ins
INSTITUTE FOR
NETWORKED SOLUTIONS

OST
Ostschweizer
Fachhochschule

European Cyber Security Challenge 2022

Bachelorarbeit

Studiengang Informatik
OST – Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Frühjahrssemester 2022

Autoren:	Marc Emch
	Floris Staub
Betreuer:	Ivan Bütler
Experte:	Benjamin Fehrensen
Gegenleser:	Andreas Rinkel

Abstract

Mit der zunehmenden Digitalisierung weltweit wird Cyber Security Jahr für Jahr zu einem immer wichtigeren Thema. Und wie in vielen anderen Bereichen der Informatik herrscht ein grosser Mangel an kompetenten Fachkräften in diesem Bereich. Um dem entgegenzuwirken und junge Talente anzulocken, herauszufordern und zu belohnen, wird jährlich die European Cyber Security Challenge organisiert.

In einem Europa-übergreifenden Event kommen die Champion-Teams der einzelnen Nationen zusammen und liefern sich über 48 Stunden einen Wettkampf auf höchstem Niveau. Dabei gilt es, vorbereitete Hacking-Challenges zu lösen, um Punkte für das eigene Team zu sammeln. Vom Cracken von Passwörtern über das Reverse Engineering von Malware bis hin zum Hacken von Servern ist alles dabei.

In dieser Arbeit wurden mehrere solcher Challenges für die European Cyber Security Challenge 2022 erarbeitet. Dies beinhaltet alles vom Erstellen eines Konzepts über dessen Realisierung bis hin zum Testen der Challenges und dem Verfassen einer Musterlösung. Um einen möglichst hohen Lerneffekt zu erzielen, soll dabei nicht nur gewährleistet sein, dass die Challenges möglichst diverser Natur sind und verschiedene Technologien abdecken und aller Art Probleme der Cyber Security behandeln, sondern auch, dass die Fälle möglichst realitätsnah konzipiert sind und somit plausibel in der echten Welt angetroffen werden könnten.

Danksagung

An dieser Stelle möchten wir uns ganz herzlich beim Betreuer unserer Bachelorarbeit, Ivan Bütler, bedanken. Vielen Dank für die vielen Inputs, die Einsichten, die kritischen Nachfragen und vor allem für das Engagement.

Inhaltsverzeichnis

1. Management Summary

1.1 Aufgabenstellung

In dieser Arbeit geht es darum, Aufgaben für die European Cyber Security Challenge 2022, die im Herbst in Wien stadtfindet, zu erarbeiten. Die European Cyber Security Challenge ist ein jährlicher Event, der von der ENISA organisiert wird. In diesem Event geht es um die Nachwuchsförderung von Cyber Security Spezialisten. An dem Event treten die besten Teams aus verschiedenen europäischen Ländern gegeneinander an und messen sich in ihren Kenntnissen in allen Bereichen der Cyber Security. Im Verlauf dieser Arbeit sollen mehrere Aufgaben entwickelt werden, welche an diesem Event von den Teilnehmern gelöst werden können. Das Entwickeln einer solchen Aufgabe beinhaltet als erstes die Ideenfindung. Danach muss die Idee umgesetzt werden, und zum Schluss muss die Aufgabe getestet und dafür eine Musterlösung erstellt werden.

Nach Absprache mit dem Betreuer setzten wir uns als konkretes Ziel, insgesamt 8 Challenges zu entwickeln.

1.2 Vorgehen

Die erste Aufgabe, welche uns in dieser Arbeit gestellt wurde, war die Findung von Ideen für Challenges. Nach einem Meeting mit dem Betreuer und gemeinsamem Brainstorming, erarbeiteten wir etliche Grobkonzepte für Challenges. In einem weiteren Meeting priorisierten wir die Challenges und begannen damit, Feinkonzepte zu erarbeiten. Daraufhin begannen wir sehr bald mit der Implementation der ersten Challenge. Dies war jedoch ein iterativer Prozess, bei dem im Laufe der Entwicklung der Challenge mehrmals Teile des Konzepts angepasst oder überarbeitet werden mussten. Während der Entwicklung testeten wir die Challenged laufend auf unseren lokalen Systemen. Nach der Fertigstellung einer jeden Challenge, musste diese aber noch ins Hacking-Lab intergriert und dort getestet werden, und es musste für jede Challenge eine Musterlösung erarbeitet werden.

