

# Hauptstudie

## Anomalie Detection in Netzwerken

Ives Schneider

## Index

1. Management Summary	1
2. Änderungen aus der Vorstudie	2
2.1. POC-LAN	2
3. Anomalie	2
3.1. Definition	2
3.2. Erkennung	2
3.3. Kill-Chain	3
3.4. Einstufung	7
3.5. Algorithmen	8
3.6. Baseline	11
4. Applikation	13
4.1. Technology	13
4.2. Architektur	14
5. Abhängigkeiten	15
5.1. Installation	15
5.2. Konfiguration	15
6. Controlling	17
7. Kosten	17
8. Weiteres Vorgehen	19
9. Freigabe	20
10. Darstellungsverzeichnis	21
11. Glossar	21
12. Anhang	22



## 1. Management Summary

Anhand der, in der Vorstudie evaluierten Softwarelösungen Splunk und PRTG, wird eine Softwarearchitektur definiert, welche im Weiteren Vorgehen zu entwickeln ist.

Die Applikation nimmt dabei Anspruch auf die bereitgestellten Funktionalitäten der beiden genannten Lösungen.

In der Hauptstudie soll definiert werden, mit welcher Architektur und mit welchen Fähigkeiten eine Applikation Ausgerüstet werden muss, um gewissen Anomalien erkennen zu können. Zusätzlich wurde mithilfe des MITRE ATT&CK Frameworks ermittelt, wie und welche Angriffe normalerweise durchgeführt werden.

### Features

Anhand der Architektur bestehen folgende kontinuierlichen Möglichkeiten, Anomalien zu erkennen:

- Erkennung neuer Hosts
- Veränderungen von Netzwerkservices
- Fehlgeschlagene Loginversuche
- Gezielte Überwachung von gewünschten Zielen

### Kosten

Da es sich um ein komplett Open-Source-Projekt handelt (minus Splunk+PRTG), entstehen ausser der aufgewendeten Arbeitszeit keine zusätzlichen Kosten.

### Projekt

Im Laufe der Hauptstudie wurde entschieden, die Testumgebung (POC-Lan) auf eine virtuelle Umgebung zu verschieben. Dies erspart zusätzlichen aufwand, die Hardwareumgebung nachträglich mit Splunk auszurüsten.

Innerhalb dieses Dokuments wird die Applikation fortlaufend mit ihrem Projektnamen (**Nidhogg**) bezeichnet.



## 2. Änderungen aus der Vorstudie

### 2.1. POC-LAN

Aufgrund eines Hardwarefailures muss leider auf die bestehende Infrastruktur verzichtet werden. Daher wurde eine virtuelle Infrastruktur mithilfe Ansible und Terraform erstellt.

Falls das POC-Netzwerk nachgebaut werden will, kann das Ansible Playbook und das Terraform Script [hier](#) gefunden werden.

## 3. Anomalie

### 3.1. Definition

Anomalien sind unerwartete Abweichungen von Regeln, im Kontext der Produktion also Abweichungen von "normalen Betriebszuständen". Diese treten meist in einem Fehlerfall auf. Sie können allerdings auch ein Hinweis auf einen Angriff bzw. eine Manipulation innerhalb eines Produktionsnetzwerkes sein. Das gilt insbesondere dann, wenn Ereignisse erstmalig auftreten, Prozesse sich anders verhalten oder Geräte miteinander kommunizieren, die es bisher nicht getan haben.

– BSI, [Monitoring und Anomalieerkennung in Produktionsnetzwerken](#)

### 3.2. Erkennung

Die Erkennung soll anhand eines Algorithmus erfolgen. Dabei soll der Algorithmus mehrere Merkmale analysieren. Grundsätzlich ist die Frage zu klären, wie eine standardmässige Kill Chain aussieht und mit welchen Massnahmen welche Schritte erkannt werden könnten.



### 3.3. Kill-Chain

Eine Kill-Chain beschreibt wie ein Angreifer normalerweise Zugang und Foothold in einem Netzwerk bekommt. Um einen klaren Überblick für die Möglichkeiten zu erhalten, wie eine Anomalie erkannt werden kann, wird auf das Mitre ATT&CK Framework zurückgegriffen, welche die einzelnen Phasen beschreibt.

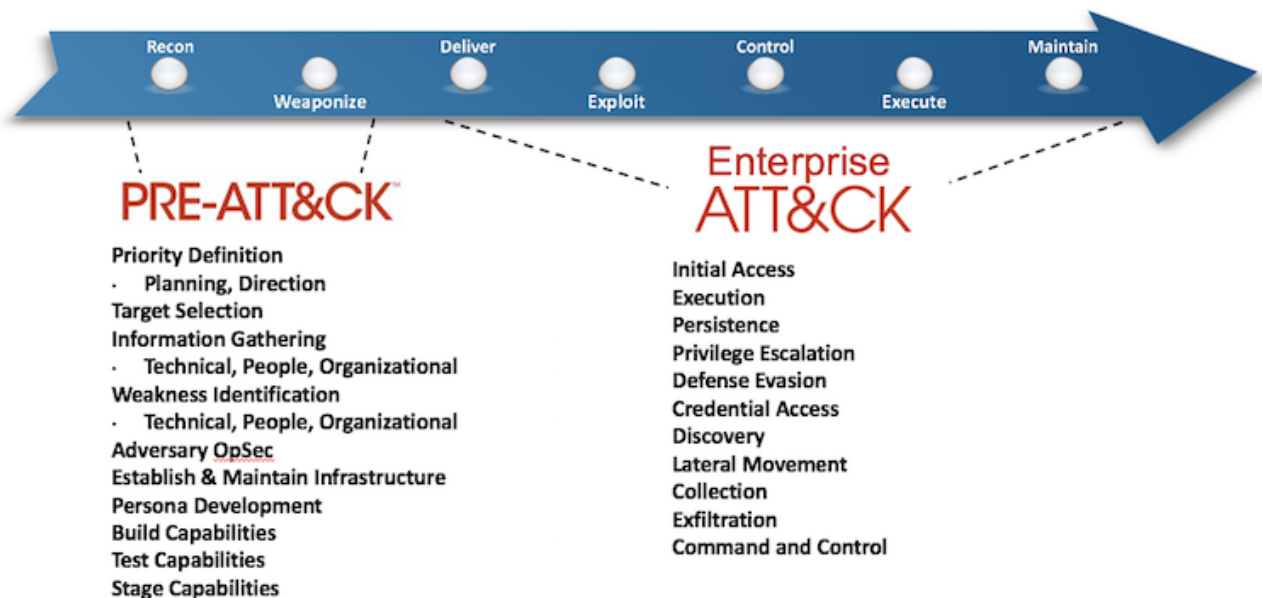


Figure 1. <https://attack.mitre.org/> - Killchain

#### 3.3.1. Recon

##### Beschreibung

In dieser Phase werden weitere Informationen über das Ziel beschafft.

