der Aufgabenstellung ausgeführt und von Ivan Bütler auf das Hacking-Lab übertragen.

4.5 Resultate und Weiterentwicklung

Überblick In diesem Kapitel werden die Resultate der Arbeit festgehalten und auf eine mögliche Weiterentwicklung eingegangen.

4.5.1 Resultate

SOC-Wazuh Das Hauptziel dieser Arbeit, ein Security Operation Center über Terraform in die WinattackLab-Architektur einzubinden, wurde erreicht. Die Entscheidung, eine SIEM-fokussierte Plattform zu wählen und eine SOAR-Plattform höchstens in einer theoretischen Erweiterung dieser Arbeit einzubeziehen, wurde bei der Gruppentrennung gefasst. Dies sollte den Erfolg der Arbeit erhöhen, da der Aufwand, sowohl eine SIEM- als auch eine SOAR-Plattform in die Infrastruktur einzubeziehen, einen zu grossen Arbeitsumfang für eine einzelne Person bedeutet hätte. Die Entscheidung, warum eine SIEM- und nicht eine SOAR-Plattform gewählt wurde, lässt sich dadurch begründen, dass der SOAR-Bereich eines SOCs im Normalfall von einem SIEM-Bereich abhängig ist. Ergo ist es sinnig, dass mit dem SIEM-Bereich begonnen wurde.

Attack-Launcher Das Ziel, den Benutzern die Möglichkeit zu geben, Attacken auf die Infrastruktur auszulösen, wurde ebenfalls erreicht. Die Benutzer können über den Attack-Launcher-Service diverse Angriffe wiederholt auslösen:

- SSH-Brute-Force-Attacke
- RDP-Brute-Force-Attacke
- Aufruf von bösartigen IP-Adressen

Übungsszenarien Da die Ziele «Security Operation Center in übergebene Infrastruktur einbinden» und «Attack-Launcher-Service kreieren» erreicht wurden, konnten Übungsszenarien erarbeitet werden. Schlussendlich wurden folgende Übungsszenarienziele erreicht:

- «Einarbeitungsszenario», welches dazu dient, dass die Studenten den ersten Umgang mit dem SOC-Wazuh erlernen
- «Interaktion mit AD und GPO im Zusammenhang mit Wazuh-Agents Szenario», welches dazu dient den Studenten den Umgang mit Active Directory und Group Policy näher zu bringen und zu zeigen, wie man einen Wazuh-Agent in einem Active Directory verteilen kann.

- «Sysmon, mimikatz und Wazuh rule Logs Szenario», welches den Studenten vor allem zeigen soll, wie man Sysmon und Wazuh konfigurieren kann, um gewisse Event zu registrieren.
- «Log Forwarder und Alerts Szenario», welches zeigen soll wie man einen Log Forwarder erstellt und wie man in Wazuh Alerts erstellt.
- «Dashboard Szenario», welches eine weitere Übung bietet, um den Studenten beizubringen, eine Übersicht der Ereignisse zu gewinnen.

4.5.2 Möglichkeiten der Weiterentwicklung

Wazuh Übungsszenario

Weitere Übungsszenarien können entwickelt werden, zum Beispiel eines mit dem Fokus das FIM-Modul zu verwenden oder eine Konfigurationsaufgabe, welche mit der MITRE ATT&CK [57] Dashboard Ansicht in Wazuh zu tun hat.

SOAR-Plattform

Ein Security Operation Center besteht normalerweise aus einem SIEM- und einem SOAR-Bereich. Da die Zeit innerhalb dieser Arbeit nur für den SIEM-Bereich reichte, bietet sich natürlich die Option an, die Infrastruktur um eine SOAR-Plattform zu erweitern.

Die Wazuh-Plattform enthält ein paar SOAR-Use-Cases, wie zum Beispiel Active Response, jedoch ist sehr gut möglich, dass noch mehr Funktionalitäten wie zum Beispiel ein Ticket-System erwünscht wären. Dementsprechend würde sich das Aufsetzen einer SOAR-fokussierten Plattform lohnen. Die Plattform TheHive, welche in der Elaborationsphase untersucht wurde, eignet sich hierfür als mögliche Kandidatin.

Aus Zeitgründen wird an dieser Stelle auf eine genaue Ausführung verzichtet. Die Machbarkeit einer Zusammenarbeit von TheHive und Wazuh ist jedoch unbestritten, da dies bereits in anderen Projekten so gelöst wurde. [58] [59]

Attack-Launcher

Der Attack-Launcher-Service kann für das Lab um weitere Angriffsszenarien erweitert werden. Ausserdem können die Übungsszenarien bzw. deren Attacken noch um den MITRE ATT&CK Kontext bereichert werden.

4.6 Projektmanagement – Soll

Überblick

Die nachfolgenden Unterkapitel von «Projektmanagement – Soll» dienen zur administrativen Übersicht des Projektes. Diese Kapitel entstanden während der Elaborationsphase.

4.6.1 Organisation des Projekts

Prozessmodell

Das Prozessmodell dieser Arbeit bezieht sich grob auf das in Daniel Kellers Vorlesung «Software Engineering 1» beschriebene Scrum+-Framework. Scrum+ ist eine Kombination aus agilem Scrum und RUP (Rational Unified Process). Letzteres wurde von Rational Software Corporation, die seit 2003 IBM gehört, entwickelt [60].

Folgend das Scrum+-Modell aus der Vorlesung:

Abbildung 7: Prozessmodell-Scrum+

Dokumentations-haltung & Versionierung

Die Dokumente und Designentwürfe werden im einem geteilten Microsoft-OneDrive-Ordner festgehalten, worauf die Teammitglieder Zugriff haben. Diese Entscheidung wurde vor allem wegen des ermöglichten kollaborativen Bearbeiten von Dokumenten getroffen.

Die Versionierung erfolgt durch Git. Ein privates Repository mit dem Namen «BA-EduSOC» wurde dafür auf GitHub erstellt.

Issues

Neben der Versionierung werden auch alle Issues im Repository getrackt. Dies dient zur Nachverfolgung des Projektstandes. Dabei werden die Issues vier Phasen zugeordnet: Inception, Elaboration, Construction und Transition.

Zusätzlich werden die Issue-Zustände «not started», «planned», «in progress», «pending review» und «done» geführt.

4.6.2 Team, Rollen und Verantwortlichkeiten

Team

Im Grunde genommen herrscht eine Gleichstellung zwischen den Teammitgliedern und alle besitzen die gleiche Entscheidungsgewalt. Bei möglichen Unstimmigkeiten liegt die Verantwortung beim Projektbetreuer, diese aufzulösen und den weiteren Verlauf des Streites zu regeln.

Grundlegende Anforderungen und Designentscheide werden ansonsten kollektiv bestimmt und mit dem Betreuer Rücksprache gehalten.

Rollen

- **Alexandra Diener**
  Verantwortlich für Administratives, Docker- und Architektur-Umgebungen und mehrheitlich im Backend tätig.

- **Lukas Schiltknecht**
  Verantwortlich für Design und mehrheitlich für die visuelle Repräsentation der verschiedenen Ressourcen zuständig.

Nach der Gruppentrennung wurden die Verantwortlichkeiten von Lukas Schiltknecht von Alexandra Diener übernommen.

4.6.3 Meetings

Betreuermeeting

Einmal pro Woche, nach Absprache mit dem Betreuer entweder dienstags oder donnerstags, findet ein Betreuermeeting auf der Plattform Teams statt, dienstags um 15:00 Uhr und donnerstags um 8:00 Uhr.

Hierfür wird jeweils am Vorabend eine kurze Übersicht der erledigten Arbeit, den aufkommenden Tasks und, wenn vorhanden, Fragen oder Blockaden in Teams kommuniziert. Bei den Betreuermeetings wurde explizit auf dem privaten Repository dieser Arbeit protokolliert.

Teammeeting

Einmal pro Woche, meistens am Mittwoch um 13:00 Uhr, halten die Teammitglieder über die VoIP-Applikation Discord ein Meeting, um den letzten Stand genauer zu besprechen und den weiteren Verlauf zu spezifizieren. Damit soll vor allem die Klärung von administrativen und fachlichen Fragen stattfinden. Neben dem wöchentlichen Meeting werden, wenn nötig und nach Absprache, zusätzliche Meetings über Discord gehalten.

Nach der Trennung erübrigten sich diese Meetings natürlich.

4.6.4 Zeitmanagement

Ursprüngliches Zeitbudget

Das Frühlingssemester 2021 dauert vom 22.02.2021 bis zum 04.06.2021 und der endgültige Abgabetermin ist am 18.06.2021. Dies ergibt eine Zeitdauer von knapp 17 Wochen.

Einer Bachelorarbeit werden 12 ECTS-Punkte gutgeschrieben und pro ECTS-Punkt erwartet man etwa 25-30 Stunden Arbeit. Daraus ergibt sich ein Zeitbudget von etwa 300-360 Stunden pro Teammitglied.

	Stunden pro Woche	Stundentotal
Schiltknecht	18–22	300–360
Diener	18–22	300–360
Total	**36–44**	**600–720**

Tabelle 4: Stundenplanung Soll

Zeitbudget Post-Trennung

Seit dem 16.03.2021 wird mit einem verändertem Zeitbudget gerechnet, da ein Teammitglied die Arbeit abbrechen musste. Da diese Trennung relativ früh stattgefunden hat, wird hier eine neue Zeitbudgetrechnung festgehalten.

Nach der Trennung ergibt sich ein Zeitbudget von etwa der Hälfte der ursprünglich für diese Arbeit vorgesehenen Zeit.

Zur Zeit der Trennung wurde vom ehemaligen Teammitglied knapp 60 Stunden für dieses Projekt investiert. Damit ergibt sich folgender Zeitbudgetplan.

	Stunden pro Woche	Stundentotal
Schiltknecht	18–22	60
Diener	18–22	300–360
Total	**36–44**	**360–420**

Tabelle 5: Stundenplanung Soll Post-Trennung

Zeiterfassung

Für die Zeiterfassung wird das Tool Clockify [61] verwendet. Clockify kann über eine Webapplikation oder ein Plugin im Browser zur einfachen Zeitmessung in einem Projekt benutzt werden. Das Tool ist gratis und erlaubt es, verschiedene sogenannte Tags einfach zu erstellen. Diese Tags dienen zur Kategorisierung der erledigten Arbeit im Projekt.

Folgende Tags wurden definiert:

- **Betreuersitzung**
  Sitzungen die mit dem Betreuer stattfinden
- **Bug Fixing**
  Jegliches Debugging und Problemlösen in der Toolchain.
- **Dokumentation**
  Jegliche Arbeit an der Projektdokumentation, Projektplan, Zwischenpräsentation und weiteren Dokumenten, die während der Arbeit erstellt wurden. Das Erarbeiten und Überprüfen der Übungsszenarien, Schreiben des Wochenreports und Definieren von Issues im Repository gehören hier auch dazu.
- **Implementation**
  Jegliche Zeit, die ins Coden investiert wird, ohne aktiv an einem Debugging-Problem zu arbeiten.
- **Recherche**
  Jegliche Zeit investiert in die Evaluation von in der Arbeit womöglich verwendeten Technologien und/oder Sammlung von Daten.
- **Setup**
  Administrative Aufgaben oder Aufbau bzw. Erstellung einer Umgebung. Beispiele sind das Aufsetzen des Fileshares, Aufbau der Toolchain oder Einrichten der Repositories.
- **Teamsitzung**
  Alle Sitzungen die ohne den Betreuer stattfinden. Darunter fallen auch Implementations- oder Dokumentationsarbeit, wenn dies während der Teamsitzung erarbeitet wurde.
- **Testing**
  Zum Testing gehören manuelle Tests. Darunter sind vor allem die Analyse

von durchgeführten Usability Tests und das Ausführen von manuellen Tests an der Infrastruktur gemeint.

4.6.5 Meilensteine

Meilensteine Zeitleiste

Wie dem Zeitleisten-Diagramm zu entnehmen ist, wurden sieben Meilensteine definiert: Kickoff, Projektplan, Requirements, End of Elaboration, Beispielszenario, Demonstration, Business Cases und End of Construction.

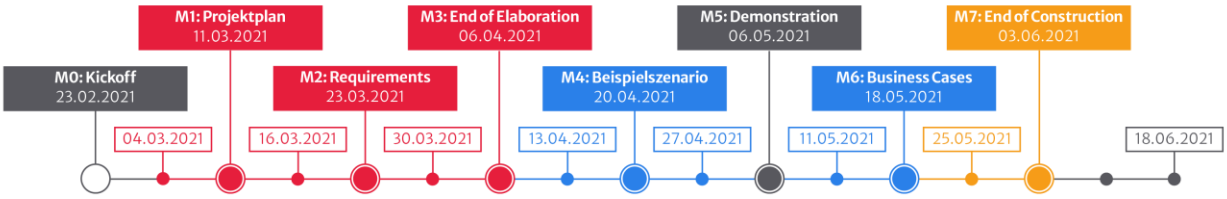


Abbildung 8: Meilensteine Zeitleiste

Meilensteine-Übersicht

Zur Übersicht der Meilensteine sind Zielsetzungen von der folgenden Tabelle zu entnehmen:

SW	KW	Meilenstein	Zielsetzung
1	08	M0: Kickoff	
3	10	M1: Projektplan	– Zielsetzung – Zeitplan – Risikoanalyse – Qualitätsmanagement – Git-Repository
5	12	M2: Requirements	– Netzwerk Architektur Entwurf – Anforderungsanalyse – Nonfunctional Requirements
7	14	M3: End of Elaboration	– User Stories – Aufgabenszenarien – Architekturentwurf – Umgebungsinitialisierung – Proof of Concept
9	16	M4: Beispiel-Szenario	– Aufgaben-Szenario auf Hacking-Lab – Verknüpfung zu WinattackLab
11	18	M5: Demonstration	– Usability Tests – Demo vom Beispiel Szenario

13	20	M6: Business Cases	–	Usability Tests
			–	Anpassungen bestehendes Szenario
			–	Mögliche weitere Szenarien
15	22	M7: End of Construction	–	Finalisierung der Szenarien
			–	Finalisierung der Dokumentation

Tabelle 6: Projektmeilensteine-Übersicht

4.6.6 Iterationen

Sprints

Wie dies im verwendeten Scrum+-Modell definiert ist, wird dieses Projekt in Sprints aufgeteilt. Die Sprintdauer ist hierbei jeweils ungefähr eine Woche, abhängig davon, ob das Betreuermeeting donnerstags oder dienstags stattfindet, da dieses Meeting als Ende der Sprints definiert wurde.

Gantt-Diagramm

Folgend ist das Gantt-Diagramm ersichtlich gemäss der Phasenaufteilung des Projektes und den Meilensteinen.

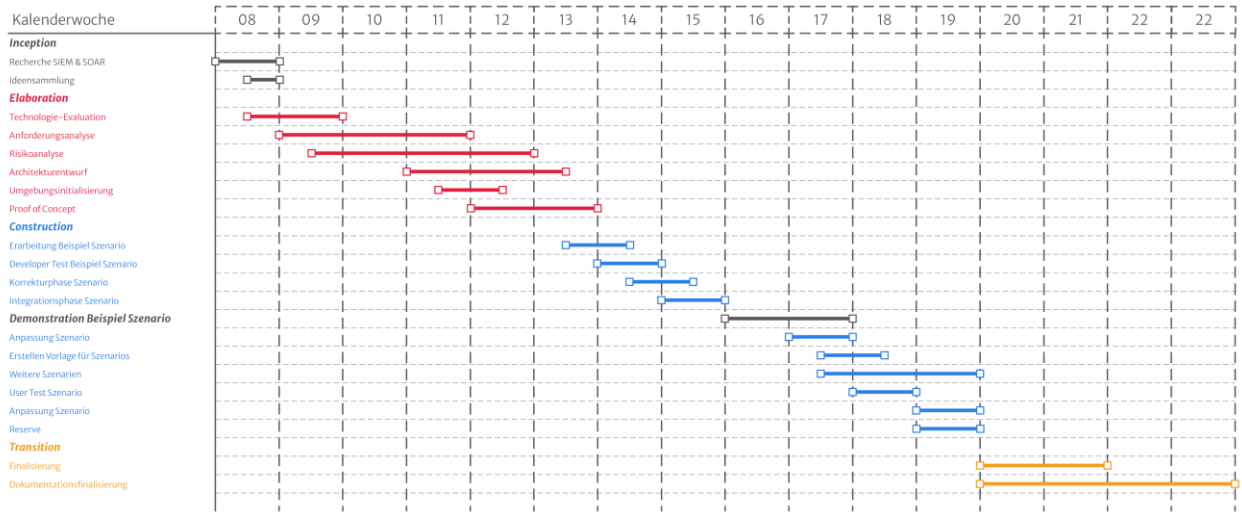


Abbildung 9: Gantt-Diagramm

4.6.7 Risikomanagement

Risikoanalyse In der Elaborationsphase wurden die folgenden Risiken erarbeitet:

#	Risiko	Beschreibung	Vorkehrung	Massnahme	[I]	[II]	[III]
1	Hacking-Lab Black-Box	Zu Beginn des Projekts ist nicht klar wie die Plattform Hacking-Lab genau aufgebaut ist und folglich wie das SOC dort eingebunden werden kann.	Demo durch Ivan Bütler über die Hacking-Lab-Plattform. Um Einblick in die Plattform zu gewähren und mögliche Fragen zu klären.	Direkte Eskalation mit dem Projektbegleiter, in diesem Fall Herr Bütler.	20	15	3
2	Know-How Cloud/ Docker	Das Wissen unter den Teammitgliedern ist bezüglich Cloud und Docker stark unterschiedlich verteilt.	Mehr Einarbeitungszeit für die Technologien einplanen und statt 10 Aufgaben nur 5 Aufgaben definieren. Dokumentation und Wikiseite aktuell halten. Bei Fragen einen Chatraum für technische Angelegenheiten führen.	Direkte Eskalation mit dem Projektbegleiter, in diesem Fall Herr Bütler, wenn Einarbeitungszeit nicht reicht.	36	30	10.8
3	Plattform-Wahl	Die 'richtige' Plattformwahl kann bei dieser Arbeit den Erfolg bedeuten, aber ebenso, wenn schlecht entschieden, den Misserfolg.	Strategie: "Fail early, fail often" So viele praktische Experimente wie möglich. Szenarien aus den praktischen Erfahrungen ableiten, nicht umgekehrt.	Saubere Dokumentation der entstandenen Arbeit und der Gründe, warum ein eventueller Rückschlag stattgefunden hat.	32	15	4.8
4	Terraform Integration	Die Arbeit mit Terraform stellt einen zentralen Bestandteil des Projekts dar und der Umgang muss von den Teammitgliedern erlernt werden. Debugging scheint nicht trivial zu sein.	Neben dem Einlesen der Dokumentation soll möglichst früh begonnen werden, Deployments und kleine Anpassungen im Code vorzunehmen, um Auswirkungen zu verstehen.	Direkte Eskalation mit dem Projektbegleiter, in diesem Fall Herr Bütler, bei gröberen Verständnisproblemen.	12	30	3.6
5	Azure Integration	Azure als Cloud-Plattform bringt gewisse Einschränkungen mit sich, was gewisse Funktionalitäten des SOC beeinflussen könnte.	Mögliche Szenarien an den Features der Cloud-Plattform und am Budget ausrichten.	Dokumentieren, sofern ein Szenario bedingt durch die Wahl des Cloudproviders nicht umsetzbar ist.	16	25	4
6	Daten-verlust	Dokumentation, bereits erarbeitete Resultate oder projektrelevante Daten könnten gelöscht, überschrieben oder anderweitig unbrauchbar gemacht werden.	Arbeit mit Versionsmanagementsystem GitHub. Sicherungskopien vor und nach jeder Bearbeitungsphase der Dokumentation, lokal.	Verlängerung der Abgabefrist beantragen.	120	3	3.6
7	Budget Überzug	Die Ressourcen auf Azure verursachen Kosten bei der OST. Dies kann schlimmstenfalls zu einem überzogenen Budget führen.	Ressourcen stets terminieren. Szenarien entsprechend den Kosten planen und diese entsprechend minimieren.	Der Projektbegleitung Bescheid geben, wenn jeweils mit den Res-sourcen gearbeitet wird.	8	10	0.8
Total					236h		30.6h
Ziffern:	[I]: Schaden auf die Laufzeit des Projekts in [h], [II]: Eintrittswahrscheinlichkeit in [%], [III]: Gewichteter Schaden in [h]						

Tabelle 7: Projektspezifische Risiken

4.7 Projektmonitoring – Ist

Überblick

In diesem Kapitel wird der Ist-Zustand gegen Ende des Projektes ausführlich beschrieben. Insbesondere wird hierbei auf die Anpassung des Projekts auf eine Einzelarbeit eingegangen.

Projekt-umkonvertierung, Gruppentrennung

Wie bereits an diversen Stellen kurz erwähnt, wurden zu Beginn der Arbeit grössere Lücken im für diese BA benötigten Wissen bei einem Teammitglied erkannt. Ursprünglicher Plan war, diesem Teammitglied mehr Einarbeitungszeit für die diversen Technologien zu geben, jedoch wurde bereits in der dritten Woche klar, dass diese zur Verfügung gestellte Zeit nicht ausreichen würde. Diese Erkenntnis war bei allen Parteien der Arbeit vorhanden und führte schlussendlich zur Eskalation in der vierten Woche und daraus resultierend wurde diese Zweier-Team-Bachelor-Arbeit zu einer Einzel-Bachelor-Arbeit umgestaltet. Da die Eskalation früh im Projekt durchgeführt wurde, konnte das Projekt relativ gut angepasst werden.