1.3 Ergebnis

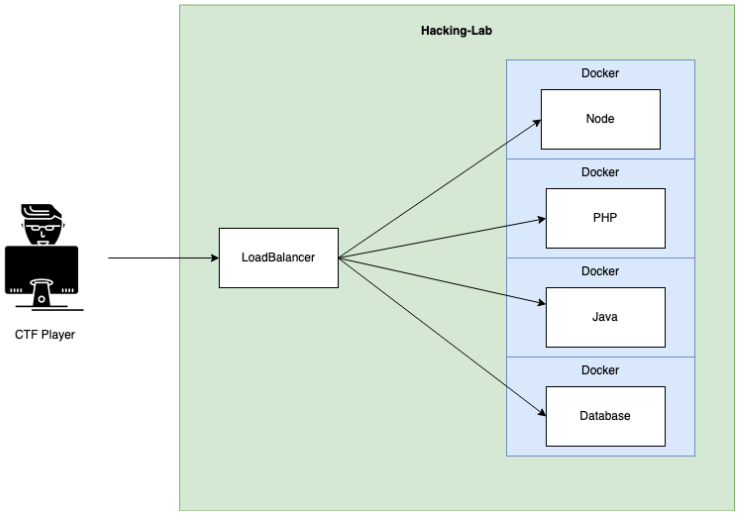

Abbildung 1 Ergebnis

Das Endprodukt dieser Arbeit umfasst 6 Challenges. Diese Challenges sind alle im Hacking-Lab deployed und wurden von Anfang bis Ende getestet. Weiterhin wurde eine Musterlösung für jede Challenge erstellt. Bei einigen Challenges umfasst dies ein Skript, welches die Challenge automatisch löst, bei anderen besteht diese aus Anweisungen, die manuell ausgeführt werden sollen.

2. Aufgabenstellung

2.1 Einführung

Im Herbst 2022 findet in Wien die European Cyber Security Challenge (ECSC) 2022 statt. Dies ist ein internationaler Hacking-Wettkampf, organisiert von der ENISA, an welchem 25 Länder mit je 10 Spielern teilnehmen, also insgesamt 250 Spieler.

2.2 Zielsetzung

Im Rahmen dieser Bachelorarbeit sollen verschiedene "geheime" Cyber Security Challenges entwickelt werden, welche für die CTF Player am ECSC zum Einsatz kommen. Mit "geheim" ist gemeint, dass der Inhalt und die Lösung der Aufgaben nicht veröffentlicht werden darf. Wer an der Entwicklung der Challenges beteiligt ist, darf ausserdem nicht mehr als Teilnehmer am Wettbewerb teilnehmen.

2.3 Themen

Die CTF Challenges können aus folgenden Kategorien sein

- Penetration Testing
- Forensics
- Incident Response
- Malware Analysis
- Reverse Engineering
- Web Hacking
- Mobile Security
- HW Challenges

Die Studierenden sollen sich zu Beginn der Arbeit Gedanken über mögliche Challenges machen und in Rücksprache mit dem Betreuer verschiedene Aufgaben umsetzen. Dabei ist auf eine thematische Aufteilung der Challenges zu achten, so dass nicht alle Aufgaben in die gleiche Kategorie fallen.

2.4 Ablauf

- Challenge Brainstorming
- Challenge Grobkonzept und Abstimmung mit Betreuer
- Challenge Feinkonzept und Umsetzung
- Challenge Testing und Quality Assurance

2.5 Level

Die Finalisten des ECSC sind geübt und versiert, CTF Challenges zu lösen. Es ist aber wichtig, dass die CTF Challenges einen echten Bezug zur IT Security haben, echte Schwachstellen enthalten und kein Lösen und Raten von Puzzles ist.

2.6 Hacking-Lab

Die Challenges müssen über das Hacking-Lab bereitgestellt werden. Zu jeder Challenge muss es eine Musterlösung geben, welche aufzeigt, wie man die Aufgabe löst.