Während dieser Phase befindet sich der Angreifer meist noch ausserhalb des zu angreifenden Netzwerkes. Allerdings ist Recon eine wiederholende Phase, welche während des gesamten Angriffs fortbeständig durchgeführt wird.

##### Erkennung

Solange sich der Angreifer ausserhalb des Netzwerkes befindet, gibt es nur eine limitierte Anzahl an Erkennungsmassnahmen.

Da der Scope von Nidhogg eher auf eine interne Erkennung liegt, wird dies auch nicht weiter verfolgt.

Intern hingegen, wird anhand eines passiven/aktiven ARP-Request Scannings das Netzwerk nach neuen Hosts durchsucht. Sollte ein neuer Host entdeckt werden, wird der Administrator via Mail auf die neue MAC Adresse Aufmerksam gemacht.



### 3.3.2. Weaponization

#### Beschreibung

Der Angreifer beschafft sich einen Exploit, welcher für die gefunden Schwachstelle in Phase I zugeschnitten ist. Die Art des Exploits spielt dabei keine Rolle.

#### Erkennung

Es gibt mehrere Möglichkeiten einen Exploit zu erkennen. Ein relativ neuer Ansatz ist mit sogenannten [YARA](#) Regeln.

Hierbei wird anhand bestimmter Regeln der Datenfluss analysiert und bei einer gewissen String-Abfolge entschieden, ob sich die Datei innerhalb der Known Malware Liste befindet.

Für Nidhogg wurde bewusst entschieden, auf eine zu intrusiven Netzwerkerkennung zu verzichten. Daher kann diese Phase nicht erkannt werden.

### 3.3.3. Delivery

#### Beschreibung

Der Exploit aus Phase II wird zum Ziel übermittelt. (Beispielsweise via E-Mail, Website etc.) Diese Phase ist einer der Keypunkte, einen Angriff erfolgreich zu verhindern.

Da die Phasen I und II sich ausserhalb des eigenen Netzwerkes befindet, besteht kein Kontakt mit dem Angreifer. Ab Phase III wird die Kommunikation mit mindestens einem Teil der Organisation aufgenommen.

#### Erkennung

Da die Kommunikation bewusst nicht überwacht wird, besteht hier keine Erkennungschance.

INPORTANT: Die Kommunikation und Host Überwachung wird auf Splunk + PRTG ausgelagert.



### 3.3.4. Exploitation

#### Beschreibung

Malware wird ausgeführt und folgt der eingebauten Logik ab.

#### Erkennung

Angriffe welche gezielt durchgeführt werden, setzen meist auf einen der folgende Vorgehensweisen:

1. E-Mail Attachement
2. Dropper
3. Download additional Malware
4. Foothold

1. Exploit
2. Reverse shell
3. Foothold

1. Exploit
2. Binding shell
3. Foothold

Nidhogg spezialisiert sich eher auf Erkennung und nicht auf Verhinderung.

Allerdings wird mithilfe des Portscanners würde eine Binding shell erkannt werden.

#### CAUTION

Dropper + Reverse Shell können leider nicht erkannt werden.



### 3.3.5. Installation (Control + Execute)

#### Beschreibung

Zusätzlicher Backdoor wird auf dem Zielsystem installiert (foothold).  
Dies ermöglicht dem Angreifer neue Verbindungen um weitere Kommandos der Malware zu senden.

#### Erkennung

Es gibt viele Möglichkeiten wie man ein Backdoor einrichten kann. Nidhogg soll die Möglichkeit haben, mindestens eine davon zu erkennen.

Table 1. Erkennung

Technik	Wird erkannt
Reverse Shell	-
Binding Shell	x
DNS backdoor	-
CnC Server	-

### 3.3.6. Command and Control (Control + Execute)

#### Beschreibung

CnC Server sendet Malware neue Instruktionen und ermöglicht dem Angreifer, Informationen aus dem Netzwerk zu ziehen.

#### Erkennung

Siehe Phase "Installation"

### 3.3.7. Actions on Objective (Maintain)

#### Beschreibung

Schritte zur Erfüllung des Ziels des Angreifers werden durchgeführt.  
Dies kann von Vernichtung von Daten beinhalten (selten) bis hin zu Datendiebstahl.

#### Erkennung

Grundsätzlich kann anhand der PRTG Auswertungen erkannt werden, ob zusätzliche Daten zu ungewöhnlichen Zeiten versendet werden.





### 3.4. Einstufung

Je nach Art des Alerts, soll eine gewisse Abfolge innerhalb von Nidhogg durchgeführt werden. Da es um eine Anomalie-Erkennung gehen wird, muss schlussendlich ein Administrator selbst entscheiden, wie schwerwiegend die gemeldete Informationen der Unternehmung schaden können.

#### 3.4.1. Event

Events sind normale Meldungen welche nicht auf schwerwiegende Anomalien hindeuten.  
z.Bsp. Hoher Netzwerkspike ohne zusätzliche Anomalien. Beispiel:

<b>NOTE</b>	High Bandwith: {IP}
-------------	---------------------

#### 3.4.2. Alert

Meldungen welche auf Downtime oder neue Geräte hinweisen. Allerdings ohne zusätzliche Informationen  
Beispiel:

<b>CAUTION</b>	New device found: {IP} {MAC}
----------------	------------------------------

#### 3.4.3. Incident

Anomalien welche auf lateral movement hinweisen könnten.  
Beispiel:

<b>WARNING</b>	New Port: {PORT} on {IP}
----------------	--------------------------



### 3.5. Algorithmen

Für die Anomalie Erkennung wird auf mehrere Algorithmen zurückgegriffen, welche im Hintergrund laufen sollen.

#### 3.5.1. New Host

Neue Hosts können auf einen Angreifer hindeuten, welcher einen freien Netzwerkport gefunden hat. Leider ist diese Variante für grössere Netzwerke, nicht einfach maintable, da die Anzahl Hosts massiv höher ist.

Nidhogg soll in der Lage sein, Administratoren über neue Hosts innerhalb eines definierten Netzwerkabschnitts zu informieren.

Somit kann die visibility massiv erhöht werden, welcher Administratoren einfacheren Durchblick über die Netzwerkinfrastruktur erbringt.

Da im POC Netzwerk die Hostanzahl begrenzt ist, wird auf eine PDO Verbindung verzichtet.