Die Anpassung der Arbeit ist vor allem im SOAR-Bereich der Anforderungen ersichtlich, da die Entscheidung getroffen wurde, den Fokus auf den SIEM-Bereich eines SOCs zu verlagern und den SOAR-Bereich zu Stretch Goals zu verschieben. Dies führte dazu, dass der SOAR-Bereich einer SOC-Plattform innerhalb dieser Arbeit kaum zum Einsatz kam.

Die durch Lukas Schiltknecht erstellten Grafiken wurden unverändert in die finale Version dieser Arbeit übernommen.

Dies beinhaltet folgende Grafiken:

- Übersicht SIEM- und SOAR-Seite aus Kapitel 3.2.1
- Use Cases Diagramm aus Kapitel 4.1.2
- Die Meilenstein-Zeitleiste aus Kapitel 4.6.5
- Das Gantt-Diagramm aus Kapitel 4.6.6

Anpassungen

Durch die Projektanpassung konnte der Schwerpunkt gut auf SIEM-fokussierte Übungsszenarien verlagert werden. Dies bedeutete natürlich die Anpassung des Scopes. Der Hauptfokus der Arbeit verweilte bei der stabilen Einbindung von einem Open Source SOC in die zu Beginn der Arbeit erhaltenen Terraform-Infrastruktur. Jedoch wurde die Anzahl der abzugebenden Übungsszenarien angepasst und, wie schon erwähnt, der Fokus auf SIEM-Aufgaben gesetzt. Zusätzliche Veränderungen sind in den ursprünglich vorgesehenen, in dieser Arbeit zu benutzenden Technologien zu finden. Technologien wie zum Beispiel Ansible wurden hier weggelassen, da sie hauptsächlich im Zusammenhang mit SOAR-Aufgaben zum Einsatz gekommen wären.

Prozessmodell

Das Projekt lief wie vorgesehen in einem typischen RUP-Modell, siehe 4.6.1, welches eine Aufteilung des Projekts in vier Phasen bedeutete.

In der ersten Phase ging es vor allem darum die Vision des Projektes mit dem Betreuer abzuklären und erste Schritte für die Kollaboration zu definieren oder zu kreieren.

In der zweiten Phase wurden vor allem bestehende Lösungen analysiert, bewertet, Diagramme erstellt und sich in diverse Technologien eingearbeitet. Dies sollte die

Chancen zu einem Aufbau einer passenden Infrastruktur erhöhen. Ausserdem wurde in dieser Phase, wegen der Erarbeitung und dem Interagieren mit diversen Technologien, früh erkannt, dass die benötigte Erfahrung bei einem Mitglied ungenügend ist. Trotz den Gegenmassnahmen sahen alle beteiligten Parteien ein, dass diese Situation eskaliert werden müsse, was mit der Entscheidung diese Arbeit zur Einzelarbeit anzupassen endete.

In der dritten Phase ging es darum, die Infrastruktur per Terraform auf Azure Cloud zu deployen, die Anwendungsszenarien zu implementieren und mit dem Hacking-Lab zu verknüpfen bzw. die Szenarien über das Hacking-Lab ausführen zu können. Ausserdem wurde eine Demo gehalten und Usability Tests durchgeführt.

In der letzten Phase ging es schlussendlich um die Finalisierung der Dokumentation, Überprüfung der Übungen auf dem Hacking-Lab und weitere administrative Aufgaben.

4.7.1 Soll-Ist-Zeitvergleich

Aufgewendete Zeit

Aus der folgenden Tabelle ist die totale aufgewendete Zeit für dieses Projekt zu entnehmen. Die Sollzeit entspricht dabei derjenigen aus Tabelle 5 Stundenplanung Soll Post-Trennung 4.6.4:

	Stunden pro Woche	Stundentotal
Schiltknecht	18–22	60
Diener	18–22	339.25
Total	**36–44**	**399.25**

Tabelle 8: Aufgewendete Zeit

Zeiterfassung

Für die genauere Untersuchung der Zeiterfassung wurde darauf verzichtet, auf die Zeit des ehemaligen Mitglieds des Projekts einzugehen. In den folgenden Abbildungen ist somit erkenntlich, wie sich die erfassten Stunden über die Dauer des gesamten Projekts bei der Studentin Alexandra Diener verteilen, bei welcher das Stundentotal bei 339 Stunden und 15 Minuten liegt.

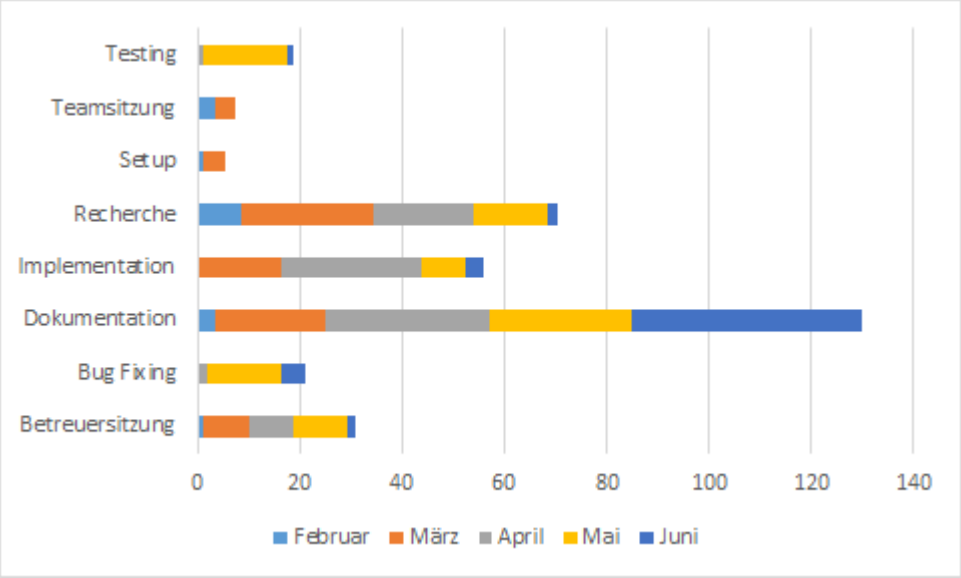


Abbildung 10: Aufgewendete Zeit pro Kategorie und Monat

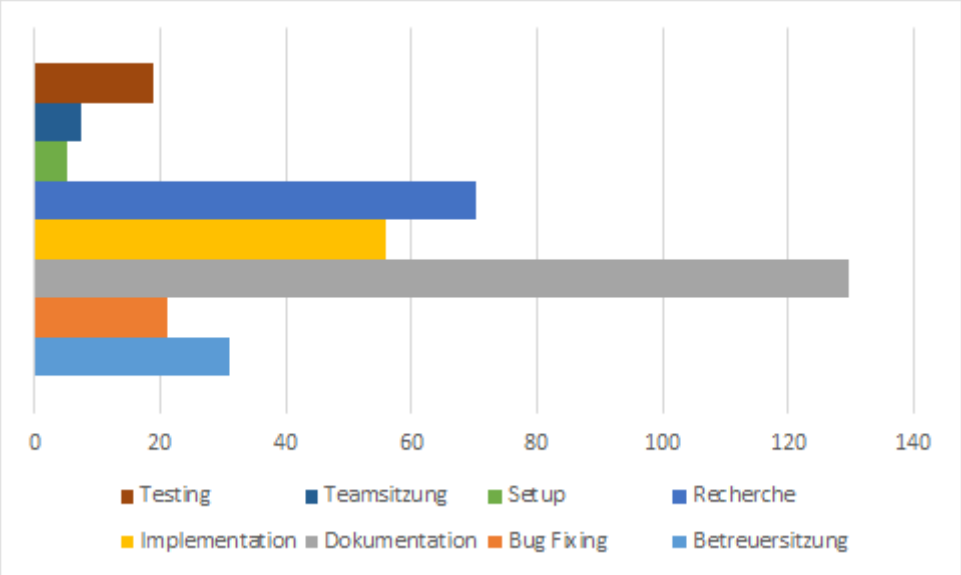


Abbildung 11: Aufgewendete Zeit pro Kategorie

Prozentuale Verteilung pro Kategorie

Dem folgenden Kreisdiagramm ist die Verteilung der total aufgewendeten Zeit auf die acht definierten Kategorien zu entnehmen. Auch hier wurde darauf verzichtet, die Untersuchung beim ehemaligen Mitglied zu machen.

Hier ist zu vermerken, dass die Kategorie Betreuersitzung nach der Trennung auch zu einer Art Ersatz der Kategorie Teamsitzung wurde und darin häufig hilfreiche Diskussionen und Untersuchungen von Technologien stattgefunden haben. Andere Kategorien wie Recherche fielen dementsprechend auch in diese Kategorie.

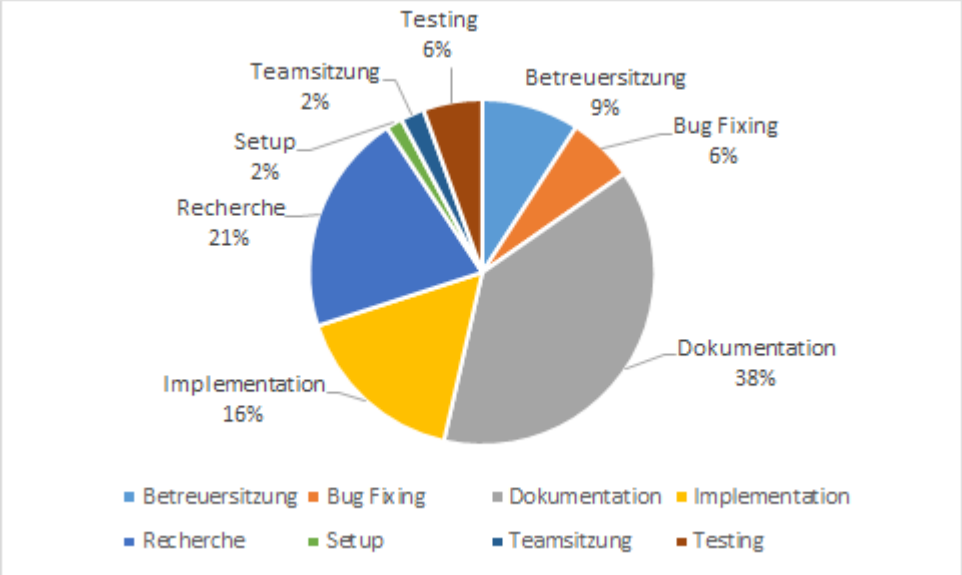


Abbildung 12: Prozentuale Verteilung pro Kategorie

4.7.2 Meilenstein-Einhaltung

Auflistung eingehaltener Meilensteine

Aus der folgenden Tabelle ist ersichtlich, wie die definierten Meilensteine aus Kapitel 4.6.5 während des Projektes eingehalten wurden.

SW	KW	Meilenstein	Eingehalten?
1	08	M0: Kickoff	Meilenstein wurde eingehalten
3	10	M1: Projektplan	Meilenstein wurde eingehalten
5	12	M2: Requirements	Meilenstein wurde eingehalten
7	14	M3: End of Elaboration	Meilenstein wurde eingehalten
9	16	M4: Beispiel-Szenario	Meilenstein wurde eingehalten
11	18	M5: Demonstration	Meilenstein wurde eingehalten
13	20	M6: Business Cases	Meilenstein wurde eingehalten
15	22	M7: End of Construction	Meilenstein wurde eingehalten

Tabelle 9: Meilenstein-Einhaltung

Auswertung

Es ist ersichtlich, dass die Meilensteine eingehalten werden konnten. Dies war jedoch nur möglich, da eine grosse Anpassung des Scopes in der vierten Semesterwoche nach der Gruppentrennung stattgefunden hat. Nach dieser Anpassung war das Erarbeiten möglichst vieler Übungsszenarien nicht mehr so stark gewichtet und es konnte relativ locker neben dem Hauptfokus der Arbeit, ein SIEM-fokussiertes SOC in die Infrastruktur einzubinden, an diesen gearbeitet werden.

4.7.3 Risiken

Eingetretene Risiken

Die definierten Risiken #2 «Know-How Cloud/Docker» und #4 «Terraform Integration» sind eingetreten und mussten eskaliert werden.

Konsequenzen

Die Eskalation des zweiten Risikos «Know-How Cloud/Docker» führte zu einer neuen Scope-Definition dieser Arbeit, da im Know-How beim anderen Gruppenmitglied grössere Lücken vorhanden waren. Schlussendlich führte die Eskalation zu einer Gruppenteilung, welche dem Projekt einen Schaden von circa 300 Stunden zugefügt hat und eine erneute Scope-Anpassung auslöste. Dieser Entscheidung wurde von allen beteiligten Parteien zugestimmt, da eine weitere gröbere Investition an Einarbeitungszeit dem Endprodukt der Arbeit nicht viel gebracht hätte. Im späteren Verlauf der Arbeit sind zusätzlich noch erhebliche Probleme mit dem Docker-traefik-Tool entstanden, welche in einer längeren Debugging-Phase und Eskalation endeten.

Das vierte definierte Risiko «Terraform Integration» ist zum Teil später im Projekt noch eingetreten, da die benutzte Terraform Version 0.14.7 einen Bug enthielt, welcher es verhinderte die Abhängigkeiten der verschiedenen virtuellen Maschinen funktionstauglich zu setzen. Dieses Problem konnte schlussendlich mit einem Upgrade auf die Terraform Version 0.15.2 gelöst werden.

Tabellarischer Überblick

In der folgenden Tabelle ist eine Gegenüberstellung des erwarteten und tatsächlich eingetretenen Schadens ersichtlich.

#	Risiko	Erwarteter Schaden [h]	Eingetretener geschätzter Schaden [h]
2	Know-How Cloud/Docker	10.8	318
4	Terraform Integration	3.6	8
Total		14.4	326

Tabelle 10: Eingetretene Risiken

4.7.4 Protokolle- und Code-Verweis

Überblick

Die Protokolle, der Code und die Übungsszenarien werden in folgenden Repositories gehalten:

Protokolle

Auf dem privaten GitHub-Repository https://github.com/CumaNeoTerra/BA-EduSOC/ unter Wiki.

Code

Der Terraform-Code und die Übungsszenarien sind unter dem privaten GitHub-Repository https://github.com/CumaNeoTerra/BA-EduSOC/ zu finden.

Der Code für die diversen Services ist auf dem öffentlichen GitHub-Repository https://github.com/Hacking-Lab/SecurityOperationsCenter zu finden.

5 Guide

5.1 Terraform Installationsverweis

Überblick

Da der Code dieser Arbeit nicht komplett frei verfügbar gemacht werden kann, wird hier lediglich auf die Installation von Terraform verwiesen. Allgemein wurde jede zusätzliche in dieser Arbeit benutzte Technologie über Terraform benutzt oder installiert.

Natürlich ist das Produkt dieser Arbeit aber über das Hacking-Lab der OST verfügbar, wenn Zugang zum Security Operation Center Lab besteht.

Terraform-Installation

Innerhalb der Arbeit wurde auf einem Linux-System, der «Hacking-Lab LiveCD» VM, Terraform installiert, weswegen hier die Installation für Linux-Systeme erfasst wird. Es ist jedoch möglich, Terraform auf anderen Betriebssystemen wie Windows oder macOS zu installieren. Unter https://www.terraform.io/downloads.html kann die 64-bit-Linux-Version heruntergeladen werden. Danach kann auf dem Linux-System das Packet auf einem Pfad entpackt werden, welcher im PATH des Systems liegt, und installiert werden.

5.2 Benutzerhandbuch

Überblick

In diesem Kapitel wird festgehalten, wie das Deployment über Terraform innerhalb der Arbeit vor allem durchgeführt wurde und auch wie der Student das Deployment über das Hacking-Lab vornehmen kann.

Terraform Deployment

Während der Entwicklungszeit wurde das Deployment auf der «Hacking-Lab LiveCD» VM durchgeführt. Es wurde hierfür ein Bash-Skript erstellt, welche zuerst die SSH-Keys neu generierte und danach das Deployment ausführte.

Hacking-Lab Deployment

Wenn Zugang zum Event «Security Operations Lab» besteht, kann im «Cyber Defense Lab» Sub-Event innerhalb der ersten gelisteten Aufgabe «SOC: Lab Deployment in Azure» über die Ressourcen deployed werden. Der schlussendlich initiierte Deployment-Manager wird danach benutzt, um die Lab-Infrastruktur sowohl zu deployen als auch zu zerstören:

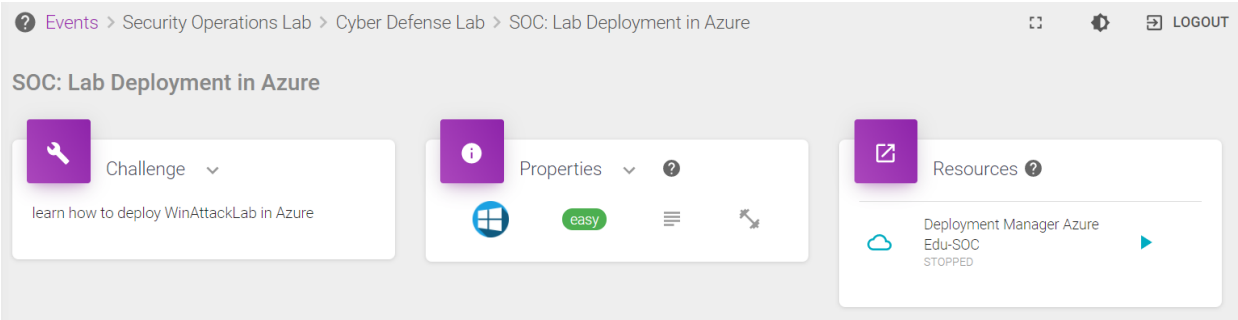


Abbildung 13: Hacking-Lab SOC-Lab Deployment Resource

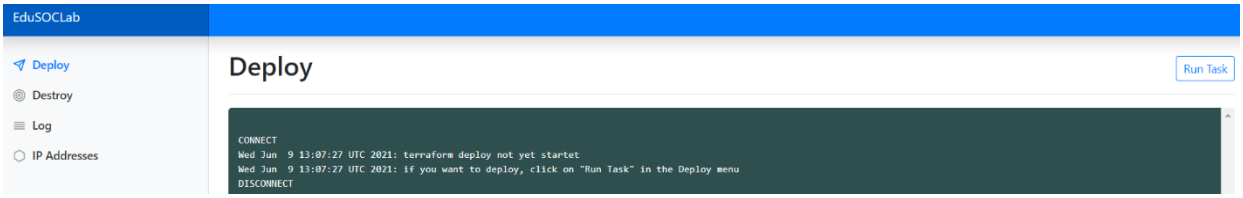


Abbildung 14: Hacking-Lab SOC-Lab Deployment

5.3 Docker-Compose-Dateien

Überblick

In diesem Kapitel werden die diversen Docker-Compose-Dateien festgehalten. Die Docker-Compose-Datei für die SOC-Plattform besteht dabei insbesondere aus den drei Services *wazuh*, *elasticsearch* und *kibana* und wurde ursprünglich von der offiziellen Wazuh-Docker-Implementation übernommen und wo nötig angepasst. Ausserdem laufen die Services im *traefik_proxy*-Netzwerk und der Grossteil der Service macht eine HTTPS-Schema und Host-Umleitung über den *traefik*-Service.

traefik

Die traefik-Docker-Compose-Datei benutzt das *traefik_proxy*-Netzwerk und besteht aus einem Service. Hier wurde auf ein Umleiten auf das HTTPS-Schema verzichtet und nur darauf geachtet, dass die Dashboard-Ansicht unter Port 8080 verfügbar ist und das zuvor erstellte, selbst signierte Zertifikat einbezogen wird. Zusätzlich zu der der Docker-Compose-Datei wurde auch eine Konfigurationsdatei für traefik erstellt. Diese Datei wird vor allem für das selbst-signierte Zertifikat benötigt.