2.7 Technologien

Für das Lösen von Challenges nutzen die CTF Player ihren eigenen Laptop (Linux, Windows, macOS). File-basierte Challenges müssen damit lösbar sein, ohne dass man kommerzielle Software beschaffen muss. Falls es einen Server-Teil zur Challenge gibt, so soll dieser über Docker Services realisiert.werden.

3. Projektdokumentation

Die Bachelorarbeit wurde zwischen dem 21.02.2022 und dem 17.06.2022 durch Marc Emch und Floris Staub durchgeführt.

3.1 Zeitplan

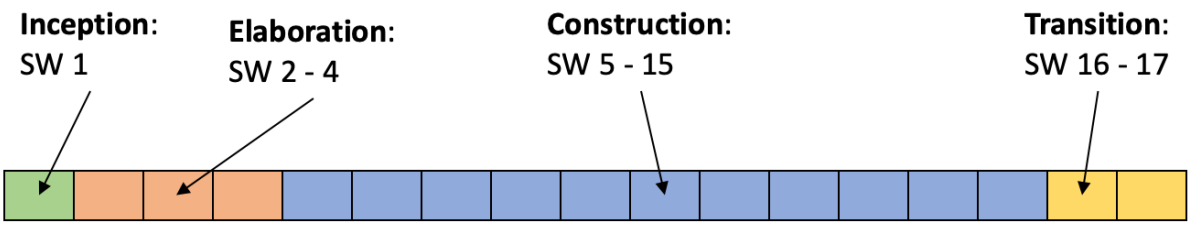

Abbildung 2 RUP Zeitplan

In der Zeitplanung haben wir uns grob an das RUP Modell gehalten. In der Inception-Phase haben wir uns mit dem Projekt vertraut gemacht. Die Elaborations-Phase wurde dazu genutzt eine Beispielapplikation für das Hacking-Lab zu erstellen. Zudem wurden in dieser Phase die Grobkonzepte für die Challenges entwickelt. Am Ende der Elaborations-Phase wurden die Challenges priorisiert nach Interesse an der Challenge. In der Construction-Phase wurden die Detailkonzepte der jeweiligen Challenges ausgearbeitet. Weiterhin wurden die Challenges implementiert. Die Transition Phase bestand hauptsächlich aus dem Deployment im Hacking-Lab, sowie der Erarbeitung der Dokumentation.

3.2 Soll-Ist-Zeitvergleich

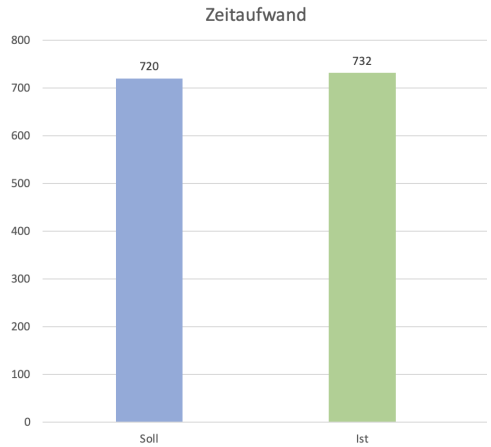

Abbildung 3 Sol-Ist-Zeitvergleich

Der Soll Zeitaufwand für die Bachelorarbeit berechnet sich wie folgt. Es sollen 30 Stunden pro Credit aufgewendet werden. Die Bachelorarbeit ergibt 12 Credits, dies resultiert in einem Soll-Zeitaufwand von 720 Stunden für beide Studenten. Wir haben insgesamt 732 Stunden für diese Bachelorarbeit aufgewendet und haben das Soll um 12 Stunden überschritten.

3.3 Rollen

In dieser Arbeit gab es nur zwei beteiligte Partien. Erstens Ivan Bütler, welcher als Auftraggeber und Betreuer der Arbeit agierte, und zweitens die Studierenden Marc Emch und Floris Staub, welche in diesem Projekt als Entwickler agierten. Das Ziel beider Parteien war es, eine Erfolgreiche Bachelorarbeit durchzuführen. Am Ende sollen Challenges herauskommen, welche für die ECSC2022 verwendet werden können.