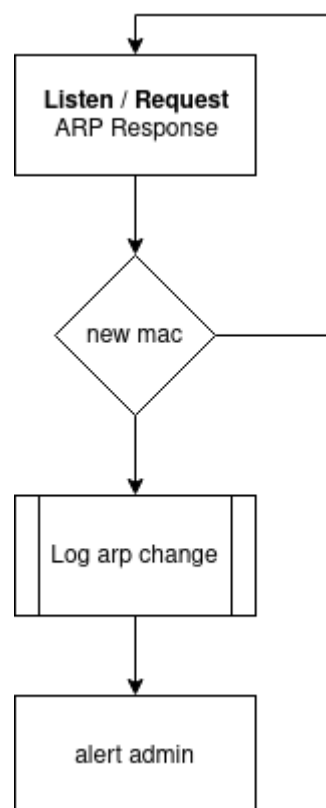


Figure 2. ARP-Change

#### IMPORTANT

Die Geschwindigkeit der Erkennung ist netzwerkgrössen + Last abhängig.



### 3.5.2. Portzustand

Durch die Konfiguration sollen offene Ports anhand einer Whitelist definiert werden.  
Falls sich der Status des Hosts ändern sollte, wird der Workflow ausgeführt.

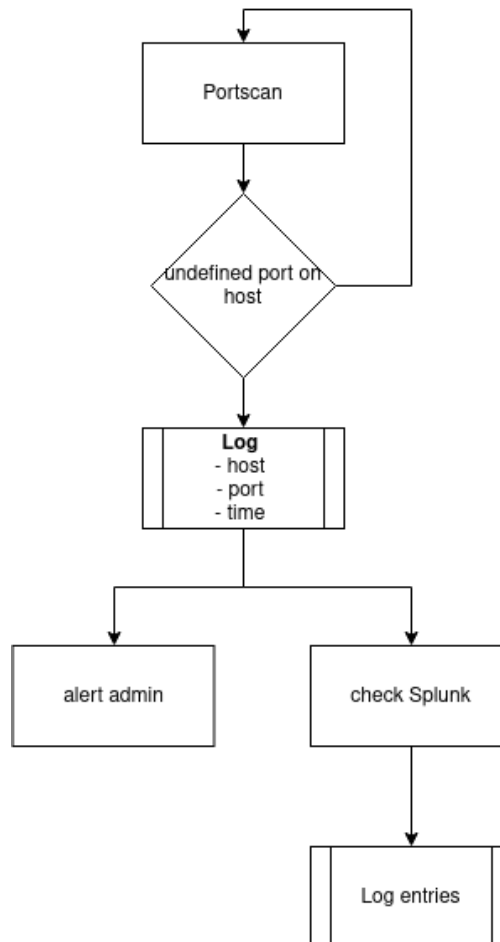


Figure 3. Änderung eines Portzustandes

### 3.5.3. Splunk failures

Da Splunk in der Vorstudie definiert wurde, soll Splunk via seinem HTTP-API angefragt werden können, ob bestimmte Änderungen geloggt wurden.

Die Checks sollen auf Reaktion eines anderen Alarms getätigt werden und somit die Benachrichtigungen verfeinern.



### 3.5.4. PRTG Meldung

PRTG besitzt die Möglichkeit, über sogenannte Notification Gruppen, Nachrichten an einen HTTP-Endpoint zu senden.

Nidhogg übernimmt hier eine passive Rolle und wartet auf calls an den REST-Endpoint.

Die ermöglicht es, nicht nur für PRTG verfügbar zu sein, sondern könnte auch via Icinga oder andere NMS angesprochen werden.

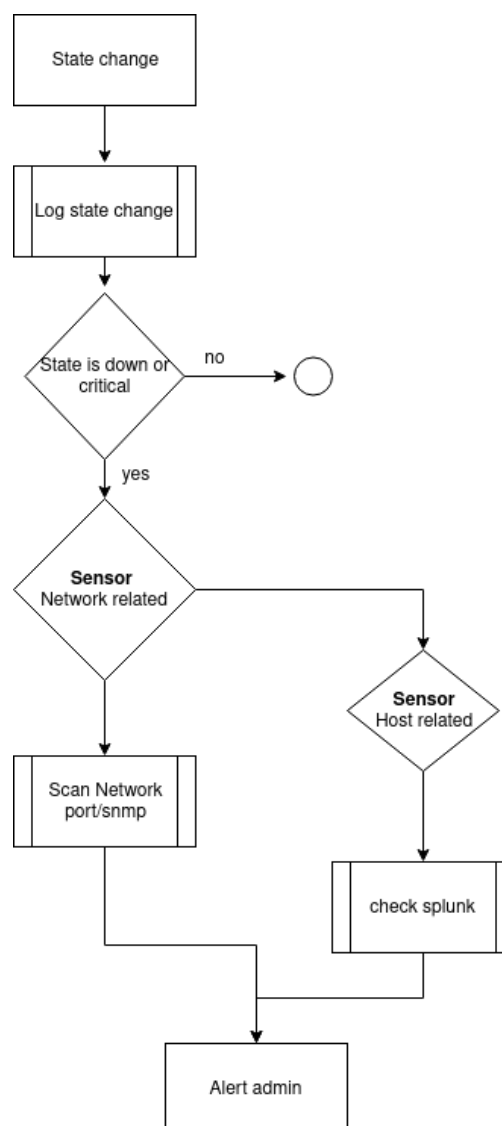


Figure 4. PRTG Meldungen



## 3.6. Baseline

### 3.6.1. Netzwerk

Das POC-Netzwerk wird bewusst etwas kleiner gehalten.  
Bestehend aus folgenden Services:

- PRTG
- Splunk
- NGINX (DVWA)
- Nidhogg
- Switch (Zyxel GS1910)

Wird eine kleine Infrastruktur simuliert, welche leicht anzugreifen ist.