Bei den nachfolgenden Auszügen wird der Tag `labels:` zum Teil auf `"traefik.enable=false"` gesetzt. Dies sind Services, welche nicht über traefik erreichbar sein sollen und hier zur Vorsichtsmassnahme auf `false` gesetzt werden. Dies hat vor allem den Grund, dass wenn Zeile 16, `--providers.docker.exposedbydefault`, auf `true` gesetzt werden würde, alle Services im gleichen Netzwerk über traefik geleitet werden würden, was mit dieser Konfiguration verhindert wird.

```yaml
version: "3.7"

services:
  traefik:
    image: "traefik:v2.0"
    container_name: traefik
    networks:
      - traefik_proxy
    command:
      - --log.level=DEBUG
      - --api.debug=true
      - --api.dashboard=true
      - --api.insecure=true
      - --providers.docker=true
      - --providers.docker.network=traefik_proxy
      - --providers.docker.exposedbydefault=false
      - --providers.file.filename=/configuration/traefik.yml
      - --entrypoints.http.address=:80
      - --entrypoints.https.address=:443
      - --serversTransport.insecureSkipVerify=true
    ports:
      - "80:80"
      - "8080:8080"
      - "443:443"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
      - "/opt/applic/soc_config/traefik/traefik.yml:/configuration/traefik.yml"
      - "/opt/applic/soc_config/traefik/certs/:/certs/"

networks:
  traefik_proxy:
    external: true
```

Abbildung 15: traefik Service

Traefik dynamische Konfiguration

Eine dynamische Konfigurationsdatei wurde für traefik hinzugefügt, um das selbstsignierte Zertifikat hinzuzufügen. Das Zertifikat selbst wird während dem Deployment der Infrastruktur auf dem Ubuntu-Server generiert.

```
## Setting up the middleware for redirect to https ##
http:
  middlewares:
    redirect:
      redirectScheme:
        scheme: https

## DYNAMIC CONFIG

tls:
  certificates:
    - certFile: /certs/cert.crt
      keyFile: /certs/cert.key
      stores:
       - default
  stores:
    default:
      defaultCertificate:
        certFile: /certs/cert.crt
        keyFile: /certs/cert.key
```

Abbildung 16: traefik dynamische Konfiguration

Wazuh

Im Wazuh-Service wurde nur jeweils ein Eintrag bei *labels* und *networks* hinzugefügt. Die Volumes und das Netzwerk wurden am Ende der Datei definiert.

```
# Wazuh App Copyright (C) 2021 Wazuh Inc. (License GPLv2)
version: '3.7'

services:
  wazuh:
    image: wazuh/wazuh-odfe:4.1.5
    hostname: wazuh-manager
    restart: always
    labels:
      - "traefik.enable=false"
    networks:
      - traefik_proxy
    ports:
      - "1514:1514"
      - "1515:1515"
      - "514:514/udp"
      - "55000:55000"
    environment:
      - ELASTICSEARCH_URL=https://elasticsearch:9200
      - ELASTIC_USERNAME=admin
      - ELASTIC_PASSWORD=admin
      - FILEBEAT_SSL_VERIFICATION_MODE=none
    volumes:
      - ossec_api_configuration:/var/ossec/api/configuration
      - ossec_etc:/var/ossec/etc
      - ossec_logs:/var/ossec/logs
      - ossec_queue:/var/ossec/queue
      - ossec_var_multigroups:/var/ossec/var/multigroups
      - ossec_integrations:/var/ossec/integrations
      - ossec_active_response:/var/ossec/active-response/bin
      - ossec_agentless:/var/ossec/agentless
      - ossec_wodles:/var/ossec/wodles
      - filebeat_etc:/etc/filebeat
      - filebeat_var:/var/lib/filebeat
```

Abbildung 17: Wazuh-Service

```
volumes:
  ossec_api_configuration:
  ossec_etc:
  ossec_logs:
  ossec_queue:
  ossec_var_multigroups:
  ossec_integrations:
  ossec_active_response:
  ossec_agentless:
  ossec_wodles:
  filebeat_etc:
  filebeat_var:

networks:
  traefik_proxy:
    external: true
```

Abbildung 18: Wazuh-Service Volumes

Elasticsearch

Im Elasticsearch-Service wurde auch nur jeweils ein Eintrag bei *labels* und *networks* hinzugefügt.

```
elasticsearch:
  image: amazon/opendistro-for-elasticsearch:1.13.2
  hostname: elasticsearch
  restart: always
  labels:
    - "traefik.enable=false"
  networks:
    - traefik_proxy
  ports:
    - "9200:9200"
  environment:
    - discovery.type=single-node
    - cluster.name=wazuh-cluster
    - network.host=0.0.0.0
    - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    - bootstrap.memory_lock=true
  ulimits:
    memlock:
      soft: -1
      hard: -1
    nofile:
      soft: 65536
      hard: 65536
```

Abbildung 19: Elasticsearch-Service

Kibana

Neben dem Hinzufügen des *traefik_proxy*-Netzwerks wurde der *kibana*-Service mit Labels angepasst. Diese Labels werden verwendet, um die erwähnte Umleitung durchzuführen. Hier ein Auszug vom mit Labels angepassten *kibana*-Service:

```
60    kibana:
61      image: wazuh/wazuh-kibana-odfe:4.1.5
62      hostname: kibana
63      restart: always
64      networks:
65        - traefik_proxy
66      labels:
67        - "traefik.enable=true"
68        - "traefik.http.routers.kibana_router_http.rule=Host(`wazuh.winattacklab.local`)"
69        - "traefik.http.routers.kibana_router_http.entrypoints=http"
70        - "traefik.http.routers.kibana_router_http.middlewares=redirect@file"
71        - "traefik.http.routers.kibana_router_https.entrypoints=https"
72        - "traefik.http.routers.kibana_router_https.rule=Host(`wazuh.winattacklab.local`)"
73        - "traefik.http.routers.kibana_router_https.tls=true"
74        - "traefik.http.services.kibana.loadbalancer.server.port=5601"
75        - "traefik.http.services.kibana.loadbalancer.server.scheme=https"
76      environment:
77        - ELASTICSEARCH_USERNAME=admin
78        - ELASTICSEARCH_PASSWORD=admin
79        - SERVER_SSL_ENABLED=true
80        - SERVER_SSL_CERTIFICATE=/usr/share/kibana/config/opendistroforelasticsearch.example.org.cert
81        - SERVER_SSL_KEY=/usr/share/kibana/config/opendistroforelasticsearch.example.org.key
82
83      depends_on:
84        - elasticsearch
85      links:
86        - elasticsearch:elasticsearch
87        - wazuh:wazuh
88
```

Abbildung 20: Kibana-Service

Attack-Launcher

Der Attack-Launcher besteht aus einem Service *attack-launcher*, welcher ebenfalls im *traefik_proxy*-Netzwerk erstellt wird. Über traefik-Labels wird der Host-Name gesetzt und ein Umleiten auf HTTPS-Schema erstellt. Der ursprüngliche Code des Web-Services wurde vom Betreuer dieser Arbeit für die Attack-Launcher Zwecke angepasst und für die Weiterentwicklung an die Arbeitnehmerin übergeben.

```
version: '3.7'

services:
  attack-launcher:
    build: .
    image: attack-launcher:latest
    hostname: 'attack-launcher'
    restart: on-failure # sometimes fails on ncrack build
    networks:
      - traefik_proxy
    environment:
      - HL_USER_USERNAME=hacker
      - HL_USER_PASSWORD=compass
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.attack_router_http.rule=Host(`attack-launcher.winattacklab.local`)"
      - "traefik.http.routers.attack_router_http.entrypoints=http"
      - "traefik.http.routers.attack_router_http.middlewares=redirect@file"
      - "traefik.http.routers.attack_router_https.entrypoints=https"
      - "traefik.http.routers.attack_router_https.rule=Host(`attack-launcher.winattacklab.local`)"
      - "traefik.http.routers.attack_router_https.tls=true"
      - "traefik.http.services.attack-launcher.loadbalancer.server.port=80"
      - "traefik.http.services.attack-launcher.loadbalancer.server.scheme=http"

networks:
  traefik_proxy:
    external: true
```

Abbildung 21: Attack-Launcher-Service

Mailcatcher

Um in einer Übung E-Mail Alerts zu versenden, wurde ein *mailcatcher*-Service der Infrastruktur hinzugefügt. Dieser ist simpel gehalten und dient lediglich zum Mocken von E-Mail-Alerts, ohne einen echten SMTP-Service einzubeziehen.

```
version: '3.7'

services:
  mailcatcher:
    image: schickling/mailcatcher
    restart: always
    networks:
      - traefik_proxy
    ports:
      - 25:1025
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.mail_router_http.rule=Host(`mailcatcher.winattacklab.local`)"
      - "traefik.http.routers.mail_router_http.entrypoints=http"
      - "traefik.http.routers.mail_router_http.middlewares=redirect@file"
      - "traefik.http.routers.mail_router_https.entrypoints=https"
      - "traefik.http.routers.mail_router_https.rule=Host(`mailcatcher.winattacklab.local`)"
      - "traefik.http.routers.mail_router_https.tls=true"
      - "traefik.http.services.mailcatcher.loadbalancer.server.port=1080"
      - "traefik.http.services.mailcatcher.loadbalancer.server.scheme=http"

networks:
  traefik_proxy:
    external: true
```

Abbildung 22: Mailcatcher-Service

6 Verzeichnisse

Herkunft der Vorlage

Das Dokument wurde auf der Basis einer Vorlage für Technische Berichte erstellt. Die Vorlage ist ein Element des „Werkzeugkastens Technische Berichte“ der Hochschule für Technik Rapperswil. Sie orientiert sich an Prinzipien des Strukturierten Schreibens.

6.1 Glossar und Abkürzungsverzeichnis

Begriff	Erklärung
CLI	Steht für Command Line Interface.
EDR	Steht für Endpoint Detection & Response.
EduSOC Lab	Steht für Educational Security Operation Center Lab und wurde von der Autorin dieser Arbeit definiert. Der Ausdruck EduSOC wurde innerhalb dieser Arbeit an diversen Stellen im Hacking-Lab wie auch im Code verwendet.
FQDN	Steht für Fully Qualified Domain Name. Domain-Name welcher absolute Adresse repräsentiert.
FIM	Steht für File Integrity Monitoring und überwacht Betriebsystems- oder Anwendungsdateien auf Veränderungen.
Fleet	Innerhalb dieser Arbeit im Zusammenhang mit Docker zu verstehen. Container Management & Deployment für ein Cluster.
Gitter	Eine Chat- und Networkings Plattform, welche Entwickler helfen soll, sich auszutauschen.
Hacking-Lab LiveCD	Die virtuelle Maschine auf welcher Terraform ausgeführt wird, ist die Hacking-Lab LiveCD von Compass Security, der Firma von Ivan Bütler. Sie besteht aus einer Kali Linux Maschine mit Hacking-Lab Erweiterungen. [6]
Hacking-Lab LiveCD	Eine Kali-Linux VM welche Hacking-Lab Erweiterungen enthält.
hardening	Innerhalb dieser Arbeit im Zusammenhang mit Docker zu verstehen. Die Sicherheit von Beispielsweise Docker-Container erhöhen, indem dieses hardening Security Feature vom Kernel eingesetzt wird.
Kerberos-Tickets	Kerberos ist ein verteilter, mit Tickets arbeitender Authentifizierungsdienst. Dementsprechend, wenn man von Kerberos-Tickets spricht, geht es um die Authentifizierung in TCP/IP-Netzwerken und der Zugang zur Nutzung von Services. Microsoft setzt Kerberos als Standardauthentifizierungsmethode in Windows-basierten Netzwerken ein.
MISP	Steht für Malware Information Sharing Platform und ist eine Open-Source-Plattform für Bedrohungsinformationen.

MITRE ATT&CK	ATT&CK steht für Adversarial Tactics, Techniques & Common Knowledge. Forscher von MITRE haben 2013 begonnen, Informationen über IT-basierte Angriffsarten auf Unternehmen zusammenzutragen und zu einem Framework zu systematisieren.
NIDS	Steht für Network Intrusion Detection System. Überwacht ein Netzwerk auf bösartige Aktivität oder Richtlinienverstösse.
NVD	Steht für National Vulnerability Database und ist ein Katalog von Software-Sicherheitslücken der US-Regierung, der regelmässig zu Warnzwecken verwendet wird.
Rootkit	Heimliche und gefährliche Art von Malware, die es Hackern ermöglicht sich auf Computern einzuschleusen. Rootkits können andere Prozesse, Dateien und Netzwerkverbindungen verstecken. Ohne Überwachungssystem bleiben solche Attacken unentdeckt.
SCA	Steht für Security Configuration Assessment und ist ein wichtiger Mitwirker, um die Sicherheitslage des Unternehmens zu erhöhen und die Angriffsfläche zu verkleinern.
SIEM	Steht für Security Information and Event Management. Es kombiniert diese zwei Konzepte für die Echtzeitanalyse von Sicherheitsalarmen aus Anwendungs- und Netzwerkkomponentenquellen.
SMTP	Steht für Simple Mail Transfer Protocol.
SOAR	Steht für Security Orchestration, Automation and Response. Darunter versteht man Software, welche unter den folgenden drei Kategorien zum Einsatz kommen: Bedrohungs- und Schwachstellenmanagement, Reaktion auf Sicherheitsvorfälle und Automatisierung von Sicherheitsoperationen.
Stretch Goal	Ein Ziel, welches in der Arbeit definiert wird, jedoch nicht zu Beginn im Fokus steht und als zusätzliches Arbeitspaket dient, falls die Arbeit schneller verläuft als gedacht.

Tabelle 11: Glossar und Abkürzungsverzeichnis

6.2 Quellenverzeichnis

[1] Wazuh, «GitHub Wazuh,» [Online]. Available: https://github.com/wazuh/. [Zugriff am 07 03 2021].

[2] Terraform, «Terraform,» [Online]. Available: https://www.terraform.io/. [Zugriff am 07 03 2021].

[3] TheHive-Project, «TheHiveDocs,» [Online]. Available: https://github.com/TheHive-Project/TheHiveDocs. [Zugriff am 07 03 2021].

[4] «Was ist SIEM und wie funktioniert es?,» [Online]. Available: https://www.fireeye.de/products/helix/what-is-siem-and-how-does-it-work.html. [Zugriff am 07 03 2021].

[5] «What is SOAR? Definition and Benefits,» [Online]. Available: https://www.fireeye.com/products/helix/what-is-soar.html. [Zugriff am 07 03 2021].

[6] «Hacking-Lab LiveCD,» [Online]. Available: https://livecd.hacking-lab.com/. [Zugriff am 08 06 2021].

[7] A. Froehlich, «Wie unterscheiden sich SOAR und SIEM voneinander?,» [Online]. Available: https://www.computerweekly.com/de/antwort/Wie-unterscheiden-sich-SOAR-und-SIEM-voneinander. [Zugriff am 07 03 2021].

[8] «GitHub GRR Rapid Response,» [Online]. Available: https://github.com/google/grr. [Zugriff am 11 03 2021].

[9] HELK, «GitHub HELK,» [Online]. Available: https://github.com/Cyb3rWard0g/HELK. [Zugriff am 12 03 2021].

[10] «GitHub OpenEDR,» [Online]. Available: https://github.com/ComodoSecurity/openedr. [Zugriff am 12 03 2021].

[11] MISP, «MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing,» [Online]. Available: https://misp-project.org/. [Zugriff am 07 03 2021].

[12] G. und H. , «GitHub DetectionLab,» [Online]. Available: https://github.com/clong/DetectionLab. [Zugriff am 07 03 2021].

[13] «System Hardening,» [Online]. Available: https://www.beyondtrust.com/resources/glossary/systems-hardening. [Zugriff am 07 03 2021].

[14] G. und H. , «DetectionLab Prerequisites,» [Online]. Available: https://www.detectionlab.network/introduction/prerequisites/. [Zugriff am 07 03 2021].

[15] Microsoft, «Install the Azure CLI,» [Online]. Available: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest. [Zugriff am 07 03 2021].

[16] Microsoft, «Windows Event Forwarding Guidance,» [Online]. Available: https://github.com/palantir/windows-event-forwarding. [Zugriff am 07 03 2021].

[17] Facebook, «GitHub Palantir osquery Configuration,» [Online]. Available: https://github.com/palantir/osquery-configuration. [Zugriff am 07 03 2021].

[18] Sysinternals, «sysmon-modular,» [Online]. Available: https://github.com/olafhartong/sysmon-modular. [Zugriff am 07 03 2021].

[19] Sysinternals, «GitHub Autoruns to WinEventLog,» [Online]. Available: https://github.com/palantir/windows-event-forwarding/tree/master/AutorunsToWinEventLog. [Zugriff am 07 03 2021].

[20] «Security Onion Solutions,» [Online]. Available: https://securityonionsolutions.com/. [Zugriff am 08 06 2021].

[21] TheHive-Project, «GitHub Cortex,» [Online]. Available: https://github.com/TheHive-Project/Cortex/. [Zugriff am 07 03 2021].

[22] «TheHive-Project/TheHive,» [Online]. Available: https://gitter.im/TheHive-Project/TheHive. [Zugriff am 30 04 2021].

[23] Wazuh, «The Open Source Security Platform,» [Online]. Available: https://wazuh.com/. [Zugriff am 07 03 2021].

[24] Wazuh, «Prepare your Wazuh Lab Environment,» [Online]. Available: https://documentation.wazuh.com/current/learning-wazuh/build-lab/index.html. [Zugriff am 03 07 2021].

[25] Elastic Stack, «Elastic Stack,» [Online]. Available: https://www.elastic.co/elastic-stack. [Zugriff am 07 03 2021].

[26] Puppet, «Welcome to Puppet 7.4.1,» [Online]. Available: https://puppet.com/docs/puppet/7.4/puppet_index.html. [Zugriff am 07 03 2021].

[27] Wazuh, «Wazuh Installation guide,» [Online]. Available: https://documentation.wazuh.com/current/installation-guide/index.html. [Zugriff am 07 03 2021].

[28] «Wazuh Use Cases,» [Online]. Available: https://documentation.wazuh.com/current/getting-started/use_cases/. [Zugriff am 08 06 2021].

[29] «Arkime Full Packet Capture,» [Online]. Available: https://arkime.com/index. [Zugriff am 12 03 2021].

[30] Open Information Security Foundation, «Suricata,» [Online]. Available: https://suricata-ids.org/. [Zugriff am 01 04 2021].

[31] «Zeek,» [Online]. Available: https://zeek.org/. [Zugriff am 01 04 2021].

[32] W. Z. Venema, «The Postfix Home Page,» [Online]. Available: http://www.postfix.org/. [Zugriff am 06 04 2021].

[33] «MailCatcher,» [Online]. Available: https://mailcatcher.me/. [Zugriff am 08 06 2021].

[34] «MailHog,» [Online]. Available: https://github.com/mailhog/MailHog. [Zugriff am 08 06 2021].

[35] «Splunk,» [Online]. Available: https://www.splunk.com/. [Zugriff am 06 04 2021].

[36] Terraform, «Command: graph,» [Online]. Available: https://www.terraform.io/docs/cli/commands/graph.html. [Zugriff am 21 03 2021].

[37] Graphviz , «Graphviz - Graph Visualization Software,» [Online]. Available: http://www.graphviz.org/. [Zugriff am 21 03 2021].

[38] «GitHub TheHive4Py,» [Online]. Available: https://github.com/TheHive-Project/TheHive4py. [Zugriff am 06 04 2021].

[39] AnsibleWorks, Inc., «Ansible,» [Online]. Available: https://www.ansible.com/. [Zugriff am 12 03 2021].

[40] «Sysmon,» [Online]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon. [Zugriff am 10 06 2021].

[41] «mimikatz Github,» [Online]. Available: https://github.com/gentilkiwi/mimikatz/wiki. [Zugriff am 08 06 2021].

[42] Terraform, «Terraform Language Documentation,» [Online]. Available: https://www.terraform.io/docs/language/index.html. [Zugriff am 03 07 2021].

[43] Microsoft, «Terraform on Azure documentation,» [Online]. Available: https://docs.microsoft.com/en-us/azure/developer/terraform/. [Zugriff am 07 03 2021].

[44] «Microsoft Azure,» [Online]. Available: https://azure.microsoft.com/. [Zugriff am 10 06 2021].

[45] «docker,» [Online]. Available: https://www.docker.com/. [Zugriff am 10 06 2021].

[46] «traefik,» [Online]. Available: https://traefik.io/. [Zugriff am 10 06 2021].

[47] InfoGuard, «Cyber Defense Services,» [Online]. Available: https://www.infoguard.ch/de-ch/angebot/cyber-defence-services. [Zugriff am 11 03 2021].

[48] Wazuh, «Learning Wazuh,» [Online]. Available: https://documentation.wazuh.com/current/learning-wazuh/index.html. [Zugriff am 11 03 2021].

[49] InfoGuard, «Response Services,» [Online]. Available: https://www.infoguard.ch/de-ch/angebot/cyber-defence-services/response-services. [Zugriff am 11 03 2021].

[50] Cynet, «Incident Response Plan Template,» [Online]. Available: https://www.cynet.com/incident-response/incident-response-plan-template/. [Zugriff am 11 03 2021].

[51] Kaspersky, «Incident Response Guide,» [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07171449/Incident_Response_Guide_eng.pdf. [Zugriff am 11 03 2021].

[52] «ISO/IEC 9126,» [Online]. Available: https://de.wikipedia.org/wiki/ISO/IEC_9126. [Zugriff am 04 04 2021].

[53] «FileBox,» [Online]. Available: https://www.filebox-solution.com/. [Zugriff am 24 05 2021].

[54] «Terraform Azure Provider,» [Online]. Available: https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs. [Zugriff am 24 05 2021].

[55] «Azure Command-Line Interface (CLI) documentation,» [Online]. Available: https://docs.microsoft.com/en-us/cli/azure/?view=azure-cli-latest. [Zugriff am 24 05 2021].

[56] «Pricing Calculator,» [Online]. Available: https://azure.microsoft.com/en-us/pricing/calculator/. [Zugriff am 30 04 2021].

[57] «MITRE ATT&CK,» [Online]. Available: https://attack.mitre.org/. [Zugriff am 08 06 2021].

[58] I. Ayadhi, «Deploying of infrastructure and technologies for a SOC as a Service (SOCasS),» [Online]. Available: https://medium.com/@ibrahim.ayadhi/deploying-of-infrastructure-and-technologies-for-a-soc-as-a-service-socass-8e1bbb885149. [Zugriff am 10 06 2021].