3.4 Testing

Durch die Gegebenheit dieser Arbeit ist es nur möglich, Systemtests zu machen. In dieser Arbeit wurden viele kleinere Applikationen entwickelt, mit einer grossen Bandbreite an verschiedenen Technologien. Daher wurd auf Unit Tests verzichtet, da diese keinen entscheidenden Mehrwert gegenüber den Systemtests bieten. Getestet wurde manuell. Jede Challenge wurde auf ihre Funktionalität und Stabilität getestet. Zudem wurde die Lösung der Challenge getestet. Dafür wurden wo möglich Skripte geschrieben, welche das Lösen der Challenge automatisieren.

3.5 Risikoanalyse

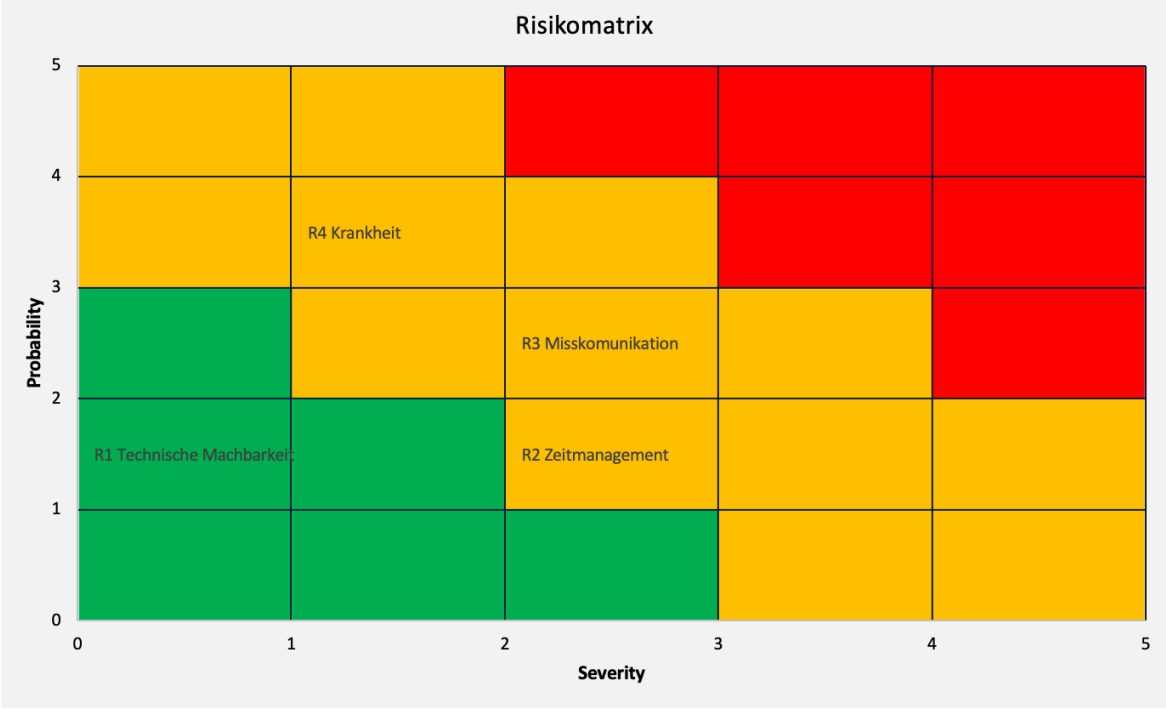

Abbildung 4 Risikomatrix

In dieser Arbeit haben wir vier grosse Risiken identifiziert

R1: Technische Machbarkeit

Das erste Risiko bezieht sich auf die technische Machbarkeit der Challanges. Es könnte vorkommen, dass eine Challenge nicht so realisiert werden kann, wie angedacht. Eine weitere Möglichkeit wäre auch, dass die Lösung einer Challenge technisch nicht machbar ist.

Um dieses Risiko zu reduzieren, haben wir uns vor jeder Challenge genau mit den Technologien, die verwendet werden sollen, auseinandergesetzt. Somit konnten wir gut abschätzen, ob die Challenges machbar sind oder nicht.

R2: Zeitmanagement

In diesem Risiko geht es darum, dass der Zeitplan nicht eingehalten werden kann aufgrund von unerwarteten Komplikationen. Wir haben dieses Risiko zu Beginn der Arbeit als eher tief eingeschätzt.