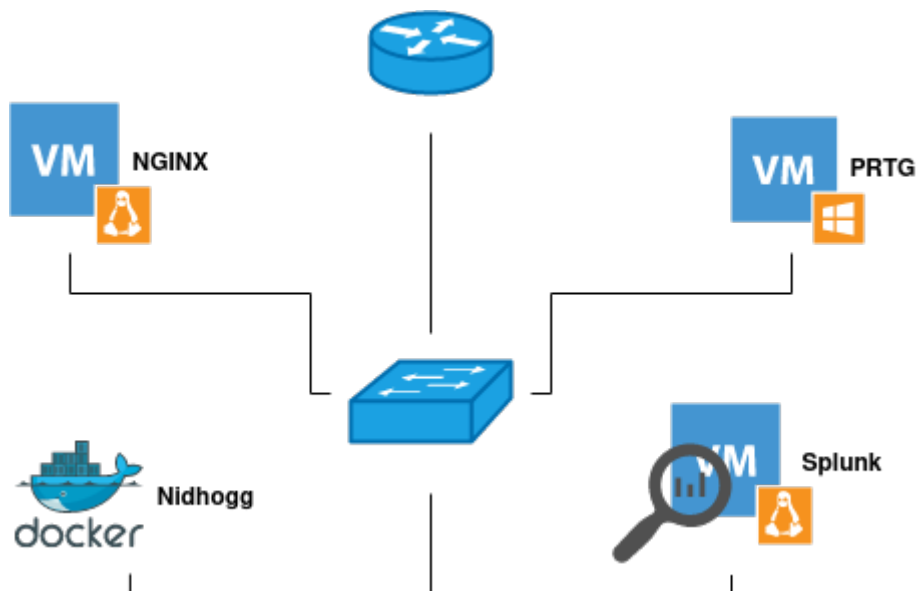


Figure 5. Netzwerkdigram

### 3.6.2. Traffic

Netzwerktraffik wird mithilfe [Nping](#) generiert.

### 3.6.3. Portmatrix

Ports

- 161 <> SNMP (Abfrage des Netzwerkstandes)
- 22 <> SSH (Monitoring)
- 8089 <> Splunk API



- 9997 <> Splunk forwarding

*Table 2. Portmatrix*

Host	PRTG	Splunk	NGINX	Nidhogg	Switch
PRTG	-	-	n/a	n/a	n/a
Splunk	22,9997	-	n/a	8089	n/a
NGINX	22,443	-	n/a	n/a	n/a
Nidhogg	22,443	n/a	n/a	-	161
Switch	161	n/a	n/a	n/a	-

Da ein grosser Teil der Überwachung auf Layer 2+3 durchgeführt wird, sind diese Teile in der Portmatrix nicht ersichtlich.



## 4. Applikation

Nidhogg soll als Anlaufstelle für Anomalie Erkennungen dienen.

Anhand diversen Merkmalen, soll erkannt werden, ob eine gemeldete Abweichung sich um eine Anomalie handelt welche genauer untersucht werden soll, oder aber um eine Abweichung, welche nicht weiterverfolgt werden muss.

Zugegriffen wird dabei auf folgende Möglichkeiten mit den Umsystemen zu kommunizieren.

### *Protokolle*

- ICMP
- SNMP
- HTTP
- ARP

Es wird versucht den Code möglichst low-level zu halten um die Performance der Umsysteme möglichs wenig zu beeinträchtigen.

### 4.1. Technology

Nidhogg wird in Rust geschrieben. Dies ermöglicht es, sicheren Quellcode zu schreiben ohne dabei Geschwindigkeit zu verlieren.

Von grossem belangen wird hierbei der Borrowchecker, lifetimes sowie das Secure Memory Management um Nidhogg möglichst erweiterbar und Ressourcen schonend zu schreiben.



## 4.2. Architektur

Nidhogg wird in einer einfachen 3-Tier Architektur entwickelt.

Die einzelnen Layers beziehen sich auf die Abschnitte des Programms.

Als Datenlayer wurde entschieden SQLite zu verwenden.

Natürlich kann argumentiert werden, dass lieber mysql/maria db etc. verwendet werden sollte.

Allerdings besteht bereits ein Programm, von welchem ich die Funktionalität wiederverwenden kann.

### First Tier

Webinterface

### Second Tier

Programm Logik

### Third Tier

SQLite

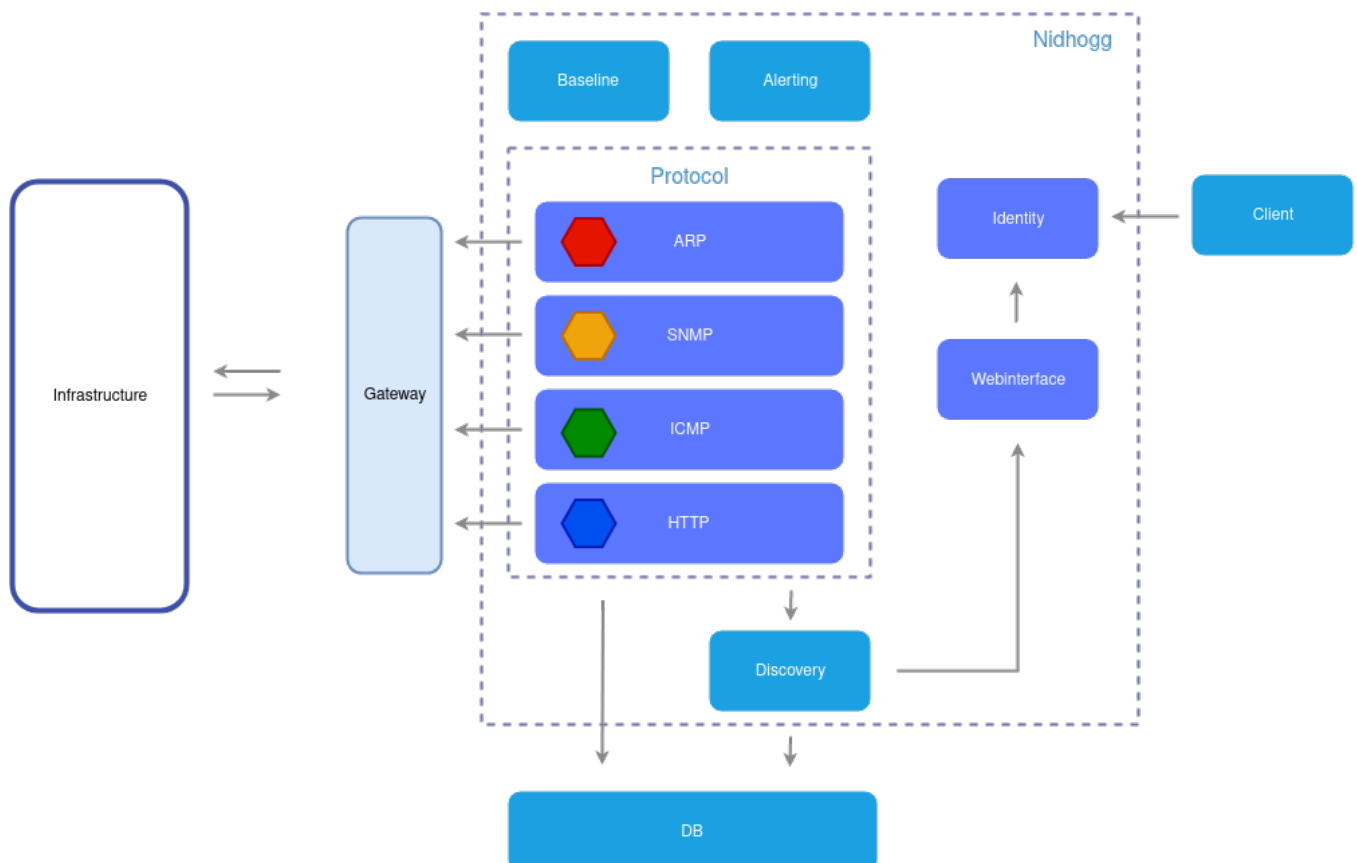


Figure 6. Nidhogg Architektur





## 5. Abhängigkeiten

Aufgrund des relativ jungen Alters von Rust, werden viele Libraries, welche in anderen Sprachen verfügbar wären, nicht existieren.