[59] «GoogleGroups - wazuh & thehive integration,» [Online]. Available: https://groups.google.com/g/wazuh/c/_pPhkjtGSzs. [Zugriff am 13 06 2021].

[60] Paul Levy and Mike Devlin, «Wikipedia: Rational Unified Process,» IBM, 2003. [Online]. Available: https://en.wikipedia.org/wiki/Rational_Unified_Process. [Zugriff am 07 03 2021].

[61] «Clockify,» 2009. [Online]. Available: https://clockify.me/]. [Zugriff am 07 03 2021].

[62] Elastic Stack, «Kibana,» [Online]. Available: https://www.elastic.co/kibana. [Zugriff am 07 03 2021].

6.3 Abbildungen

6.4 Tabellen

7 Danksagung und Fazit

7.1 Danksagung

Mein Dank gilt meinem Betreuer, Herr Ivan Bütler, für sein grosses Verständnis, seine Geduld, seine Verfügbarkeit und grosse Hilfsbereitschaft durch das Projekt hindurch.

Ich danke Christoph Streiff für das Gegenlesen und das Feedback der Dokumentation, sowie seine ermunternden Worte und hilfreichen Dokumentations-Tipps.

Darüber hinaus danke ich meinen vier Usability Testern, namentlich:
Matteo Demasi, Julia Fritsche, Jana Kravarik und Christoph Streiff.

Last but not least, danke ich meiner Familie, ganz speziell meiner Mutter Angela Diener, für ihr Verständnis und Geduld in dieser nicht ganz einfachen Zeit.

7.2 Fazit

Beim «Security Operation Center Lab» Projekt handelte es sich um ein riesiges Unterfangen, welches Know-How zu diversen Technologien benötigte. Dies war durch das Projekt hindurch nicht einfach zu jonglieren und, wie zu Beginn der Arbeit ersichtlich, nicht für jeden geeignet durchzuführen. Die turbulenten ersten paar Wochen waren wahrhaftig keine einfache Zeit.

Trotz meiner Vorerfahrung in vielen der Technologien, die in dieser Arbeit benutzt wurden, war es dennoch eine grosse Herausforderung sich diesem Projekt zu widmen und diese Technologien zusammen mit Terraform und der Infrastruktur in Einklang zu bringen. Ich war dementsprechend sehr glücklich darüber, dass sich mein Betreuer mehrmals pro Woche Zeit genommen hat und wir mögliche Probleme ausdiskutiert haben oder er mir Einführungen in gewisse Themengebiete gegeben hat.
Mit jeder weiteren Woche, mit Ausnahme der letzten zwei Wochen, war die Recherche immer ein grosser Teil der Arbeit und es wurde im Endeffekt in dieser Arbeit extrem viel an zusätzlichem Know-How angeeignet. Dies war nicht immer ganz einfach für die Ausdauer und ich bin stolz darauf, dass schlussendlich ein funktionstaugliches SOC-Lab entstanden ist.

Bezüglich den Übungsszenarien bin ich generell glücklich über das Resultat, jedoch sollte bei einer Weiterarbeit die Prozedur der Übungsszenarienerarbeitung geändert werden. Das wiederholte Überprüfen und Korrigieren der Übungsszenarien und Übungen auf dem GitHub Repository und dem Hacking-Lab haben im Endeffekt sehr viel Zeit beansprucht. Ich denke hier wäre es besser gewesen, wenn eine Einarbeitungszeit in das Erstellen von Übungen im Hacking-Lab stattgefunden hätte und von Beginn an das Übungsszenario dort festgehalten und zu einer Übung ausgearbeitet worden wäre.

Alles in allem bin ich jedoch sehr glücklich mit dem Endresultat dieser Arbeit und hoffe, dass sich das Resultat gut für das kommende Cyber Defense Modul eignen wird.

Anhang

A Übungsszenarien

Überblick

Die im Projekt entwickelten Übungsszenarien werden hier festgehalten. Das Format unterscheidet sich dabei zum Rest des Dokuments, da die Szenarien im Markdown-Format ausgearbeitet wurden.

A.1 Übungsszenario 0

The main target in this first exercise is to have first interactions with the Wazuh-SOC. The focus is set to filtering possibilities, rules and understanding what a Wazuh Agent and Wazuh Manager are.

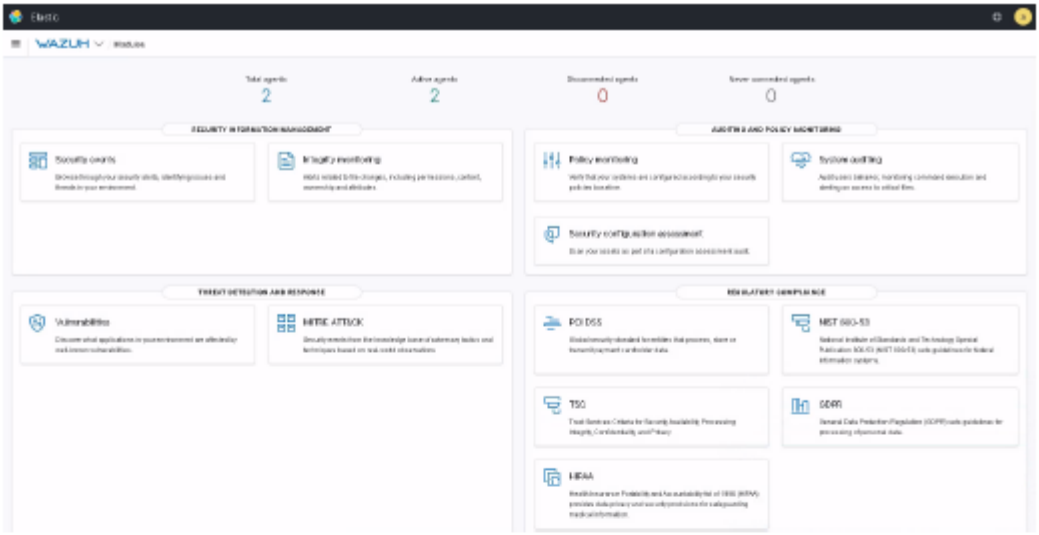
Prerequisites

- WinAttackLab is running
  - Wazuh is ready to use (https://wazuh.winattacklab.local)
- SSH-brute-force attack is executable
  - A script is provided and can be executed multiple times
  - Attacker is the Kali-Linux Client & the victim is the Ubuntu Client

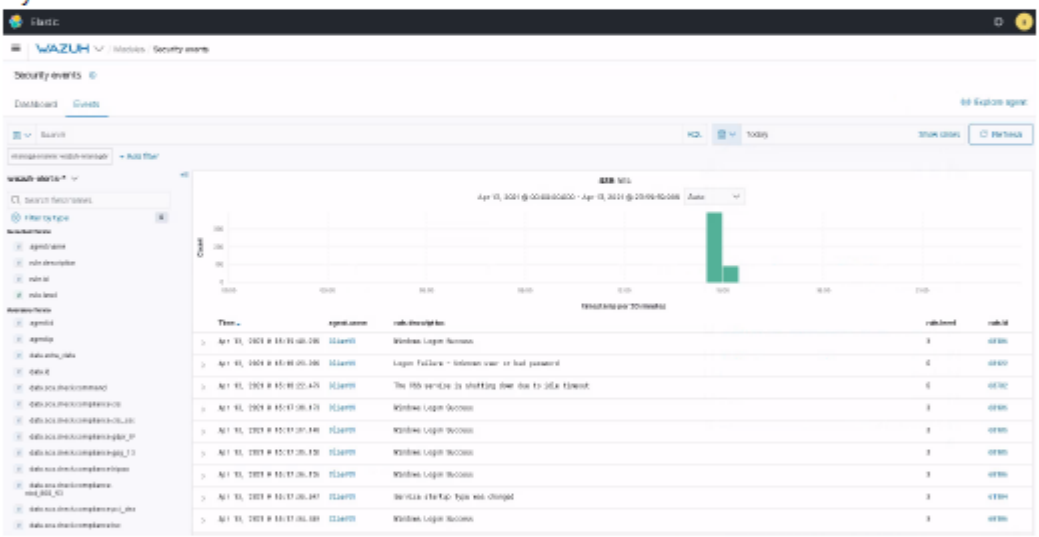
Tasks

1. Log-in to Wazuh
   - Open Chrome browser on https://wazuh.winattacklab.local
   - admin:admin

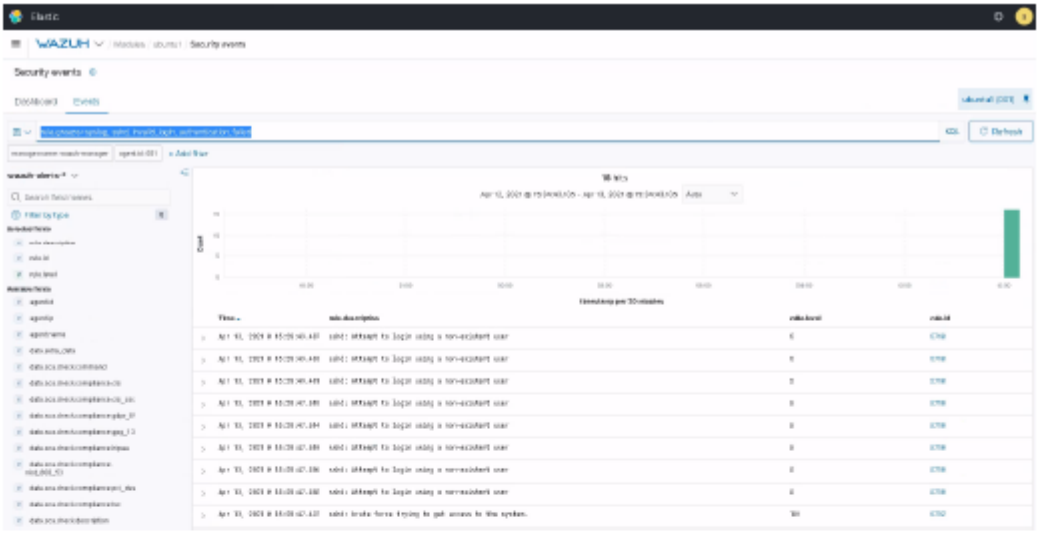
   - You should see an overview of your agents and different sections of Wazuh. What exactly is an agent?



2. Explore a little, focus on Wazuh's capabilities and where to find and filter Security Events.
    - Try to filter some events and understand how Wazuh classifies rules.



    - What is the Wazuh Manager exactly?
    - Try to find the path, where the Wazuh Agent is installed on the Windows Client. Which file is used to configure the connection to the Wazuh Manager and where exactly in the file is it defined?
        - `C:\Program Files (x86)\ossec-agent\`
        - `ossec.conf`
3. Launch the SSH-brute-force attack and check if the attack is shown in the Security Events
    - Should be the case, however, there might be too many events to find this event quickly -> Too much noise
4. Set a filter, so you can see the event quicker:
    - E.g. `rule.groups=syslog, sshd, invalid_login, authentication_failed`
    - Attack can be launched multiple times if needed
    - You should see the brute force security event better now. Why is an event classified with rule id 10 and other times with 5?



Learning Targets

1. Interacted with the Wazuh-SOC and gained an overview of its capabilities.
2. Understand what a Wazuh Agent and what a Wazuh Manager is.
3. Created a filter and understand the importance of a good filter in a SOC-System.
4. Understand `rules` and Wazuh's classification of them.

Additional Resources

In this exercise an additional script is used, which launches a SSH-brute-force attack. It can be triggered multiple times.
The script is hosted on the Kali-Linux VM.

Additionally, a closer look should be taken at the Wazuh ruleset and classification:

- Wazuh_Ruleset.pdf
- Rules classification documentation

A.2 Übungsszenario 1

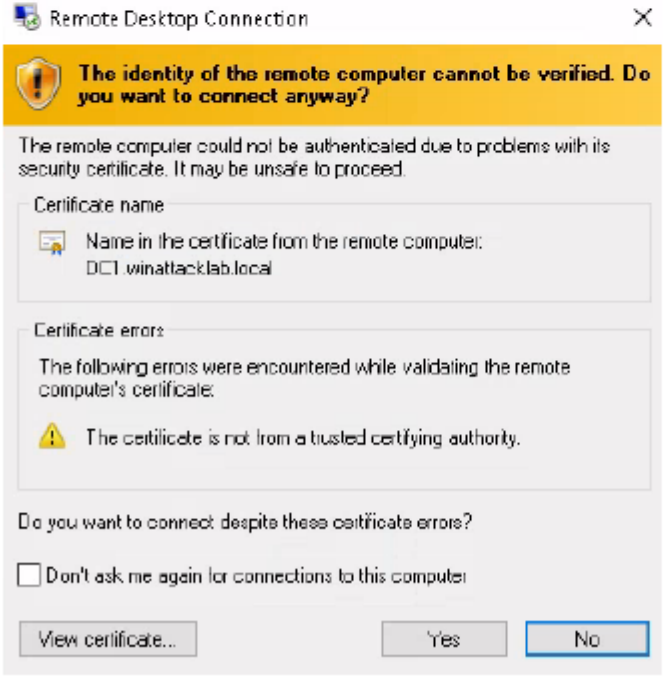
The goal of this exercise is to understand Active Directory (AD) and log-in logs. Additionally, you should learn how to distribute, configure and deploy Wazuh Agents over an AD through Group Policy.

Prerequisites

- WinAttackLab is running
  - Wazuh is ready to use (https://wazuh.winattacklab.local)
- Student/User can access the Windows 10 Client via RDP and log-in with AD credentials
- RDP log-in is shown in Event View

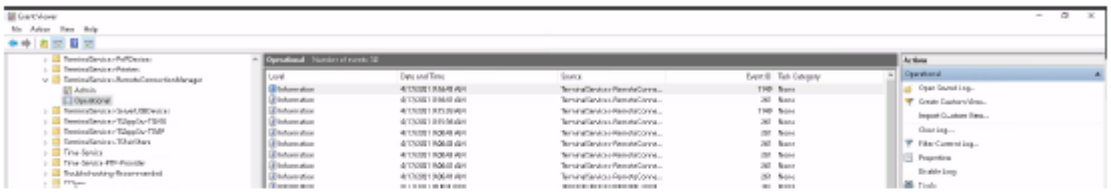
Tasks

1. Log-in to the Windows 10 Client and access the Wazuh Dashboard. Can you see any log-in attempts that are registered from another Windows machine, besides the current Windows Client?
   - Currently only an Ubuntu and Windows client have a Wazuh Agent
2. Log-in to the Domain Controller via RDP
   1. Run mstsc.exe
   2. Log-in credentials:
      - dc1.winattacklab.local
      - [dc1_username:password]
   3. Accept certificate errors (possibly hidden in the background after log-in try)



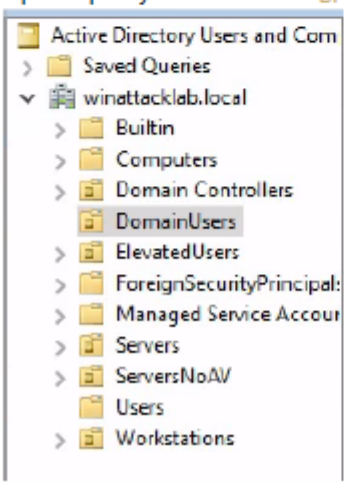
3. Check if a Network Connection Event with ID 1149 was logged on the Domain Controller, thanks to the log-in

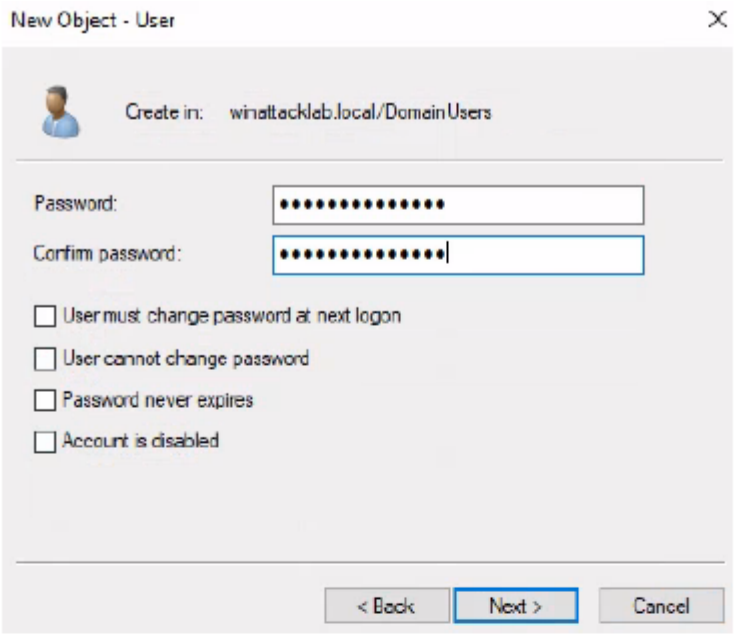
- Search for Event Viewer and start it
- Go to TerminalServices-RemoteConnectionManager:
  `Application and Services Logs --> Microsoft --> Windows --> TerminalServices-RemoteConnectionManager`
- Event with ID 1149 should be viewable. What does this event mean exactly? What are the phases of a RDP-connection?



4. Now, via AD & Group Policy configurations, you will create a Wazuh Agent on each server in winattacklab.local

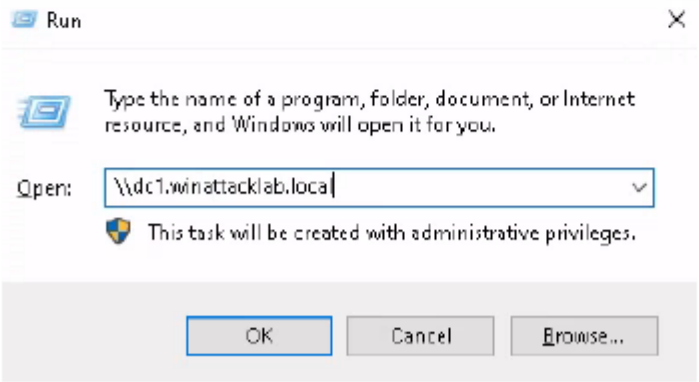
   1. In Server Manager under Tools, in the top right corner, open `Active Directory Users and Computers` & `Group Policy Management`
   2. In `Active Directory Users and Computers` you will create a new User in `DomainUsers`, use rightclick.
      - Deactivate password change in your next log-on.
      - This step is mainly here to understand how to add a new user and that it is important to create a new user under DomainUsers if domain access rights are needed.
      - Update policy via Powershell: `gpupdate /force`





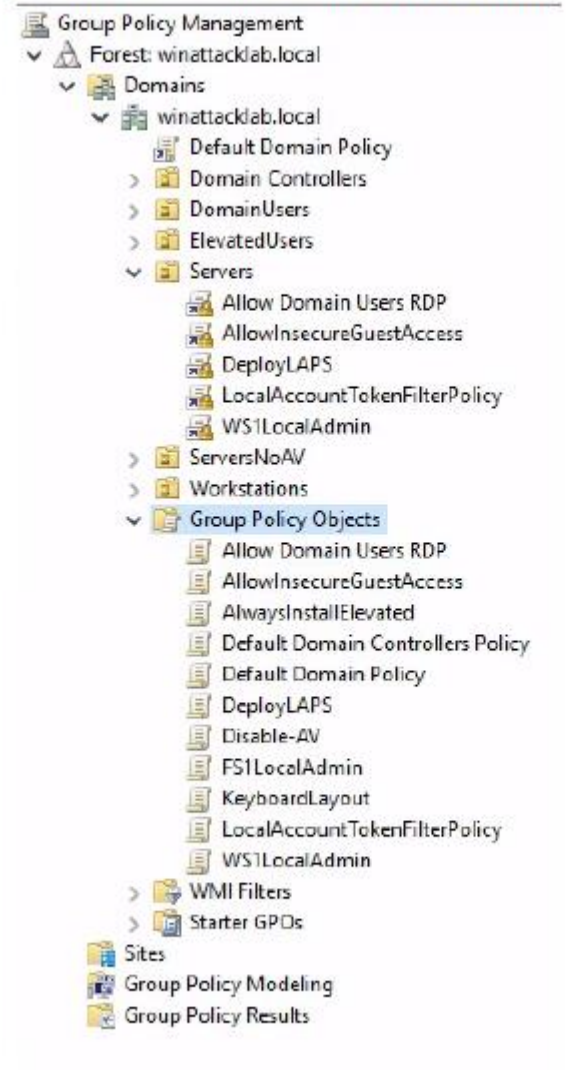
- Back on the Windows Client, try to log-in to the Windows Server with the newly created user
  - ws1.winattacklab.local
  - [newuser:password]
- Since there is no Wazuh Agent running on the Windows Server, you should not see any log-in attempt event in Wazuh coming from this Windows Server. Is this the case?
- Disconnect from the Windows Server

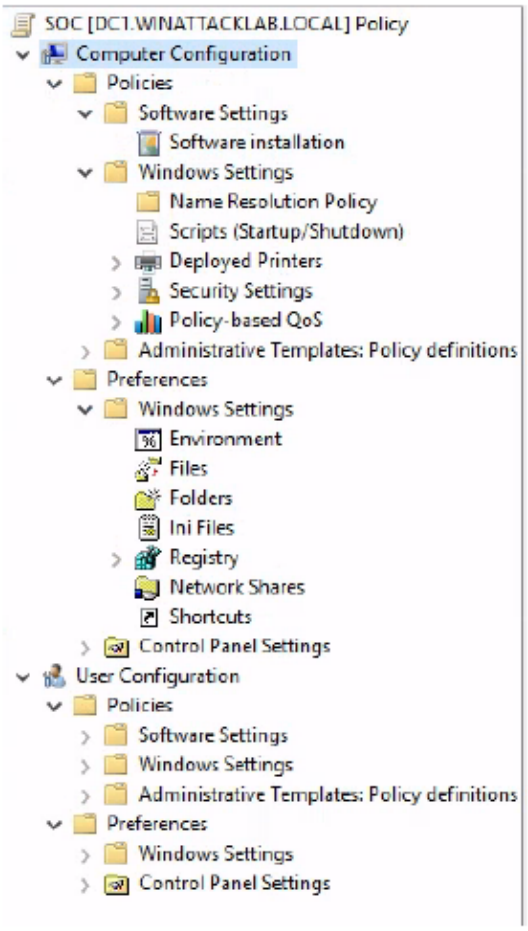
3. Back on the Domain Controller, you will add a prepared Powershell script to a path, which resides under the dc1.winattacklab.local AD.
   - On the Domain Controller, start the `Run` tool. Open `\\dc1.winattacklab.local` and create the folder `soc` under `\\dc1.winattacklab.local\SYSVOL\winattacklab.local\`.