R3: Misskommunikation

Dieses Risiko behandelt das Problem der Kommunikation. Es kann vieles missverstanden oder falsch kommuniziert werde. Dies kann zu einem unerwünschten Ergebnis führen.

Dieses Risiko haben wir reduziert, indem wir uns regelmässig mit allen Beteiligten der Bachelorarbeit ausgetauscht haben. Zudem haben wir in jedem Meeting klare Ziele bis zum Nächsten Treffen definiert.

R4: Krankheit

Das Letzte Risiko, welches wir identifiziert haben, ist das krank werden einer oder beider Entwickler. Sollte dies eintreffen, kann es uns im Zeitplan zurückwerfen. Das kann darin resultieren, dass das Projekt nicht rechtzeitig abgeschlossen werden kann. Dies ist ein Risiko, welches leider nicht massgeblich verringert werden kann.

3.5.1 Eingetretene Risiken

Im Verlauf der Bachelorarbeit sind zwei der oben gezeigten Risiken eingetroffen. Die eingetroffenen Risiken sind R2: Zeitmanagement und R4: Krankheit. Das Risiko R2: Zeitmanagement ist eingetroffen, da wir mit einer einfach zu realisierenden Challenge begonnen haben und dadurch unseren Zeitplan zu optimistisch gestaltet haben. Dadurch haben wir uns zu wenig Zeit gelassen, Challenges zu realisieren und zu testen, welche viel aufwändiger waren als ursprünglich gedacht.
Das zweite Risiko, welches eingetroffen ist, ist R4: Krankheit. Über die Dauer des Projekts waren beide Studierenden teilweise krank, was einen Ausfall von mehreren Tagen zufolge hatte.
Diese beiden Umstände haben uns im Zeitplan zurückgeworfen, was darin resultierte, dass wir nicht den vollen geplanten Umfang der Arbeit realisieren konnten.

3.6 Fazit

Uns hat diese Arbeit sehr gefallen, und wir fanden es eine sehr lehrreiche Erfahrung. Wir kamen mit einer grossen Bandbreite an verschiedenen Technologien in Berührung, was es nicht immer einfach machte, sich zurecht zu finden und stetig vorwärts zu kommen. Dennoch haben wir letztendlich alle Hürden gemeistert.

Wir haben insgesamt 6 Challenges erstellt, und sind mit deren Qualität sehr zufrieden. Der einzige Problempunkt ist, dass es derer 6 sind, und nicht 8 wie ursprünglich geplant. Durch die vielen wechselnden Technologien und anfängliche leichte Erfolge haben wir uns im Zeitplan definitiv verschätzt. Daraus konnten wir lernen, und werden bei zukünftigen Projekten vorsichtiger mit dem Zeitplan umgehen.

4. Technischer Bericht

4.1 Anforderungen

Um einen erfolgreichen Ablauf des CTFs zu gewährleisten, müssen einige technische Anforderungen erfüllt werden. Da an der ECSC 25 Teams à je 10 Teilnehmern mitmachen, muss jede Challenge mindestens 250 Nutzer verkraften können, welche alle gleichzeitig versuchen, diese eine Challenge zu lösen. Zudem darf es nicht möglich sein, dass ein Team eine Challenge kaputt machen kann, und diese für andere Teams dann nicht mehr lösbar ist.

Inhaltlich soll ausserdem ein Bezug zur Realität gegeben sein, d. h. Schwachstellen oder Schwierigkeiten sollen möglichst so konzipiert sein, dass sie plausibel in echten Systemen angetroffen werden könnten. Des Weiteren darf ein wenig Ausprobieren ruhig gefordert werden, aber die Challenges sollen weder dem Zufall überlassen sein, noch ein Ratespiel darstellen. Konkret bedeutet dies, es soll für die Teilnehmer immer Anhaltspunkte geben, an denen sie abschätzen können, ob sie auf dem richtigen Weg sind oder nicht. Zudem soll der Schwierigkeitsgrad natürlich für die Teilnehmer angemessen sein - es handelt sich bei jedem Team um nationale Sieger. Dennoch müssen die Challenges alle relativ zeitnahe lösbar sein, da die ECSC nur 48 Stunden dauert.