Dies bedeutet, dass einige Module selbst geschrieben werden müssen oder bereits bestehende Module auf die Funktionalität zugeschnitten werden müssen.

Allerdings kann bereits auf einige Abhängigkeiten eingegangen werden, welche benötigt werden:

Table 3. Abhängigkeiten

Abhängigkeit	Version	Grund	URL
Nmap	latest	Portscanning	<a href="#">nmap</a>
libpcap	latest	Arpscanning	<a href="#">libpcap</a>
serde	1.0.102	(de)serialization	<a href="#">serde</a>
actix_web	1.0.8	Webinterface/RESTful	<a href="#">actix</a>

### 5.1. Installation

Um die Installation möglichst einfach zu halten, soll Nidhogg einfach über CLI installiert werden können.

Unter Linux wird dies mithilfe des .deb Formates zustande gebracht.

Windows Systeme werden eine portable Binary erhalten.

Die komplette Installationsdokumentation wird in eigenem Dokument mitgeliefert.

### 5.2. Konfiguration

Einstellungen sollen via eines YAML Files vorgenommen werden können.

Um möglichst flexibel zu bleiben, sollen default Einstellungen mit Nidhogg mitinstalliert.

Wichtig ist es, dass einzelne Funktionalitäten deaktiviert werden können.

Ein erster Draft ist bereits verfügbar:



### config\_draft.yml

```
mail:
  server: "smtp.gmail.com"
  username: "user@gmail.com"
  password: ""
  email: "user@i-401.xyz"

splunk:
  server: "splunk:8089"
  username: ""
  password: ""
  interval: 500

snmp:
  server: "127.0.0.1"
  community: "my_comm"
  oid: "1.3.6.100.1.2.3.5.1.1.0"

portscan:
  portspec: "/etc/nidhogg/portspecs.yml"
  mappings: "/etc/nidhogg/mappings.xml"
  timeout: 500

arpscan:
  interface: "eth0"
  db: "/etc/nidhogg/arp.db"
  timeout: 500
  mac:
    - "00:17:88:28:9f:ca"
```



### Mappings\_draft.xml

Da ausserhalb der Hauptkonfiguration, ebenfalls noch Einstellungen für die wiederholende Portscan-Funktion gemacht werden muss, kommen zwei zusätzliche Konfigurationsdateien hinzu.

```
{ "mappings":  
  [  
    {  
      "hostname": "artoria",  
      "id": "i-0",  
      "ips": ["10.2.1.121"],  
      "name": "artoria",  
      "portspec": "artoria"  
    }  
  ]  
}
```

### porspec\_draft.yml

Die Portspec Datei wird dafür genutzt, Hosts aus dem Mappings.xml, mit den Einstellungen der Ports zu verbinden.

```
portspecs:  
  - name: artoria  
    ports:  
      - id: 22  
        state: open  
      - id: 25  
        state: closed
```

Der komplette Konfigurationsumfang wird in eigenem Dokument mitgeliefert.

## 6. Controlling

Die vollumfänglichkeit der gesetzten Ziele wird innerhalb dieses Kapitels getestet und angeschaut.

Tests werden innerhalb des neu definierten POC-Netzwerks in einer kontrollierten Umgebung durchgeführt.

Test und Controlling wird in einem zusätzlichen Dokument mitgeliefert.

## 7. Kosten

Die Kosten belaufen sich auf die aufgewendete Arbeitszeit.

Da die Entwicklung mit Open-Source Modulen in einer Sprache welche, unter MIT Lizenziert, entwickelt wurde, steht es frei die Applikation für nicht kommerzielle Zwecke zu verwenden.

Da das POC-Netzwerk relativ klein gehalten wurde, entstehen auch keine Kosten in Sachen Log-Collector bzw.



NMS.



## 8. Weiteres Vorgehen

Grundlegende Informationen sind soweit alle Vorhanden um mit der Erstellung von Nidhogg zu beginnen. Da es wesentlich mehr Faktoren benötigt, als anfangs angenommen, wird die Applikation in kleinere Pakete aufgeteilt und via Workspaces zusammengefasst.