   - The prepared Wazuh Agent installationscript is found under `C:\Users\LAB_ADMIN\install_wazuh_agent.ps1`. Copy it to the new folder `soc`.
4. In `Group Policy Management` under `Group Policy Objects`, create a new Group Policy Object.

- Give the item the name SOC and edit it via rightclick.





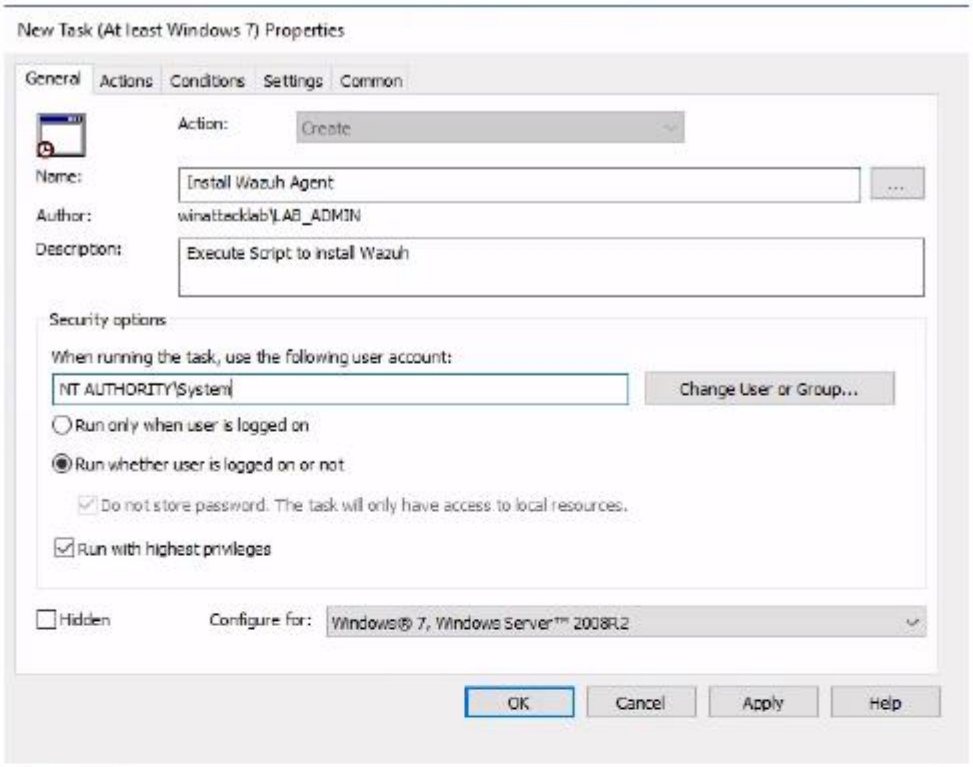
- Under `Computer Configuration --> Preferences --> Windows Settings` in `Folders`, create a new folder. Use the path `C:\soc`
- Under Common
  - Enable `Apply once and do not reapply.`

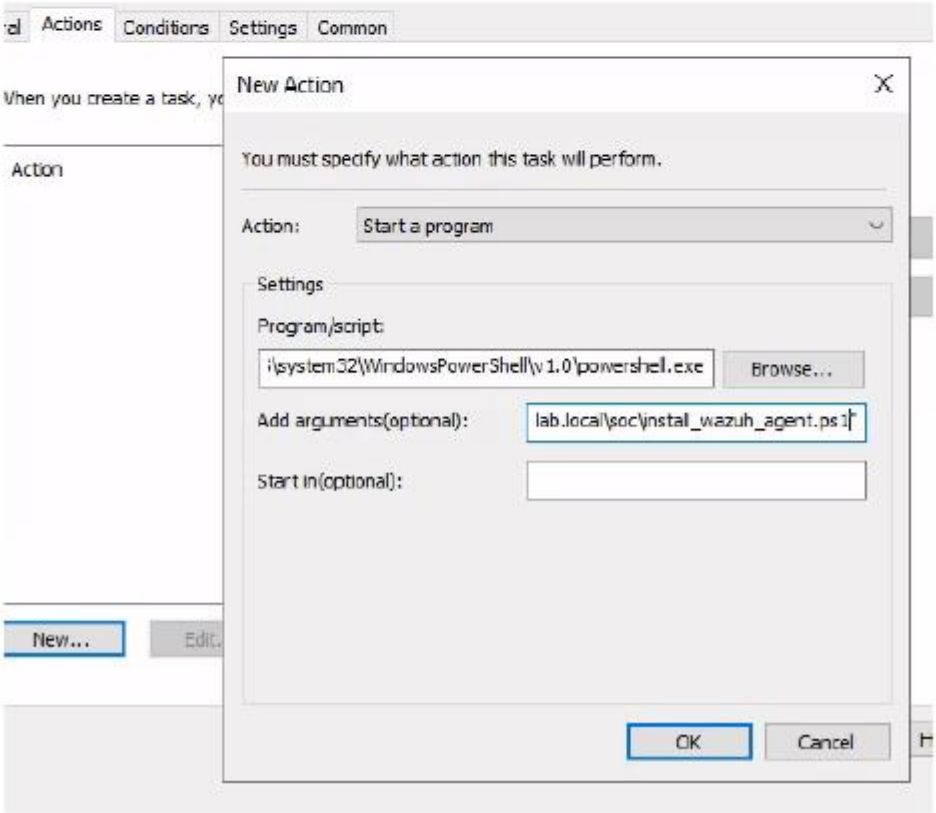
- In `Files`, create a new file.
  - In Source define the path `\\dc1.winattacklab.local\SYSVOL\winattacklab.local\soc\install_wazuh_agent.ps1`
  - In Destination define `C:\soc\install_wazuh_agent.ps1`

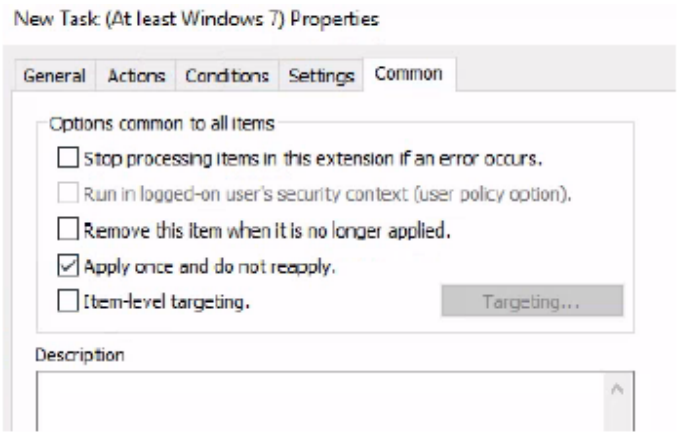
- Click Apply and OK.

- Under `Computer Configuration --> Preferences --> Control Pannel Settings` in `Scheduled Tasks`, create a new `Immediate Task (At least Windows 7)` and configure it:
  - Under `General`
    - Click on `Change User or Group`, `Check Names` on SYSTEM and click `OK`.
    - Choose `Run whether user is logged on or not` and enable `Run with highest privileges`.

    - Configure for: should be set to Windows 7, Windows Server 2008R2.
- Under Actions
    - Create a new Action and set it to Start a program
    - Set Program/script to C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.exe
    - Set Add arguments(optional) to -ExecutionPolicy Bypass -command "& C:\soc\install_wazuh_agent.ps1"
- Under Common
    - Enable Apply once and do not reapply.


New Task (At least Windows 7) Properties
General
Actions
Conditions
Settings
Common
Action:
Create
Name:
Install Wazuh Agent
Author:
winattecklab\LAB_ADMIN
Description:
Execute Script to install Wazuh
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


Actions
Conditions
Settings
Common
Action
New...
New Action
You must specify what action this task will perform.
Action:
Start a program
Settings
Program/script:
i\system32\WindowsPowerShell\v1.0\powershell.exe
Browse...
Add arguments(optional):
lab.local\soc\install_wazuh_agent.ps1
Start in(optional):
OK
Cancel



5. Drag-and-drop the new policy to `Servers` and also to `ServersNoAV`. This will set the new policy for a Windows Server and a File System.
6. Now enforce the new group policy configurations. This is done, so you don't have to wait until the new configurations are applied on the servers. Open a Powershell on the Domain Controller and execute the following lines:
   - `$computers = Get-ADComputer -Filter *`
   - `$computers | ForEach-Object -Process {Invoke-GPUpdate -Computer $_.name -RandomDelayInMinutes 0 -Force}`

5. Now, log-in to the Windows Server with the new user again and check in Wazuh if the log-in attempt was registered. In Wazuh you should now also notice the additional Agents.

6. Create a filter in Wazuh to only show the unsuccessful login attempts. Can you find an event with rule ID 10? Why is this the case?

Learning Targets

1. Understand how to configure something in Active Directory and Group Policy
2. Understand how to deploy a Wazuh Agent via AD and Group Policy
3. Made some interactions with the Event Viewer
4. Read about the main phases of a RDP-connection and their logging. Understand that Event 1149, despite the name, does not mean, that the user was authenticated, but that a network connection was established.
   1. Network Connection
      - ID: 1149 - "User authentication succeeded"
   2. Authentication - ID: 4624 & 4625
      - ID: 4624 - "An account was successfully logged on"
      - ID: 4625 - "An account failed to log on"
   3. Logon
      - ID: 21 - "Remote Desktop Services: Session logon succeeded:"
      - ID: 22 - "Remote Desktop Services: Shell start notification received:"
   4. Session Disconnect/Reconnect
      - ID: 24 - "Remote Desktop Services: Session has been disconnected:"
      - ID: 25 - "Remote Desktop Services: Session reconnection succeeded:"
      - ID: 39 - "Session X has been disconnected by session Y"

- ID: 40 - "Session X has been disconnected, reason code Z"

5. Logoff
   - ID: 23 - "Remote Desktop Services: Session logoff succeeded:"

Additional Resources

Blog: https://ponderthebits.com/2018/02/windows-rdp-related-event-logs-identification-tracking-and-investigation/

A.3 Übungsszenario 2 Teil 1

The main targets in this exercise are to interact with Sysmon, configuring it to detect the hacker-tool mimikatz and Know-How of how to configure rules in Wazuh.

Prerequisites

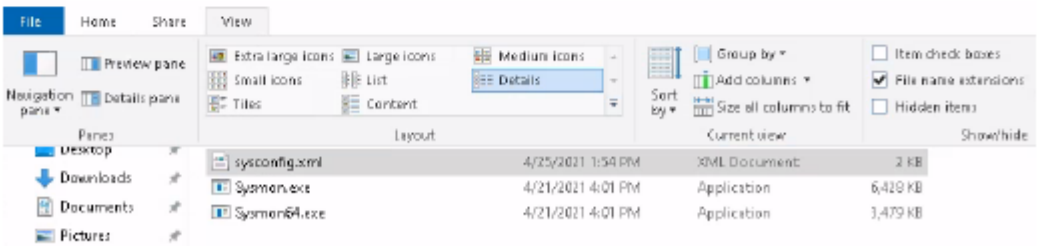
- WinAttackLab is running
  - Wazuh is ready to use (https://wazuh.winattacklab.local)
- Student/User can access the Windows 10 Client via RDP and log-in with lab_admin credentials.
- Sysmon xml configuration files -> added prepared sysconfig.xml for the students to try out and a solutionsysconfig.xml

Tasks

1. In this first step you need to install and configure Sysmon to register process creation which includes `lsass.exe` and `mimikatz.exe` on the Domain Controller. The lsass process is known to be associated with the hacker tool mimikatz. What is mimikatz? It is a tool developed in 2007 and is still used in obtaining Windows credential sets via RAM, hash dumps, Kerberos exploitation, as well as pass-the-ticket and pass-the-hash techniques.
   1. Open a Powershell as local admin (enter the lab_admin credentials) and enter the following:

```
$client = New-Object System.Net.WebClient
$client.DownloadFile("https://download.sysinternals.com/files/Sysmon.zip",
"C:\Sysmon.zip")

Expand-Archive "C:\Sysmon.zip" -DestinationPath "C:\Sysmon"
Remove-Item -Path "C:\Sysmon.zip" -Recurse
```

   2. You will need to add a configuration file for Sysmon under `C:\Sysmon`. Create the file `sysconfig.xml` and copy paste the content of the xml-file provided in this exercise.
   - If you struggle with setting the file type, enable `File name extensions` under `View`:



   - If you are interested in a general Sysmon configuration file you can find it here. However, the file cannot be directly downloaded, because of formatting reasons. In short it will trigger the following error if downloaded directly `XML file contains fatal error 41:Specification mandate value for attribute data`. Instead you need to copy paste the raw-data into a xml-file.

3. Install Sysmon with the configuration file via Powershell with `.\Sysmon64 -accepteula -i sysconfig.xml`. After the installation you should find Sysmon in the Event Viewer under `Applications and Services Logs --> Microsoft --> Windows`

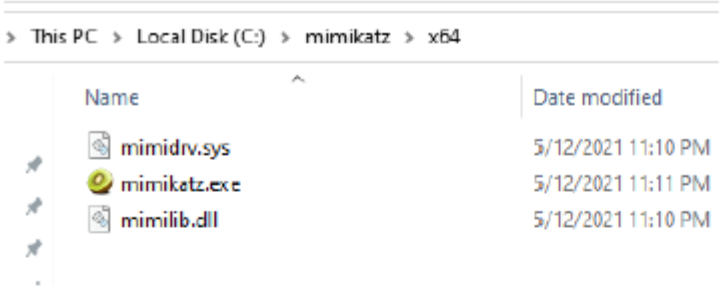
- If you need to update the Sysmon configuration, you can do so with `.\Sysmon64 -c sysconfig.xml`

2. The current configuration will not suffice to detect mimikatz in action, it will only find different types of script executions, like PowerShell. Before we change the configuration however, we will install mimikatz first.
   1. First you need to make sure that the real-time protection is deactivated, otherwise mimikatz executable and binaries will be deleted. Go to `Virus & threat protection settings` and deactivate it if it is not:

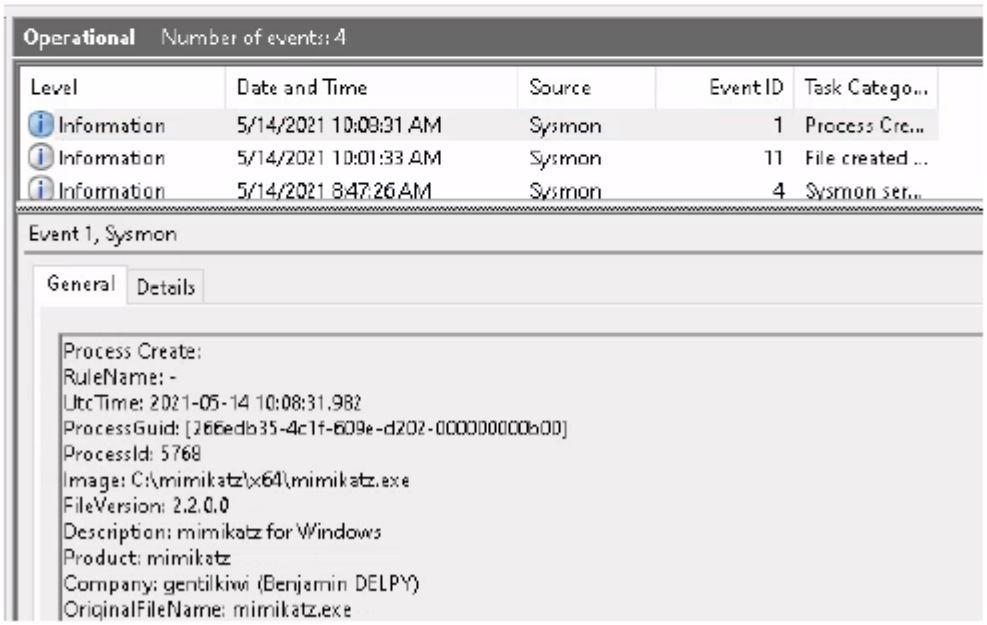
   2. Similar to the Sysmon installation, we will use PowerShell (Releases are found here):

```
$client = New-Object System.Net.WebClient
$client.DownloadFile("https://github.com/gentilkiwi/mimikatz/releases/downlo
ad/2.2.0-20210512/mimikatz_trunk.zip", "C:\mimikatz.zip")

Expand-Archive "C:\mimikatz.zip" -DestinationPath "C:\mimikatz"
Remove-Item -Path "C:\mimikatz.zip" -Recurse
```

3. Can you find the mimikatz executable? If not, make sure your real-time protection is deactivated.



4. Try executing mimikatz over a PowerShell with `.\mimikatz.exe`. Currently you should not find any registered activity under Sysmon in the Event Viewer, please configure the xml-file and update the Sysmon configuration that it does detect it. (A solutionsysconfig.xml is provided for this step). Re-execute mimikatz to see when an event was caught caused by mimikatz. Additionally, the task is to find non-encrypted mimikatz activity for the ease of this exercise. Therefore, it suffices to configure the xml file to check for mimikatz.exe process creation and for lsass.exe and mimikatz.exe process access. Finally, you can refresh the Event Viewer with F5 and check for the following event:



3. Moving on, we will check how to configure the Wazuh settings to see Sysmon's findings and also define the severity of the finding. Wazuh provides an overview on How to collect Windows logs in general.