Und zu guter Letzt soll für jede Challenge, wenn möglich, eine kleine Hintergrundgeschichte erfunden werden, welche sie auf irgendeiner Weise in das Thema "Klimawandel" einbettet. Dies muss aber nicht zwingend bei allen Challenges gegeben sein.

4.2 Docker

Um den technischen Anforderungen gerecht zu werden, wurden alle Challenges als Docker-Images implementiert. Dies erlaubt es jedem Teilnehmer, eine separate Instanz zu starten, sodass sich keine zwei Teilnehmer in den Weg kommen. Es erlaubt den Teilnehmern ebenfalls, die Challenges auf Wunsch zurückzusetzen, falls nach einem gescheiterten Lösungsversuch irgendwas im Docker nicht mehr funktioniert.

Diese Docker-Images werden den Teilnehmern über das Hacking-Lab[1] zur Verfügung gestellt. Per Knopfdrunk im Web-Interface kann jeder Docker einzeln gestartet werden und ist dann über eine dynamisch generierte URL erreichbar. Der Inhalt des Docker-Images bleibt den Teilnehmern dabei verborgen.

Die Docker-Images basieren auf Alpine Linux, wobei das Hacking-Lab mehrere vorgefertigte Basis-Images[2] zu Verfügung stellt. So gibt es z. B. vorgefertigte Images für Nginx mit PHP, einfach nur Nginx, oder bloss das Alpine Basis-Image ohne vorkonfigurierte Inhalte. Mithilfe des "hl-challenge"-Generators[3] kann eines dieser Images ausgewählt werden, und basierend darauf wird ein Docker-Grundgerüst

[1] https://hacking-lab.com/
[2] https://github.com/Hacking-Lab/alpine-base
[3] https://github.com/Hacking-Lab/generator-hl-challenge

erstellt, welches dann für die jeweilige Challenge angepasst werden kann. Für die Handhabung der Systemdienste innerhalb des Dockers wird das s6-Overlay verwendet, welches die Konfiguration und das Management von Prozessen mittels simplen Bash-Skripten erlaubt.

4.3 Dynamische Flags

Die "Flags" beim "Capture the Flag" sind in der Regel Zeichenketten der Form "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx", wobei die Ziffern 0-9 und Buchstaben a-f als Zeichen erlaubt sind (ein Flag könnte also z. B. " f65fb683-2435-51b0-31f0-d4696279bbfd" lauten). Diese Flags müssen von den Teilnehmern durch das Lösen von Challenges gefunden werden, und dann in einem Online-Formular eingegeben werden, wo sie automatisch geprüft werden.

Das Hacking-Lab bietet hierbei "dynamische Flags" an, welche die Eigenschaft haben, dass bei jedem Start eines Dockers ein neues Flag generiert wird. Damit soll unterbunden werden, dass Teams sich gegenseitig Lösungen zuschicken können.

4.4 Challenges

Die technischen Details der einzelnen Challenges sind vertraulich und sind deshalb in der veröffertlichten Version dieser Arbeit nicht enthalten. Eine Version mit allen technischen Details wurde dem Betreuer direkt abgegeben.

5. Glossar

Alpine Linux	Minimalistische Linux Distribution mit Fokus auf Geschwindigkeit und Kompaktheit.
CTF	Capture the Flag; Ein Wettkampf, in welchem mehrere Teams um das Erbeuten einer "Flagge" kämpfen. Oftmals auch für Hacker-Wettkämpfe verwendet, bei welchen durch das Lösen von "Challenges" Punkte für das eigene Team geholt werden können.
Deploy	Veröffentlichung einer Applikation auf einem Server
Docker	Eine Container-Lösung, um Applikationen voneinander akzukapseln und portabel zu machen.
Flag	Eine Zeichenkette, meist im UUID Format, welche als Lösung einer CTF-Challenge dient.

6. Verzeichnisse

6.1 Quellen und Literaturverzeichnis

Hacking-Lab (Stand 17.06.2022) https://hacking-lab.com/

Hacking-Lab alpine-base (Stand 17.06.2022) https://github.com/Hacking-Lab/alpine-base

Hacking-Lab Challenge Generator (Stand 17.06.2022) https://github.com/Hacking-Lab/generator-hl-challenge

6.2 Abbildungsverzeichnis