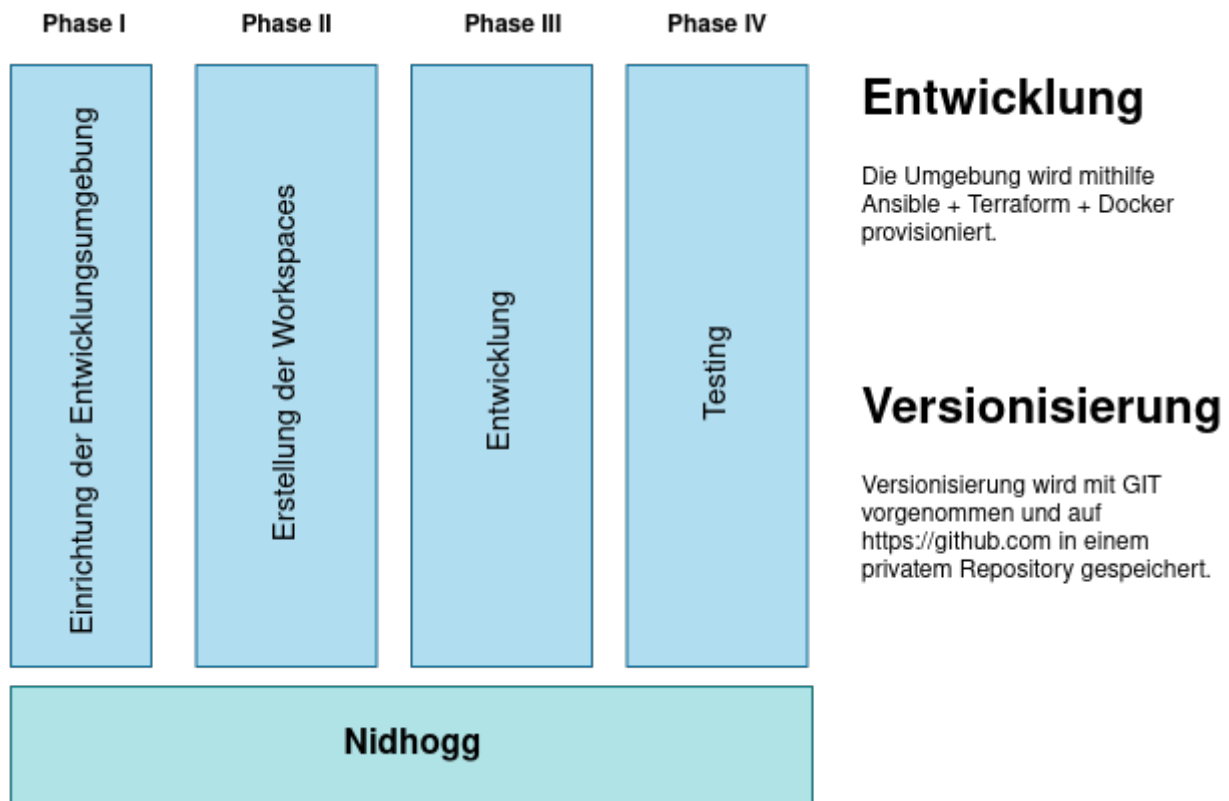


Figure 7. Weiteres Vorgehen



## 9. Freigabe

### Auftraggeber

Technische Berufsschule Zürich  
Sihlquai 101  
8090 Zürich  
[admin.hf@tbz.zh.ch](mailto:admin.hf@tbz.zh.ch)

### Projektleitung

Ives Schneider  
Binzstrasse 19  
8712 Stäfa  
[ives.schneider@i-401.xyz](mailto:ives.schneider@i-401.xyz)

### Experte

Marco Sieber  
[marco.sieber@tbz.ch](mailto:marco.sieber@tbz.ch)



## 10. Darstellungsverzeichnis

Figur. 1	Killchain
Figur. 2	ARP-Change
Figur. 3	Änderung eines Portzustandes
Figur. 4	PRTG Meldungen
Figur. 5	Netzwerkdiagramm
Figur. 6	Nidhogg Architektur
Figur. 7	Weiteres Vorgehen

## 11. Glossar

### **NMS**

Network Monitoring System - Überwachungssystem

### **POC**

Proof of concept - Konzeptbeweise

### **Kill-Chain**

Phasen eines Angriffs



## 12. Anhang



# Nidhogg

Installationsdokumentation

Ives Schneider



Index

- 1. Info. . . . . 1
- 2. Requirements . . . . . 1
- 3. Installation . . . . . 1
- 4. Configuration . . . . . 2
  - 4.1. Main configuration file . . . . . 2
  - 4.2. Portscan configuration files . . . . . 3
- 5. Uninstall . . . . . 4



## 1. Info

If you're planning to run nidhogg with its scanning capability, you won't get around to sadly run it as root. (or with cap rights)

This is due the nmap scanning parameter and arp scanning mechanism.

Ideally you're using the already created docker, to mitigate some of the security concerns, not all though.

## 2. Requirements

### Linux

- libpcap0.8
- nmap
- curl
- sqlite3-0

### WARNING

If you're running Ubuntu chances are high that you'll need to create a symlink for libpcap  

```
ln -s /usr/lib/x86_64-linux-gnu/libpcap.so.1.9.1 /usr/lib/x86_64-linux-gnu/libpcap.so.1
```

## 3. Installation

### From source

1. Clone repository or download source
2. Build nidhogg `cargo build --release`
3. Manually create config files (see example config.yml)
4. Copy service file to `/etc/systemd/system`
5. Activate Unit `systemctl enable nidhogg`

### Binary (Ubuntu)

1. Download latest .deb
2. Install with dpkg `dpkg -i xx.deb`
3. Configure application (`/etc/nidhogg/`)
4. Enable unit `systemd enable nidhogg`

### Binary (Windows)

Get the Docker image from:

<https://hub.docker.com/r/b401/nidhogg>

### Docker

<https://hub.docker.com/r/b401/nidhogg>



## 4. Configuration

### WARNING

All configflags are mandatory

See examples for more indepth settings.

### 4.1. Main configuration file

Defines most configuration aspects of nidhogg.

All config flags are mandatory but every functionality can be disabled.

*config.yml*

```
# Choose on which address and port the webserver should listen
webserver:
  enable: true
  username: "admin"
  password: "admin"
  address: "0.0.0.0"
  port: "8080"

# Enable or disable mail notifications
mail:
  enable: true
  server: "smtp.gmail.com"
  username: "user@gmail.com"
  password: ""
  email: "user@i-401.xyz"
  from: "user@gmail.com"

# Set address and login to remote Splunk endpoint (multiple endpoints are possible)
splunk:
  enable: true
  server: "splunk:8089"
  username: ""
  password: ""
  interval: 500

# Set remote snmp server address and community + oids.
# Multiple oids and servers are possible
snmp:
  enable: true
  server: "127.0.0.1"
  community: "my_comm"
  oid: "1.3.6.100.1.2.3.5.1.1.0"

# Enable/disable portscanning (requires root rights)
portscan:
  enable: true
  portspec: "/etc/nidhogg/portspecs.yml"
  mappings: "/etc/nidhogg/mappings.xml"
  timeout: 500
```



```
# Enable/disable arp scanning (requires root rights)
# Use mac list to whitelist devices. (You won't get notifications if those devices getting
connected)
arpscan:
  enable: true
  interface: "eth0"
  db: "/etc/nidhogg/arp.db"
  timeout: 500
  mac:
    - "00:17:88:28:9f:ca"
    - "00:55:da:50:40:64"
    - "34:7e:5c:31:10:e8"
    - "c8:3c:85:3e:e8:dd"
    - "f4:4d:30:68:9b:d4"
```

## 4.2. Portscan configuration files

Defines which target and which ports should be in a special state.

If a port is undefined, it will be ignored in the final report.

mappings.xml is used to bind a spec to a target.

Special thanks to [nmap-analyze](#)

*portspecs.yml*

```
portspecs:
  - name: artoria
    ports:
      - id: 22
        state: open
      - id: 25
        state: closed
```

*mappings.xml*

```
{ "mappings":
  [
    {
      "hostname": "artoria",
      "id": "i-0",
      "ips": ["10.2.1.121"],
      "name": "artoria",
      "portspec": "artoria"
    }
  ]
}
```



## 5. Uninstall

Uninstalling is as easy as installing.

If you've installed nidhogg via .deb, just remove the deb with apt.