Here an excerpt of the most important information about eventchannel for Wazuh >= 3.9.0:

Source	Rule IDs	Rule file
Base rules	60000 - 60099	0575-win-base_rules.xml
Security	60100 - 60599	0580-win-security_rules.xml
Application	60600 - 61099	0585-win-application_rules.xml
System	61100 - 61599	0590-win-system_rules.xml
Sysmon	61600 - 62099	0595-win-sysmon_rules.xml
Windows Defender	62100 - 62599	0600-win-wdefender_rules.xml
McAfee	62600 - 63099	0605-win-mcafee_rules.xml
Eventlog	63100 - 63599	0610-win-ms_logs_rules.xml
Microsoft Security Essentials	63600 - 64099	0615-win-ms-se_rules.xml
Others	64100 - 64599	0620-win-generic_rules.xml

1. As you can see, Sysmon rules are defined in the rule file `0595-win-sysmon_rules.xml`, which is located in the Wazuh Manager on the Ubuntu Server. The actual file is found under `/var/ossec/ruleset/rules` on the Ubuntu Server, but it will be viewed via the dashboard in a later step.
2. For the Wazuh agent on the Windows Client to collect Sysmon events, you need to configure the `ossec.conf` file in `C:\Program Files (x86)\ossec-agent\`. Add the following `<localfile>` entry:

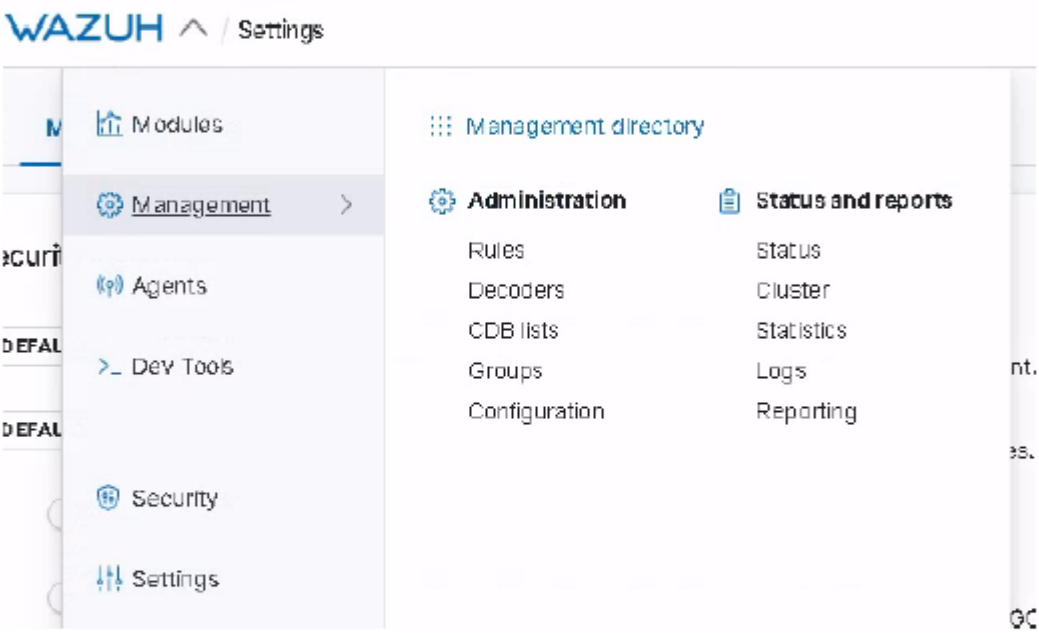
ossec - Notepad

File Edit Format View Help

```
    <events_per_second>500</events_per_second>
  </client_buffer>

  <!-- Log analysis -->
  <localfile>
    <location>Application</location>
    <log_format>eventchannel</log_format>
  </localfile>

  <localfile>
    <location>Microsoft-Windows-Sysmon/Operational</location>
    <log_format>eventchannel</log_format>
  </localfile>
```

```
<localfile>
  <location>Microsoft-Windows-Sysmon/Operational</location>
  <log_format>eventchannel</log_format>
</localfile>
```

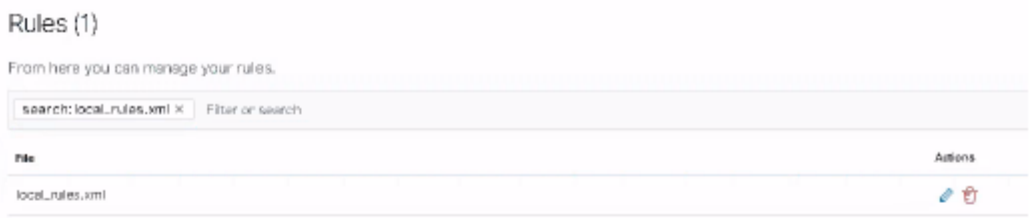
3. Restart the Windows Client Wazuh agent via PowerShell: `Restart-Service -Name wazuh`
   You should see the "Ossec agent started" rule description with id 503 in Wazuh.
4. Now, concerning the rules file on the Ubuntu Server. These rules files can also be found in the Wazuh Dashboard under `Management --> Rules`:



5. Take a look at the default Sysmon rules file `0595-win-sysmon_rules.xml` by searching for it, clicking on `Manage rules` and clicking on the eye icon. Note: all of these rules files are the default state and are generally not edited. If configurations need to be made, they are done in a newly defined `local_rules.xml` file which overrules other rules if defined there:

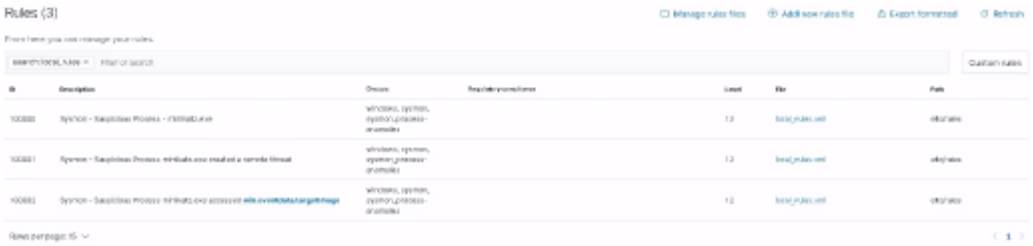
6. Now go back and search for `local_rules.xml`. In this file you will define the additional mimikatz related configuration for Wazuh. Click on the pencil icon to edit it and overwrite the

current content with the following:

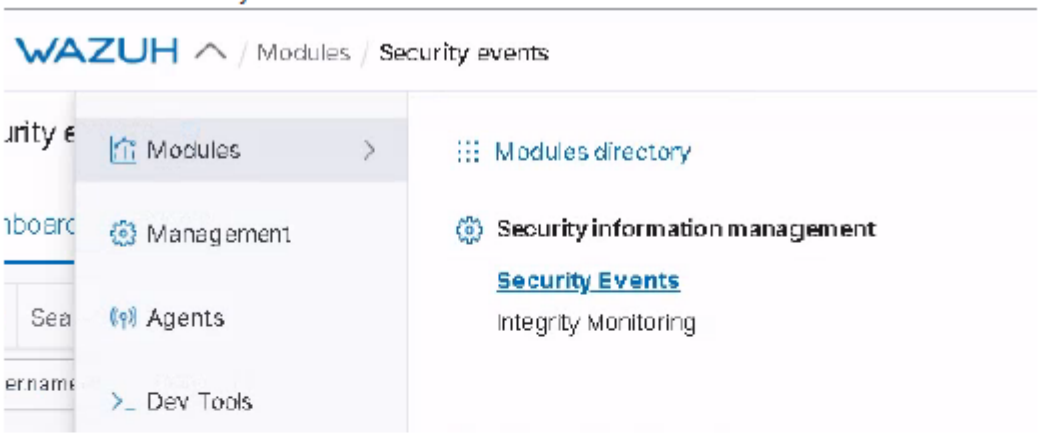



```
<group name="windows, sysmon, sysmon_process-anomalies,">
   <rule id="100000" level="12">
   <if_group>sysmon_event1</if_group>
   <field name="win.eventdata.image">mimikatz.exe</field>
   <description>Sysmon - Suspicious Process - mimikatz.exe</description>
   </rule>

   <rule id="100001" level="12">
   <if_group>sysmon_event8</if_group>
   <field name="win.eventdata.sourceImage">mimikatz.exe</field>
   <description>Sysmon - Suspicious Process mimikatz.exe created a remote
thread</description>
   </rule>

   <rule id="100002" level="12">
   <if_group>sysmon_event_10</if_group>
   <field name="win.eventdata.sourceImage">mimikatz.exe</field>
   <description>Sysmon - Suspicious Process mimikatz.exe accessed
$(win.eventdata.targetImage)</description>
   </rule>
</group>
```

7. Save the rules file and restart the manager with the button in the top right corner. Wait until the manager is back online. Go back to the rule search bar and check the dashboard if it shows the following (Click Manage rules files to switch between file and content view):



8. Go back to the Security Events view and execute mimikatz via PowerShell:



Refresh the page and you should now see the mimikatz log:

Learning Targets

1. Knowledge of how to configure Sysmon and detecting mimikatz
2. Know-How of how to edit rules configuration files of Wazuh

Additional Resources

- Download Sysmon
- Sysmon Configuration
- TrustedSec Sysmon Community Guide

A.4 Übungsszenario 2 Teil 2

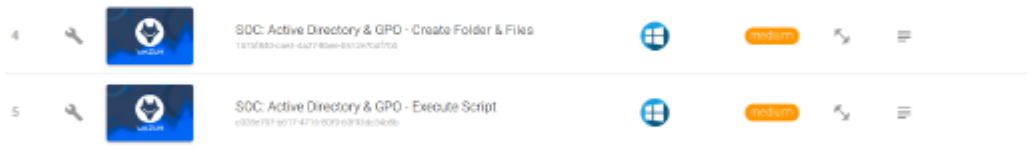
The main targets in this exercise are to create a log-forwarder to Wazuh via Group Policy and learn how to create alerts in Wazuh. The forwarded logs will be from Sysmon. The student has to install Sysmon and a Wazuh agent via GPO on the DC and all servers, which don't have it installed it yet. The alert in Wazuh will be an incident caused by registered mimikatz activity.

Prerequisites

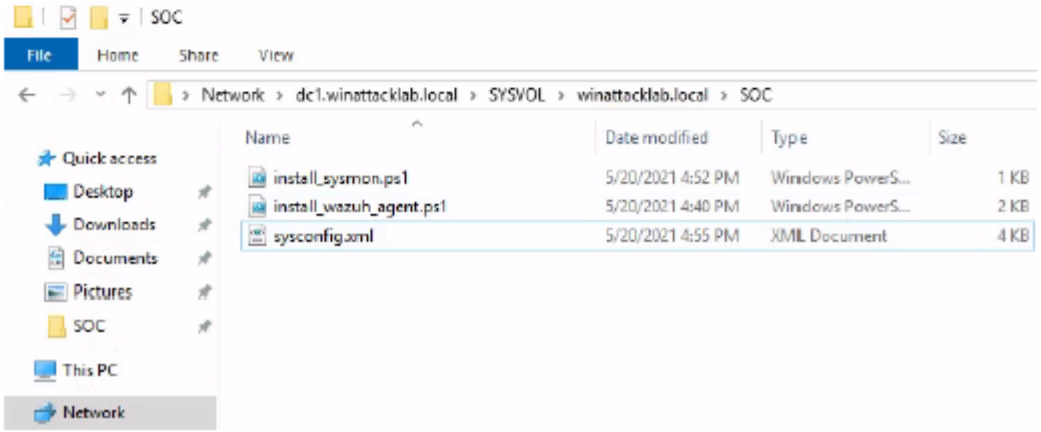
- WinAttackLab is running
  - Wazuh is ready to use (https://wazuh.winattacklab.local)
- Student/User can access the Windows 10 Client via RDP and log-in with AD credentials.
- Scripts/Files:
  - install_sysmon.ps1
  - install_wazuh_agent.ps1
  - sysconfig.xml (the same as the solutionsysconfig.xml file from exercise 4)

Tasks

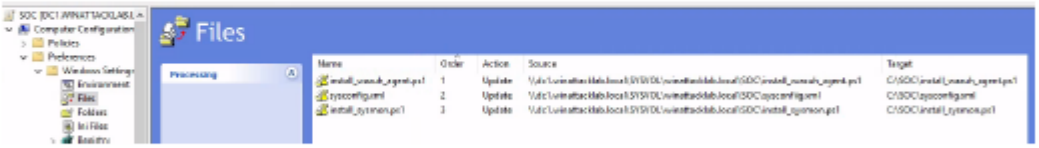
1. The first step includes using GPO to distribute the Wazuh agent and Sysmon onto all servers and the DC (Domain Controller) in the AD (Active Directory). Make sure that you have solved the exercises 3-5 where you are introduced to the basics of GPO and AD.
   1. Log-in to the Domain Controller using the AD lab_admin user
   2. If the created Group Policy of the exercises 4-5 does not exist on your deployment, please make sure to create it following the steps from those exercises.



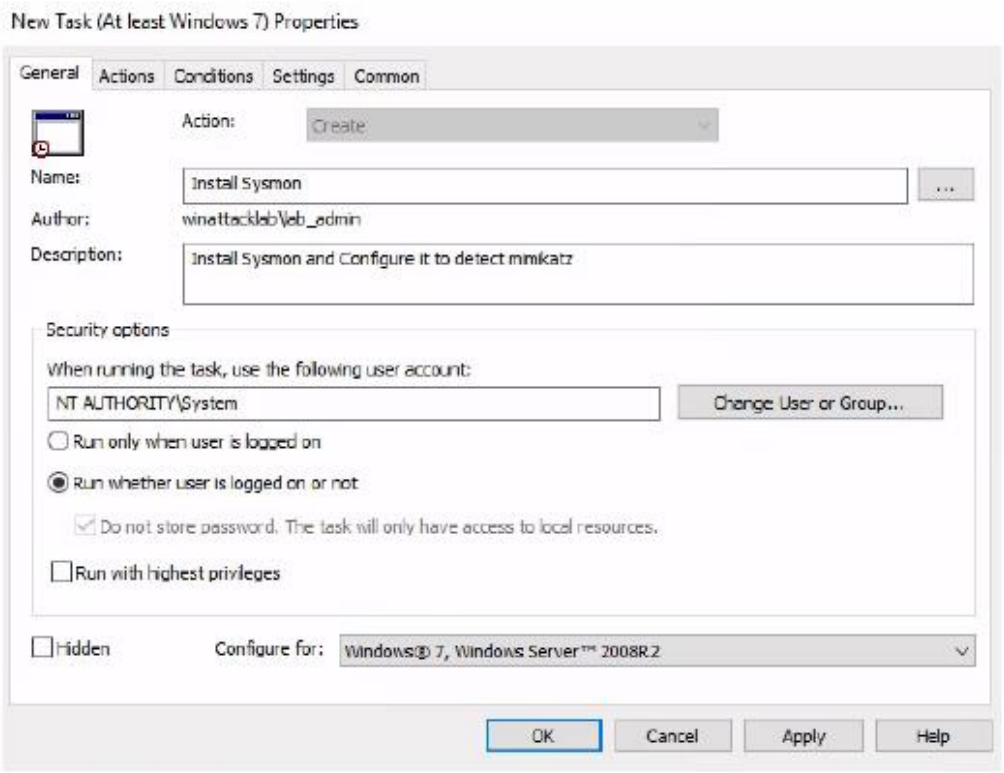
   3. At this step you should already have the path `\\dc1.winattacklab.local\SYSVOL\winattacklab.local\SOC\` with the install_wazuh_agent.ps1 script, thanks to exercise 4-5. Additionally, please create the files `install_sysmon.ps1` and `sysconfig.xml`, which are provided in this exercise.

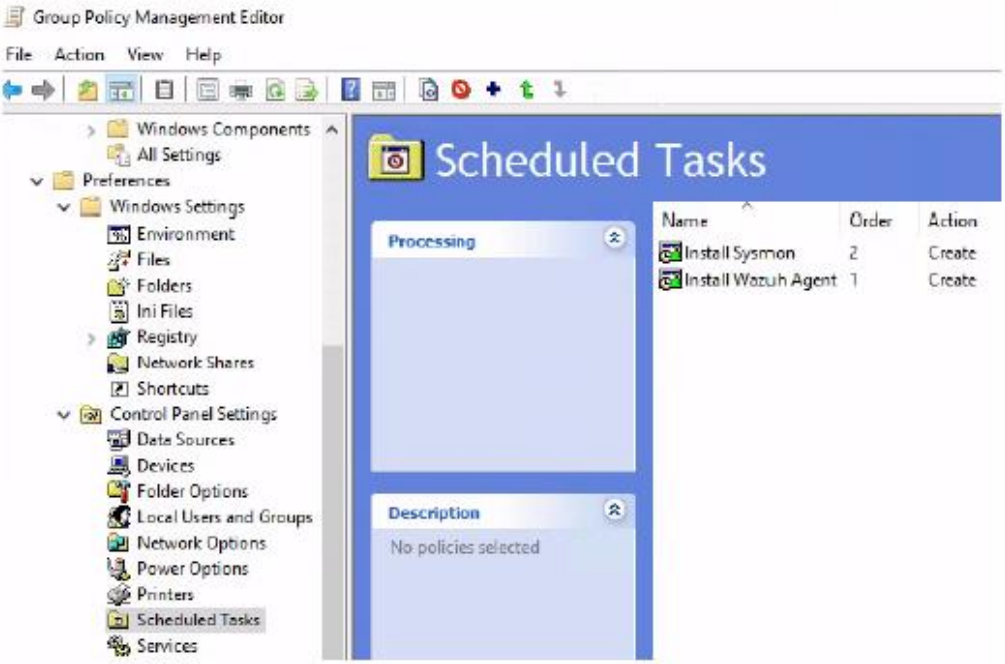


4. In the `Group Policy Management` please edit your SOC `Group Policy Object` and go to `Files` under `Computer Configuration --> Preferences --> Windows Settings`. Please also add an entry for `install_sysmon.ps1` and `sysconfig.xml`.
   - Important: Make sure that the sysmon.xml file is ordered prior to Sysmon PowerShell script file!
   - Important: Be sure to replace the install_wazuh_agent.ps1 script with the one provided in this exercise if you have one already saved under the SOC network path from the previous exercise.



5. Create new tasks in Scheduled Tasks under `Computer Configuration --> Preferences --> Control Panel Settings`. You possibly already have the install Wazuh agent script from exercise 5, please add a new task for Sysmon in the way described in exercise 5.





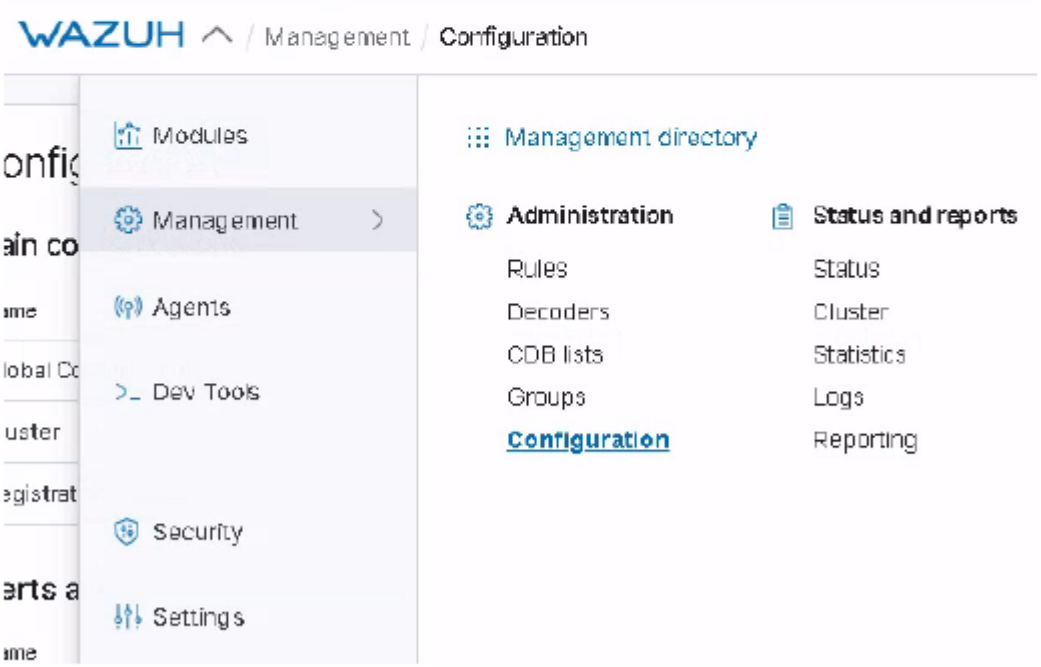
6. Close the `Group Policy Management Editor` and be sure to link the GPO to `Domain Controllers`, `Servers` and `ServersNoAV`.
7. Execute the following commands to force the policy onto the servers and the DC in the AD:

```
$computers = Get-ADComputer -Filter *
$computers | ForEach-Object -Process {Invoke-GPUpdate -Computer $_.name -
RandomDelayInMinutes 0 -Force}
```

8. Check in the Wazuh Dashboard on your Windows 10 Client if the agents where added and if Sysmon has been installed on the Domain Controller. While on the DC, also check if the Wazuh agent's ossec.conf file contains the necessary log analysis content to register Sysmon logs.

2. To receive all AD logs you need to adjust the ossec.conf file of the Wazuh manager as well. This can be done over the dashboard on the Windows 10 client.

1. First you need to configure the Wazuh manager ossec.conf file. Over the dashboard you can find it under `Management --> Configuration`



2. Click on `Edit configuration` in the top right corner and edit the line 11 and 13 to yes. Also set the SMTP server to 10.0.1.16 and the email alert level to 10. With these configurations you will be able to receive all AD logs and also use the Wazuh email alert feature. (Hint: If you want to

receive more than 12 emails an hours, elevate the number on line 17)

< Manager configuration

Edit **ossec.conf** of **Manager**

```
<!--
  Wazuh - Manager - Default configuration for centos 7.9
  More info at: https://documentation.wazuh.com
  Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>yes</logall>
    <logall_json>no</logall_json>
    <email_notification>yes</email_notification>
    <smtp_server>10.0.1.16</smtp_server>
    <email_from>ossecm@example.wazuh.com</email_from>
    <email_to>recipient@example.wazuh.com</email_to>
    <email_maxperhour>12</email_maxperhour>
    <email_log_source>alerts.log</email_log_source>
    <agents_disconnection_time>10m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
  </global>

  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>10</email_alert_level>
  </alerts>
```

3. Save and restart the Wazuh manager (top right buttons)
4. Don't forget to adjust the local_rules.xml file as described in exercise 7, if you have not done so in this deployment.

Rules (1)

From here you can manage your rules.

search: local_rules.xml × Filter or search

File	Actions
local_rules.xml	

```
<group name="windows, sysmon, sysmon_process-anomalies,">
   <rule id="100000" level="12">
   <if_group>sysmon_event1</if_group>
   <field name="win.eventdata.image">mimikatz.exe</field>
   <description>Sysmon - Suspicious Process - mimikatz.exe</description>
   </rule>

   <rule id="100001" level="12">
   <if_group>sysmon_event8</if_group>
   <field name="win.eventdata.sourceImage">mimikatz.exe</field>
   <description>Sysmon - Suspicious Process mimikatz.exe created a remote
thread</description>
   </rule>

   <rule id="100002" level="12">
   <if_group>sysmon_event_10</if_group>
   <field name="win.eventdata.sourceImage">mimikatz.exe</field>
   <description>Sysmon - Suspicious Process mimikatz.exe accessed
$(win.eventdata.targetImage)</description>
   </rule>
</group>
```

3. For the last step, please go back on the DC and install the hacker-tool mimikatz and execute it. Be sure to deactive the real-time protection in `Virus & threat protection settings`

    1. Install mimikatz:

```
$client = New-Object System.Net.WebClient
$client.DownloadFile("https://github.com/gentilkiwi/mimikatz/releases/downlo
ad/2.2.0-20210512/mimikatz_trunk.zip", "C:\mimikatz.zip")

Expand-Archive "C:\mimikatz.zip" -DestinationPath "C:\mimikatz"
Remove-Item -Path "C:\mimikatz.zip" -Recurse
```

    2. Execute mimikatz: `C:\mimikatz\x64\mimikatz.exe`
    3. In the dashboard you should find the mimikatz incident under Security Events now. If you are having trouble finding the event, remember to use the filter. For example `data.win = mimikatz`
    4. To view the sent email alerts, please go to https://mailcatcher.winattacklab.local. Since the email alert level was changed to 10, we can also trigger an email alert via using the SSH brute force attack from https://attack-launcher.winattacklab.local

Learning Targets

1. Be able to create a log-forwarder over GPO
2. Know-How of how to configure the Wazuh manager ossec.conf file

A.5 Übungsszenario 3

This exercise intends to show, how to extend the CDB (constant database) list in Wazuh and what kind of use it has. Specifically, we will extend Wazuh with some blacklisted IP addresses and use a simulated request upon some of those addresses. Related to this we will configure Suricata, a network intrusion detection system (NIDS), in Wazuh to show how it is done and possibly find malicious activity in return.