*Ubuntu / apt based*

```
apt remove nidhogg
```

*Compiled from source / Binary*

```
rm -r /etc/nidhogg && rm /usr/bin/nidhogg
```

# Nidhogg

Konfiguration / Wartung

Ives Schneider

## Index

1. Info.....	1
2. Konfiguration .....	1
2.1. PRTG.....	1
2.2. Andere NMS.....	2
2.3. SPLUNK .....	2
2.4. SNMP .....	2
3. Wartung.....	3





## 1. Info

Nidhogg ist ein Netzwerkanomalien detection tool.

Da es keine aktive Aufzeichnungen des Netzwerkverkehrs macht und auch nur indirekt mit den Hosts kommuniziert, benötigen die Umsysteme einige Konfigurationen.

### *Requestsize*

Der Portscan generiert pro Host insgesamt 2005 Pakete (118254 Bytes).

Dies könnte zu problemen führen, je nach Anzahl überwachter Host.

#### **IMPORTANT**

Bitte im Hinterkopf bewahren





## 2. Konfiguration

### 2.1. PRTG

Die Kommunikation zwischen PRTG und Nidhogg geschieht auf einer Einwegverbindung via HTTP(s).

Um Sensor Meldungen an Nidhogg zu melden, wird ein sogenanntes Notificationtemplate benötigt.

### Account Settings

-  **My Account**  
Manage your personal account settings like email address, timezone, and audible alarms.
-  **Notification Templates**  
Manage notification methods like email or push. Define how you are informed if a notification trigger is activated.
-  **Notification Contacts**  
Manage notification contacts that PRTG uses to send you notifications. They are unique for each user account.
-  **Schedules**  
Manage schedules to pause monitoring for groups, devices, sensors, and notification delivery based on time and day of the week.



Bei der HTTP Action muss folgende URL eingegeben werden:

---

☒ **Execute HTTP Action**

URL ⓘ

http://nidhogg.lab.i-401.xyz:8080/sensor/%host/%sensor/%status

SNI (Server Name Indication) ⓘ

☒ Do not send SNI (default)  
☐ Send SNI

HTTP Method ⓘ

☒ GET  
☐ POST  
☐ PUT  
☐ PATCH

---

#### Erklärung

- %host - Hostname auf welchen der Sensor alarm geschlagen hat.
- %sensor - Name des Sensors
- %status - Up/Down

Summarize kann deaktiviert werden.

Alle anderen Einstellungen können selbst definiert werden.

Nun kann auf den einzelnen Sensoren Nidhogg als notification Endpoint angegeben werden.

## 2.2. Andere NMS

Solange ein NMS die Möglichkeit besitzt, HTTP Requests an Endpunkte zu versenden, kann Nidhogg angeschlossen werden.

## 2.3. SPLUNK

Splunk benötigt keine weitere Konfiguration um mit Nidhogg zu kommunizieren.

## 2.4. SNMP

Die Kommunikation geschieht über SNMPv2.

Daher sollte ein sicheres Read-Community Passwort gesetzt sein.

Write-Community wird nicht genutzt.



### 3. Wartung

#### *Log cleanup*

Um die Nidhogg logs zu leeren reicht es folgenden Befehl durchzuführen:

```
sudo rm /etc/nidhogg/arp.db
```

Weitere Wartungen werden nicht benötigt.

Nidhogg

Controlling

Ives Schneider

## Index

1. Info.....	1
2. Zeitplan.....	2
3. Meilensteine.....	3
4. Testing.....	4
4.1. Case #1 .....	4
4.2. Case #2 .....	5
4.3. Case #3 .....	6
4.4. Case #4 .....	7
4.5. Case #5 .....	8
4.6. Case #6 .....	9
4.7. Case #7 .....	10
4.8. Case #8 .....	11
4.9. Case #9 .....	12



## **1. Info**

Dieses Dokument beschreibt das Controlling welches während der Arbeit eingesetzt wurde.  
Es beinhaltet das Zeitmanagement sowie die Testcases für das entwickelte Programm.



Task			~ Aufwand (h)												
Erhebung / Analyse			KW36	KW37	KW38	KW39	KW40	KW41	KW42	KW43	KW44	KW45	KW46	KW47	
Ziele definieren	2	Soll IST	<div></div>												
IST-Zustand ermitteln	3	Soll IST	<div></div>	<div></div>											
Projektschnittstellen ermitteln	2	Soll IST	<div></div>	<div></div>											
Projektantrag erstellen	5	Soll IST	<div></div>	<div></div>	<div></div>										
Total															
Anforderungsermittlung			KW36	KW37	KW38	KW39	KW40	KW41	KW42	KW43	KW44	KW45	KW46	KW47	
Infrastruktur analysieren	6	Soll IST		<div></div>											
Produkte Evaluieren	8	Soll IST		<div></div>	<div></div>	<div></div>									
SWOT-Analyse	1	Soll IST		<div></div>	<div></div>										
Issues erstellen	2	Soll IST			<div></div>	<div></div>									
Total			17												
Realisierung			KW36	KW37	KW38	KW39	KW40	KW41	KW42	KW43	KW44	KW45	KW46	KW47	
Evaluierte Produkte installieren / konfigurieren	8	Soll IST			<div></div>	<div></div>	<div></div>	<div></div>							
Issues abarbeiten	35	Soll IST				<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>		
Tool einbinden	8	Soll IST					<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>		
Testing	Fortlaufend				<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	
Total			51												
Dokumentation			KW36	KW37	KW38	KW39	KW40	KW41	KW42	KW43	KW44	KW45	KW46	KW47	
Grundgerüst erstellen	0.5	Soll IST	<div></div>												
Allgemeine Dokumentation	Fortlaufend		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	
Anleitungen	10	Soll IST								<div></div>	<div></div>	<div></div>	<div></div>		
Abschluss (Titelblatt, Ausdruck, Binden)	2	Soll IST										<div></div>	<div></div>	<div></div>	
Total			12.5												
Diverses			KW36	KW37	KW38	KW39	KW40	KW41	KW42	KW43	KW44	KW45	KW46	KW47	
Reserve	7.5	Soll IST												<div></div>	
Total			7.5												

Geplante Arbeitszeit 100  
Insgesamt benötigte Arbeitszeit 0

Legende

Geplanter Aufwand

Benötigte Zeit

Variable Zeit

Meilensteine



### 3. Meilensteine

#### *KW36 | Projektantrag*

Der erste Meilenstein konnte ohne Probleme eingehalten werden.

Nach einigen Verbesserungen wurde der Projektantrag am 13.09.2019 angenommen.

#### *KW38 | Evaluation*

Die Produktevaluation konnte bereits etwas früher abgeschlossen werden.

Dies konnte aufgrund bereits vorhandenem Wissen der zu evaluierenden Software geschehen.