The source file used will contain blacklisted IPs, which will be extracted. It will be downloaded from https://urlhaus.abuse.ch/downloads/csv_recent onto the Ubuntu Server in one of the steps, where Wazuh is hosted.

Prerequisites

- WinAttackLab is running
  - Wazuh is ready to use (https://wazuh.winattacklab.local)
- Student/User can access the Windows 10 Client via RDP and log-in with lab_admin credentials.
- Student/User can access the Ubuntu Server with SSH key-exchange via the Kali-Linux jump host -> SSH private key and public Kali-Linux IP are available.

Tasks

1. In this first step we will check where CDB lists can be overviewed in the Wazuh manager.
   1. Under `Management --> CDB lists` you will find a couple of lists already. Take a look at some of them and their structure and also read the Wazuh documentation concerning CDB lists https://documentation.wazuh.com/current/user-manual/ruleset/cdb-list.html.

   2. In general, these lists can be edited here, but it is much easier to import new lists directly on the Wazuh manager host. One reason is that the source files usually aren't in the required CDB format and need to be reformatted.

2. Having checked the CDB list location in the dashboard, the next step is to SSH onto the Ubuntu Server and download the new list to be used.
   1. To connect to the Ubuntu server, we have to use SSH via a jump host, since it has no public IP. Please connect first to the Kali-Linux host and then to the Ubuntu host using SSH key-exchange for authentication. The following command can be used to do so (set private_key_path and kali_public_ip):

```
ssh -i $private_key_path -o ProxyCommand="ssh -i ./modules/kali1-
client/kali_private_key -W %h:%p LAB_ADMIN@$kali_public_ip"
LAB_ADMIN@10.0.1.16
```

2. On the Ubuntu server, elevate your user rights to `root` and `cd` into the docker volumes directory, which contains the Wazuh CDB lists:
   `cd /var/lib/docker/volumes/wazuh-docker_ossec_etc/_data/lists`
3. We are downloading the IP-blacklist file (check the next step) from here `https://urlhaus.abuse.ch/downloads/csv_recent`. As you can see from the screenshot, this file contains a lot of useless data for our CDB and needs to be reformatted:

4. Using the following command we are extracting unique IP-addresses only and appending a `:` to each line:
   `wget -O- https://urlhaus.abuse.ch/downloads/csv_recent/ | grep -v ^# | awk '{FS=","}{print $3}' | awk '{FS="/"}{print $3}' | awk '{match($0,/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/); ip = substr($0,RSTART,RLENGTH); print ip}' | awk 'NF > 0' | awk '{print $1":"}' | sort | uniq -d > abuse_blacklistip`

5. Don't disconnect yet!

3. Since we are going to make requests to malicious addresses in this lab, we will also include the logs of Suricata a NIDS (Network Detection Intrusion System).

   1. In general, the idea would be to add a `<localfile>` section to every agent and have a NIDS on each host with an agent. In this lab we will however only add it on the Ubuntu server.
   2. On the Ubuntu server: In `/var/ossec/etc` you will find the `ossec.conf` file of the Ubuntu agent. The NIDS Suricata is already installed and the logs can be found here: `/var/log/suricata/`

3. You can check the latest output of `eve.json` with `tail -n1 /var/log/suricata/eve.json | jq .` We are using the tool jq as a JSON beautifier.
4. While Suricata is already installed, Wazuh still needs a `<local_file>` entry in the Wazuh agent, for Wazuh to check the NIDS' logs. Please add the following entry in `/var/ossec/etc/ossec.conf`:

```
<localfile>
     <log_format>json</log_format>
     <location>/var/log/suricata/eve.json</location>
</localfile>
```

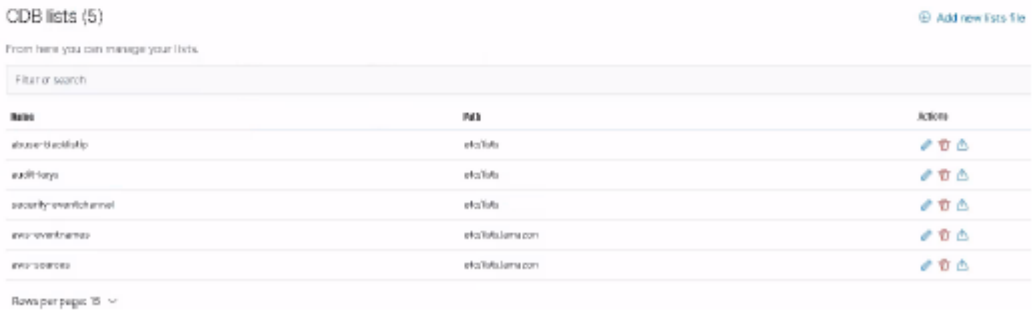
5. Restart the Wazuh agent with: `systemctl restart wazuh-agent`

4. Now, concerning the downloaded file `abuse_blacklistip`: Wazuh needs it to be in the `.cdb` format, which will be done in this step. The conversion will be done back on the Windows 10 client via the dashboard:

1. Back on the Windows 10 Client, please go to `Management --> Configuration` and scroll down to the default ruleset. Extend it with the new list path `<list>etc/lists/abuse-blacklistip</list>`:

```
302   <ruleset>
303     <!-- Default ruleset -->
304     <decoder_dir>ruleset/decoders</decoder_dir>
305     <rule_dir>ruleset/rules</rule_dir>
306     <rule_exclude>0215-policy_rules.xml</rule_exclude>
307     <list>etc/lists/audit-keys</list>
308     <list>etc/lists/amazon/aws-eventnames</list>
309     <list>etc/lists/security-eventchannel</list>
310     <list>etc/lists/abuse-blacklistip</list>
311
312     <!-- User-defined ruleset -->
313     <decoder_dir>etc/decoders</decoder_dir>
314     <rule_dir>etc/rules</rule_dir>
315   </ruleset>
```

2. Save and restart the Wazuh manager
3. Check if the CDB list was added in `Management --> CDB lists`:



5. Having added the new CDB list, we will now add a new local rule to compare possible requests being made to this list.

1. Go to `Management --> Rules` and extend the `local_rules.xml` file. Make sure that the rule id is unique. Below you can see the aditional rules, the top one being for the CDB list and the second one for Suricata:

```
<group name="attack,">
   <rule id="100100" level="10">
      <if_group>web|attack|attacks</if_group>
      <list field="srcip"
lookup="address_match_key">etc/lists/abuse_blacklistip</list>
      <description>Blacklisted IP was accessed</description>
   </rule>
</group>

<group name="curator">
   <rule id="100101" level="3">
      <decoded_as>json</decoded_as>
      <field name="@source">suricata</field>
      <description>Curator logs</description>
   </rule>
</group>
```

2. Save and restart the Wazuh manager.
3. After these configurations, you should now be able to see the alerts under `Security Events` if requests to the blacklisted ip addresses are being made. (Sadly at the end of BA Construction phase not working, since the proxy, which should redirect the requests, was not implemented)

Learning Targets

1. Understand the concept of CDB lists and be able to extend them in Wazuh
2. Know-how of SSH key-exchange proxy command

Security Questions

1. Why is it easier to create a new list on the Ubuntu server directly?
2. Explain the SSH jump host command. What do the parameters mean?
3. Explain the `awk` sections in the reformatting process of the blacklisted ip file

Additional Resources

- abuse.ch URLhaus
- IP set blocklists
- Using OSINT to create CDB lists and block malicious IPs
- Building a blacklist database (video)
- jq

A.6 Übungsszenario 4

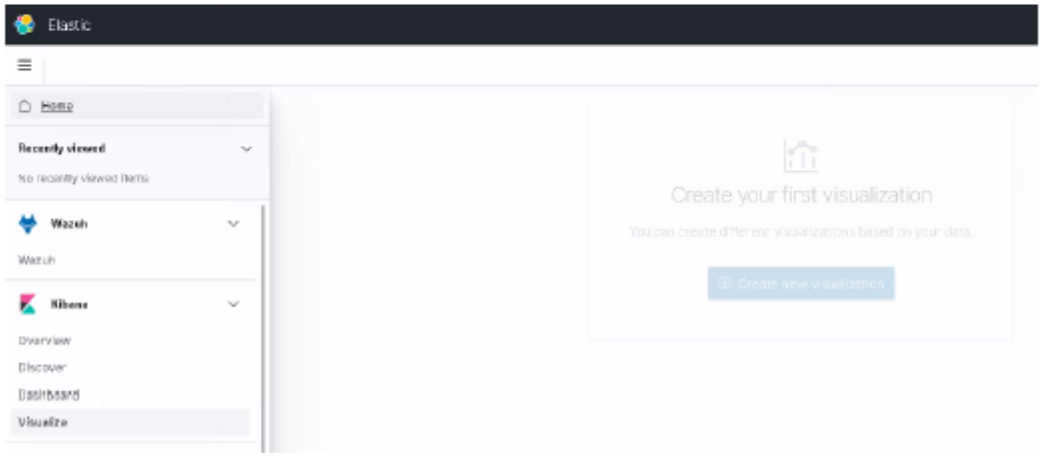
In this exercise the creation of a new dashboard in Wazuh is shown.

Prerequisites

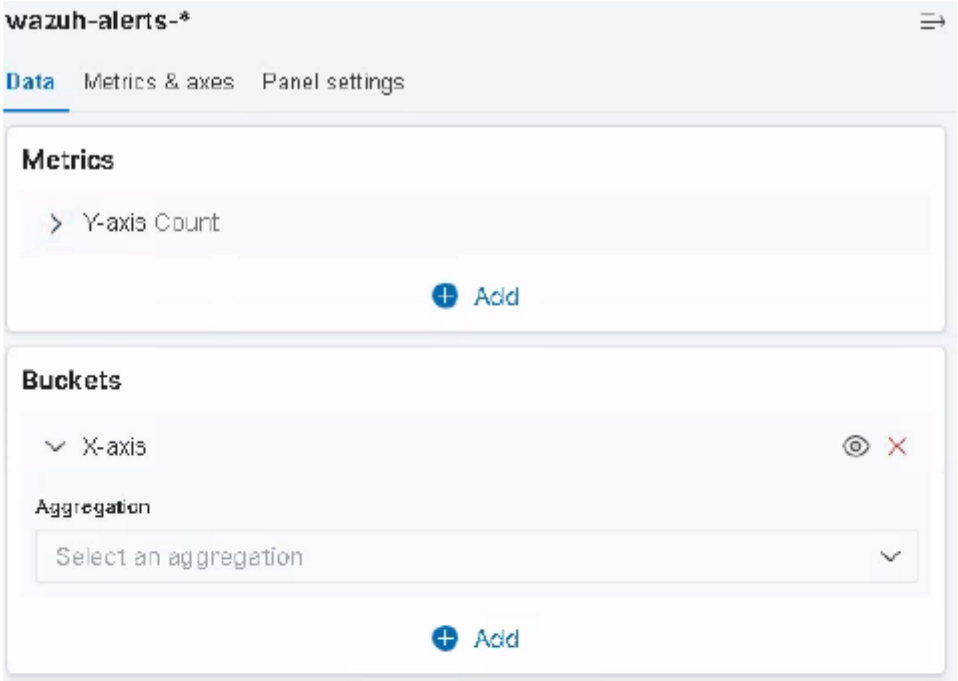
- WinAttackLab is running
  - Wazuh is ready to use (https://wazuh.winattacklab.local)
- Student/User can access the Windows 10 Client via RDP

Tasks

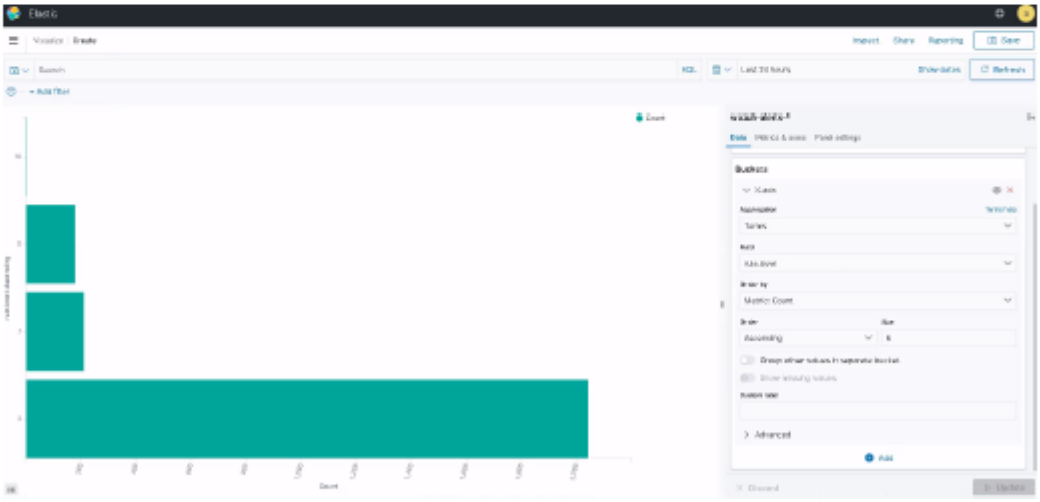
1. To create a new dashboard we first need to create some visualizations which might interest us and then add them to the new dashboard. All of this is being done in Kibana, which is actually what Wazuh is using for its dashboard visualization:
   1. In Wazuh please click on the menu icon in the top left corner and proceed to `Kibana --> Visualize` and create a new visualization:



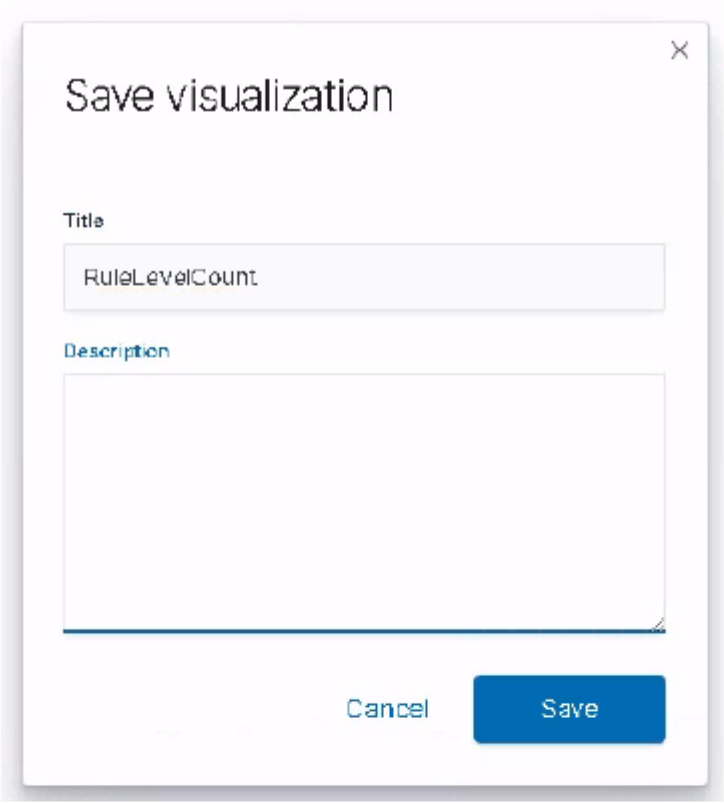
   2. Proceed with choosing `horizontal bars` and `wazuh-alerts-*` and add a new bucket (Buckets are aggregations of data that are sorted according to your search criteria):



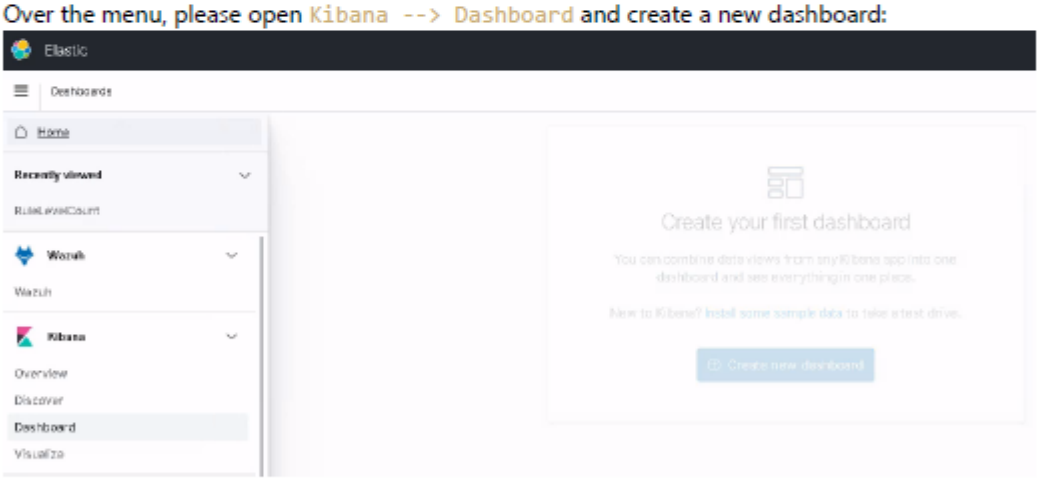
3. Select `Terms` in Aggregation, `rule.level` in Field and `Metric: Count` in Order by. Set Order to Ascending for better visualization as well and click on Update to view the result on the left. (Hint: If you are missing events with rule level 10, use the SSH-Bruteforce attack from the attack-launcher):



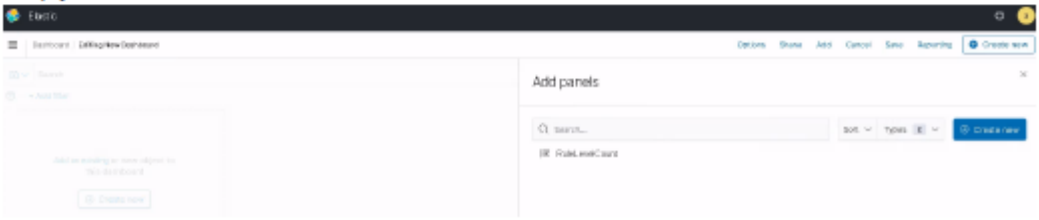
4. You can export some visualizations, this one included, to a CSV file. To do so, you simply need to click on `Inspect` in the top right corner.
5. To save the visualization for later use, click on the Save button in the top right corner, give it a title and save it:



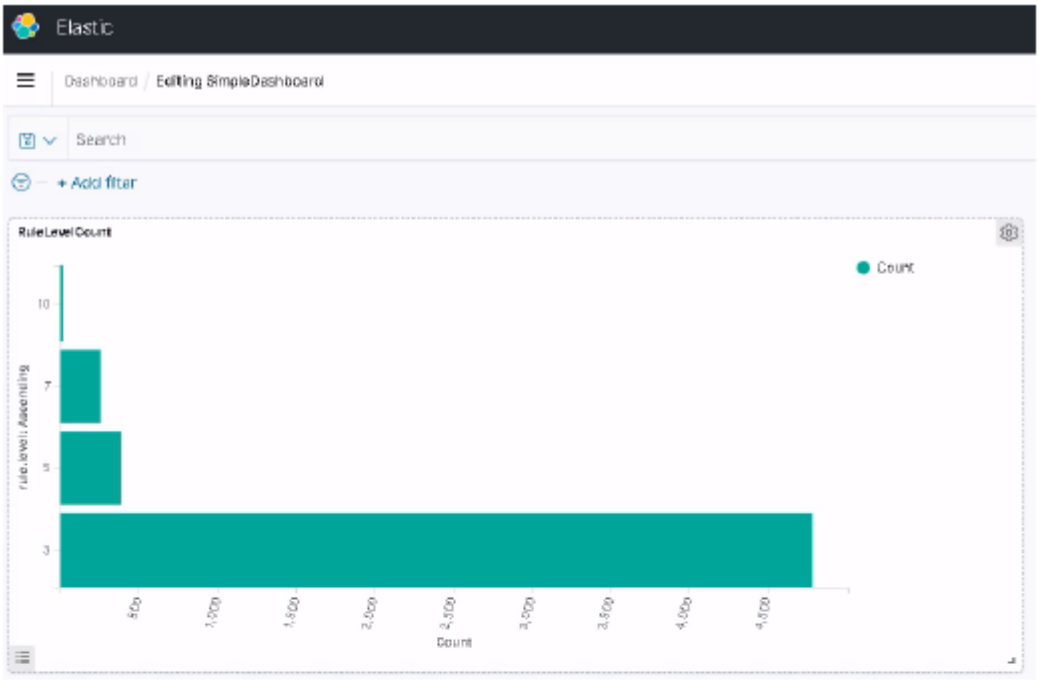
2. Having created a simple visualization, we now move on to the dashboard creation:
    1. Over the menu, please open `Kibana --> Dashboard` and create a new dashboard:



    2. Select `Add an existing` to add the previously created visualization. To add the visualization, simply click on it:

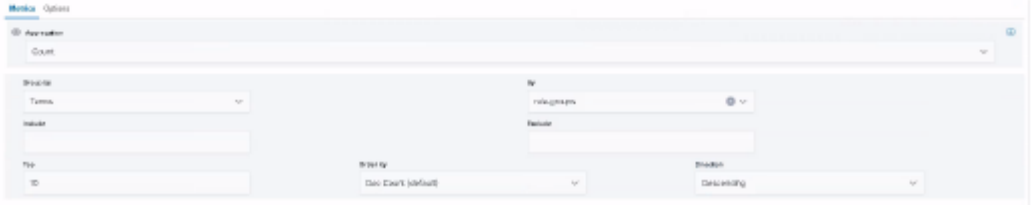


    3. You can save the dashboard by clicking the save button in the top right corner. Similar as with visualizations, give it a name and save it. Your dashboard should look similar to this:

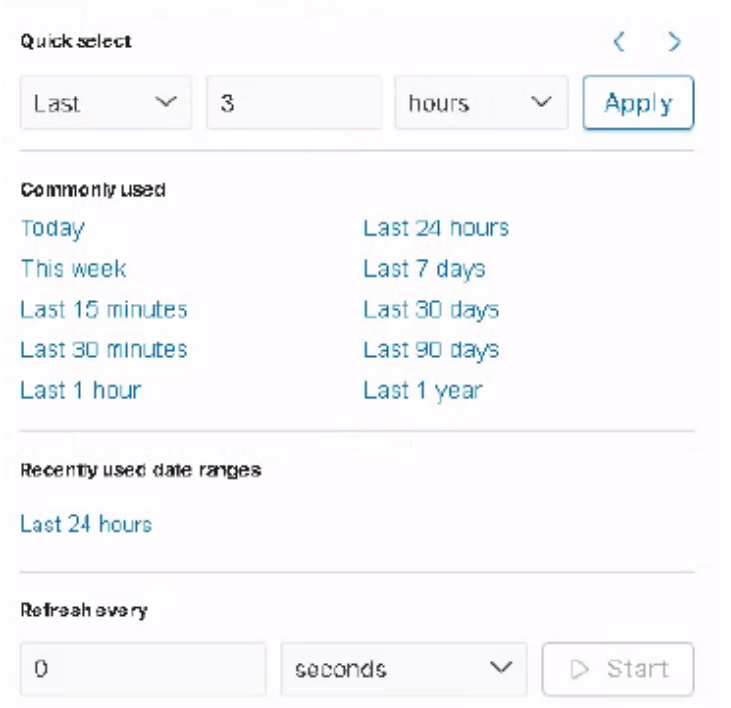


3. The current dashboard is very simplistic and not of much use, therefore we will add new visualizations:
    1. In your new dashboard, click on the button in the top right corner to create a new visualization and select TSVP (Time Series Visual Pipeline):

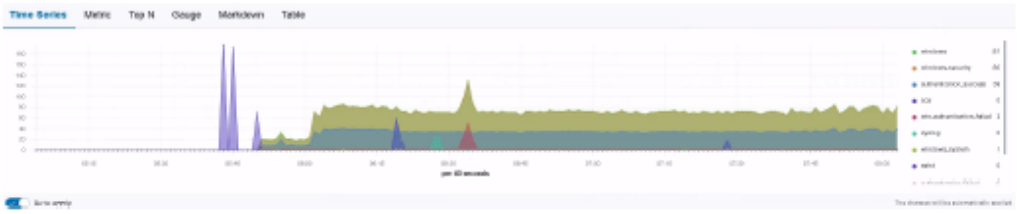
    2. You can change the color by clicking on the color template left to the Label input. However, this graph is not really helpful either. Scroll to the bottom and set Group by to `Terms` and By to `rule.groups`.



3. You now might look upon a squashed visualization. You can adjust the date range at the top. Simply click on calendar icon left to `Last 24 hours` and set the range to last 3 hours and apply it:



4. You should now have something similar to this, which visualizes the encountered amount of the different rule groups, which gives you an idea where most alerts are being registered from. Save your TSVP graph:

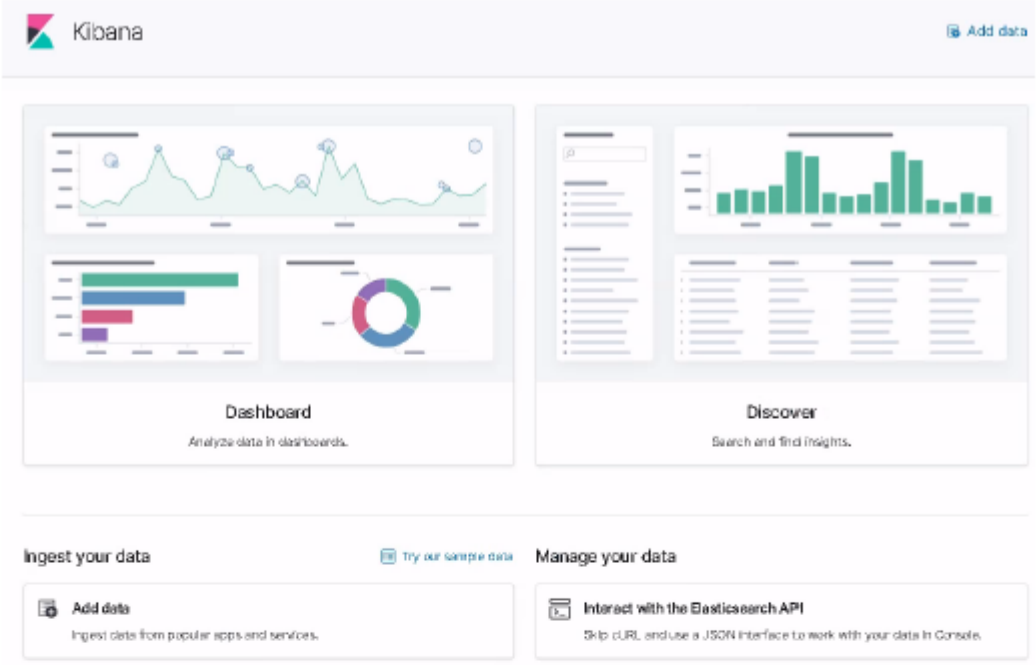


4. Back in your dashboard, please create at least one new visualization by your own design. Finally, you should export your dashboard as a report:

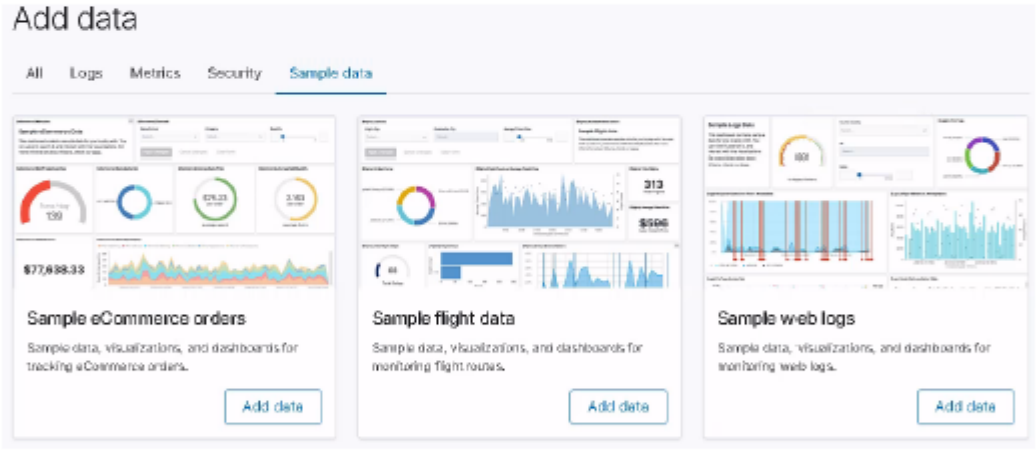
    1. Experiment a bit and create/add a new visualization to the dashboard. In your dashboard you can also drag and drop the visualizations and scale them.
    2. To create a report, you simply need to click `Reporting` in the top right corner. You can choose between generating a PDF or PNG file.
    3. Please upload your new dashboard as a PNG file to complete this exercise.

5. Optional step: To give you an idea of what a Wazuh dashboard creation is capable of, we will take a view at one of the provided dummy dashboards:

    1. Click on the menu icon in the top left and access `Kibana --> Overview` and on the bottom left click `Add data`:



2. Click on `Sample data` and then `Sample web logs` and wait a few seconds:



3. Over the menu go back to `Kibana --> Overview` and click on `Dashboard`. Select `[Logs] Web Traffic` and you'll find your mocked dashboard.

Learning Targets

1. Know how to create a new dashboard in Wazuh with newly defined visualizations
2. Know how to generate a report of the new dashboard

Additional Resources

- Edit visualizations in main dashboard
- Timelion visualization

B Terraform – Ubuntu-Server setup.sh Skript

```
#!/bin/bash

id > /tmp/mylog
sed -i "s/ubuntu/${hostname}/" /etc/hosts
export DEBIAN_FRONTEND="noninteractive"

# Create path for service applications and sample path
tooldir=/home/${admin_username}/tools
mkdir -p {$tooldir,/opt/applic/}
cd "$tooldir"

apt-get update -y

# Install basic necessaties like docker, pip, ca-certificates & add gpg
apt-get install curl apt-transport-https ca-certificates software-properties-common -y
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable"
apt-get update -y
apt-cache policy docker-ce
apt-get install docker-ce -y
curl -L "https://github.com/docker/compose/releases/download/1.28.6/docker-compose-$(uname -s)-$(uname -m)" \
-o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
apt-get install python3-pip -y
pip3 install python-pip -y

usermod -aG docker ${admin_username}
systemctl start docker
systemctl enable docker


# Wazuh https://documentation.wazuh.com/current/docker/wazuh-container.html
sysctl -w vm.max_map_count=262144


# Clone Wazuh, additional tools and SOC configuration repositories
cd /opt/applic/
git clone https://github.com/wazuh/wazuh-docker.git -b v4.1.5 --depth=1
git clone https://github.com/Hacking-Lab/SecurityOperationsCenter.git soc_config
git clone https://github.com/SecureAuthCorp/impacket.git

# Override Wazuh docker-compose.yml
cp /opt/applic/soc_config/soc/docker-compose.yml /opt/applic/wazuh-docker/docker-compose.yml


# Install smb to send self-signed certificate to Windows 10 client
apt-get install smbclient -y
cd /opt/applic/impacket/
pip3 install .


# Generate self-signed certificates
cd /opt/applic/soc_config/traefik/certs/
openssl req -newkey rsa:2048 -x509 -nodes -keyout cert.key -new -out cert.crt -subj \
/CN=*.winattacklab.local -reqexts SAN -extensions SAN -config <(cat /etc/ssl/openssl.cnf \
<(printf '[SAN]\nsubjectAltName=DNS:*.winattacklab.local, IP:10.0.1.16')) -sha256 -days 3650
# Reason for SAN extension:
# https://serverfault.com/questions/880804/can-not-get-rid-of-neterr-cert-common-name-invalid-error-in-chrome-with-self

# Start traefik
docker network create traefik_proxy
cd /opt/applic/soc_config/traefik/
docker-compose up -d

# Start Wazuh
```

```
cd /opt/applic/wazuh-docker/
docker-compose up -d

# Start attack launcher
cd /opt/applic/soc_config/attack-launcher/
docker-compose up -d

# Start mailcatcher
cd /opt/applic/soc_config/mailcatcher/
docker-compose up -d

# Install jq tool -> JSON parsing tool, used to beautify JSON output as well
apt-get update -y
apt-get install jq -y

# Install Suricata -> NIDS for Wazuh
mkdir -p /opt/applic/suricata_config
cd /opt/applic/suricata_config
apt-get install suricata -y
wget https://rules.emergingthreats.net/open/suricata-4.0/emerging.rules.tar.gz
tar zxvf emerging.rules.tar.gz
rm /etc/suricata/rules/* -f
mv rules/*.rules /etc/suricata/rules/
rm -f /etc/suricata/suricata.yaml
wget -O /etc/suricata/suricata.yaml http://www.branchnetconsulting.com/wazuh/suricata.yaml
systemctl daemon-reload
systemctl enable suricata
systemctl start suricata

# Add Suricata eve.json to Wazuh logs docker volume with softlink
mkdir /var/lib/docker/volumes/wazuh-docker_ossec_logs/_data/suricata/
# Softlink for the log files, with this the Wazuh manager can also be configured to check the
# Suricata logs via the dashboard ossec.conf file
ln -s /var/log/suricata/ /var/lib/docker/volumes/wazuh-docker_ossec_logs/_data/suricata/

# Restart Wazuh
cd /opt/applic/wazuh-docker && docker-compose restart

# To open the dashboard from the host over the jump host via ssh -X
apt-get install xdg-utils -y
apt-get install firefox -y

# Install Wazuh Agent on Ubuntu Server
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -
echo "deb https://packages.wazuh.com/4.x/apt/ stable main" | tee -a \
/etc/apt/sources.list.d/wazuh.list
apt-get update -y
apt-get install wazuh-agent -y

WAZUH_MANAGER="${ubuntu1_server_ip}" WAZUH_REGISTRATION_SERVER="${ubuntu1_server_ip}" \
apt-get install wazuh-agent -y

cd /var/ossec/etc/
sed -i "s/MANAGER_IP/${ubuntu1_server_ip}/" ossec.conf

systemctl enable wazuh-agent
systemctl start wazuh-agent


secs=3600
endTime=$(( $(date +%s) + secs ))

# Wait for Windows 10 client to be reachable -> max 1h
while [ $(date +%s) -lt $endTime ]; do
    nc -w 2 -z ${windows_client_ip} 139 2>/dev/null;
    if [ "$?" -eq 0 ]; then
        echo "Windows 10 Client is reachable!"
        break
    else
        echo "Windows 10 Client is unreachable.."
        sleep 15s
```

```
    fi
done
# https://stackoverflow.com/questions/11176284/time-condition-loop-in-shell

# Send self-signed cert to Windows 10 Client
cd /opt/applic/soc_config/traefik/certs/
smbclient -U "${admin_username}%${admin_password}" \
//${windows_client_ip}/c$ -c 'cd ./ ; put cert.crt'
```

C Usability-Test-Vorlagen

C.1 Usability Test #1

Usability Test #1: Übungsszenario 00

Laufnummer: 1-

Testdetails:

Datum:
Proband:
Funktion / Rolle:

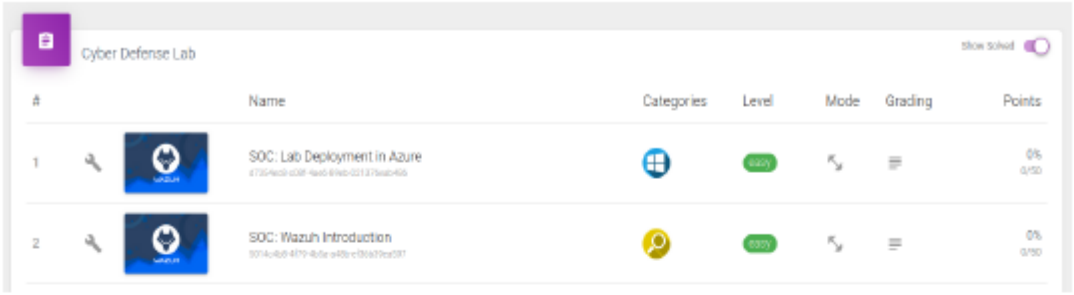
Einführung

Um den Usability Test durch zu führen, müssen Sie sich auf der Hacking-Lab Seite (https://ost.hacking-lab.com) zuerst registrieren, wenn Sie noch keinen Account haben.

Wenn Sie eingeloggt sind, dann können Sie unter Events den Token unter „REDEEM ACCESS TOKEN" einlösen. Der Token ist: **[für die publizierte Dokumentation entfernt]**

Danach sollten Sie den Event Security Operations Lab freigeschaltet bekommen. Klicken Sie darauf und gehen Sie weiter zu Cyber Defense Lab.

Aufgabenstellung:



Sie sind ein Student und haben sich für das Cyber Defense Modul eingeschrieben. Nun haben Sie die erste Vorlesung hinter sich und haben den Auftrag sich mit einem Security Operation Center (SOC) vertraut zu machen. Glücklicherweise wird die Wazuh, "Security Information and Event Management", kurz SIEM, Plattform über das Hacking Lab von Ivan Bütler zur Verfügung gestellt. Ihre Aufgabe ist, den Schritten auf dem Hacking Lab zu folgen und erste Eindrücke der Plattform zu bekommen. Lösen Sie die Aufgaben 1 und 2 im Cyber Defense Lab.

Auswertung:

Bitte folgende Punkte beantworten und die jeweilige Aufgaben Nummer angeben:

- Hat das Deployment mit dem Deployment Manager funktioniert?
- War es klar wie man auf die Windows 10 Maschine verbindet?
- War es klar wie man auf Wazuh verbindet?
- War das System performant?

- War die Aufgabenstellung verständlich?
- Haben die Schritte in der Aufgabenstellung funktioniert?

  Wenn nicht, bitte angeben, wo Probleme entstanden sind

- Haben Sie Schritte oder Beispiele vermisst?

  Wenn ja, bitte angeben wo genau

- Wie lange haben Sie für das Lösen der Aufgabe gebraucht?
- Beurteilen Sie den Lernerfolg dieser Übung

Was ist ihr generelles Feedback zur Aufgabe?

C.2 Usability Test #2

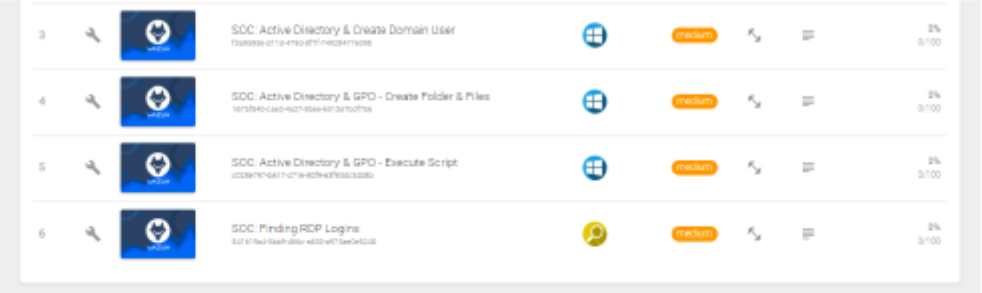
Usability Test #2: Übungsszenario 01

Laufnummer: 1-

Testdetails:

Datum:
Proband:
Funktion / Rolle:

Aufgabenstellung:



Nach der Einführungsübung zu Wazuh gibt es nun eine grössere Aufgabe. In dieser geht es vor allem um das Verständnis von Group Policy, Active Directory und Events. Sie sollen schlussendlich einen neuen Wazuh Agent über Group Policy kreieren können und danach RDP-Login-Events in Wazuh erblicken können. Folgt bitte den Schritten, die in den Aufgaben 3 bis 6 definiert sind.

Auswertung:

Bitte folgende Punkte beantworten und die jeweilige Aufgaben Nummer angeben:

- War es klar wie man auf den DC-Server verbindet?
- War das System performant?

- War die Aufgabenstellung verständlich?
- Haben die Schritte in der Aufgabenstellung funktioniert?

  Wenn nicht, bitte angeben, wo Probleme entstanden sind

- Haben Sie Schritte oder Beispiele vermisst?

  Wenn ja, bitte angeben wo genau

- Wie lange haben Sie für das Lösen der Aufgabe gebraucht?
- Beurteilen Sie den Lernerfolg dieser Übung

Was ist ihr generelles Feedback zur Aufgabe?

C.3 Usability Test #3

Usability Test #3: Übungsszenario 02

Laufnummer: 1-

Testdetails:

Datum:
Proband:
Funktion / Rolle:

Aufgabenstellung:

Die Aufgaben vom zweiten Übungsszenario befassen sich stark mit der Konfiguration vom Sysmon Monitoring-Tool und den Wazuh „rules“. Die Idee ist, ein Verständnis für diese Konfiguration zu entwickeln und wie man Sysmon und Wazuh konfigurieren muss, um das hacker-tool mimikatz zu entdecken. Folgen Sie den Schritten in der Aufgabe 7 und 8.

Auswertung:

Bitte folgende Punkte beantworten und die jeweilige Aufgaben Nummer angeben:

- War es klar wie man die Konfiguration in Sysmon machen kann?
- Was es klar wie man die Konfiguration in den Wazuh „rules“ machen kann?
- War das System performant? Wenn nicht, welches System war zu langsam?

- War die Aufgabenstellung verständlich?
- Haben die Schritte in der Aufgabenstellung funktioniert?

  Wenn nicht, bitte angeben, wo Probleme entstanden sind

- Haben Sie Schritte oder Beispiele vermisst?

  Wenn ja, bitte angeben wo genau

- Wie lange haben Sie für das Lösen der Aufgabe gebraucht?
- Beurteilen Sie den Lernerfolg dieser Übung

Was ist ihr generelles Feedback zur Aufgabe?