#### *KW45 | Issues*

Die Issues (Aufbau des Programms), nahm mehr Zeit in Anspruch als anfangs geplant.

Allerdings ist dies nicht weiter schlimm, da beim gekennzeichneten Meilensteins, bereits ein POC des Tools vorhanden war.

#### *KW44 | Einbindung*

Der Meilenstein verschob sich ebenfalls um ein paar Tage, dies war aufgrund einer nicht Komplette fertigen Software Konfiguration, welche danach fertiggestellt wurde.

#### *KW47 | Abschluss*

Obwohl einige der anderen Meilensteine sich etwas verschoben hatten, konnte der Abschluss Termingerecht eingehalten werden.





## 4. Testing

### 4.1. Case #1

<b>Title</b>	Start & Listen
<b>Beschreibung</b>	Nidhogg startet und hört auf den gewünschten Port (8080)
<b>Config</b>	<pre>webserver:   ip: "0.0.0.0"   port: 8080</pre>
<b>Eingabe</b>	<pre>./nidhogg</pre>
<b>Check</b>	<pre>curl localhost:8080</pre>
<b>Soll</b>	Request antwortet mit 200
<b>Ist</b>	<pre>` HTTP/1.1 200 OK content-length: 1994 content-type: text/html date: wed, 13 Nov 2019 17:20:53 GMT `</pre>
<b>Erfolgreich</b>	Ja



## 4.2. Case #2

<b>Title</b>	Portscan
<b>Beschreibung</b>	<p>Der eigene Computer wird als zu überwachende Maschine eingetragen. Danach wird ein Port geöffnet, welcher als geschlossen angegeben ist. Hierbei wird die automatische Benachrichtigung getestet. d.H. der Portscan sollte nach angegebener Zeit automatisch durchgeführt werden.</p>
<b>Config</b>	<pre>portspecs:   - name: yorha     - id: 8081       state: closed</pre>
<b>Eingabe</b>	n/a
<b>Check</b>	Automatische Mail wird abgewartet.
<b>Soll</b>	<pre>Time: [Nov 17 17:15:49]  [Scan] Pass: 0 Fail: 1 Analysis: IP: 10.0.0.25 Result: [ALERT] Port: 8081 - State: OpenButClosed</pre>
<b>Ist</b>	<pre>Time: [Nov 17 17:15:49]  [Scan] Pass: 0 Fail: 1 Analysis: IP: 10.0.0.25 Result: [ALERT] Port: 8081 - State: OpenButClosed</pre>
<b>Erfolgreich</b>	Ja



### 4.3. Case #3

<b>Title</b>	Arpscan
<b>Beschreibung</b>	Es wird während dem Betrieb eine zusätzliche VM im Netzwerk aufgeschalten. Nach dem hochfahren, sollte die IP einen Broadcast ARP-Request durchführen um den DHCP Server zu finden.
<b>Config</b>	<pre>arpscan:   enable: true   interface: "eth0"   db: "/etc/nidhogg/arp.db"   timeout: 50   mac:     - 00:00:00:00:00:00</pre>
<b>Eingabe</b>	n/a
<b>Check</b>	Automatische Mail wird abgewartet.
<b>Soll</b>	<pre>[Time] New device found: 00:50:56:93:58:5c</pre>
<b>Ist</b>	<pre>[[Nov 17 17:20:01]] New device found: 00:50:56:93:58:5c</pre>
<b>Erfolgreich</b>	Ja



#### 4.4. Case #4

<b>Title</b>	Web - Authentication
<b>Beschreibung</b>	Das Webinterface muss eine Authentifizierung besitzen, welche nach einem erfolgreichem Login den Zugang zu weiteren Funktionalitäten bietet. = Hier wird getestet, ob die Informationen welche in config.yml angegeben sind, respektiert werden.
<b>Config</b>	<pre>webserver:   enable: true   username: "admin"   password: "hunter2"   address: "0.0.0.0"   port: "8080"</pre>
<b>Eingabe</b>	<pre>Username: admin Password: admin - Username: admin123 Username: blah - Username: 1=1 or 1;-- Password: 1=1 or 1;--</pre>
<b>Check</b>	Manueller check
<b>Soll</b>	Login sollte nicht erfolgreich sein
<b>Ist</b>	Es wird kein Session-Token erstellt.
<b>Erfolgreich</b>	Ja



#### 4.5. Case #5

<b>Title</b>	Deaktivierung von Features
<b>Beschreibung</b>	In der config.yml wird der Arp-Scan deaktiviert. Der Arpscan sowie die Url /arp sollte nun nicht mehr verfügbar sein.
<b>Config</b>	<pre>arpscan:   enable: false   interface: "wlp58s0"   db: "/etc/nidhogg/arp.db"   timeout: 500   mac:     - "00:00:00:00:00:00"</pre>
<b>Eingabe</b>	n/a
<b>Check</b>	Es wird eine zusätzliche Maschine hochgefahren. Manueller check auf /arp und timeout abwarten.
<b>Soll</b>	Es sollte kein Alert-Mail versendet werden.
<b>Ist</b>	Es wird kein Arp-alert versendet.
<b>Erfolgreich</b>	Ja



#### 4.6. Case #6

<b>Title</b>	Portscan: Unmögliche Konfiguration
<b>Beschreibung</b>	In der Portspec wird angegeben, dass ein Port sowohl offen wie auch geschlossen sein muss.
<b>Config</b>	<pre>portspecs:   - name: artoria     ports:       - id: 22         state: open       - id: 22         state: closed</pre>
<b>Eingabe</b>	./nidhogg
<b>Check</b>	n/a
<b>Soll</b>	Es sollte die erste Value genommen werden.
<b>Ist</b>	Nidhogg meldet einen Fehler sobald SSHd nicht mehr aktiv ist.
<b>Erfolgreich</b>	Ja



#### 4.7. Case #7

<b>Title</b>	Portscan: Unbekannter Port wird geöffnet
<b>Beschreibung</b>	Ein zusätzlicher Port (30718) wird mit netcat geöffnet und simuliert eine binding shell.
<b>Config</b>	<pre>portspecs:   - name: artoria     ports:       - id: 22         state: open</pre>
<b>Eingabe</b>	./nidhogg
<b>Check</b>	Das Webinterface /port wird manuell aufgerufen & die automatische Mail wird abgewartet.
<b>Soll</b>	nidhogg sollte den unbekannten Port melden.
<b>Ist</b>	10.0.0.36 - Port: 30718 - State: Open -- Time: [Nov 17 16:58:41]  Pass: 0 Fail: 1 Analysis: IP: 10.0.0.36 Result: [ALERT] Port: 30718 - State: ClosedButOpen
<b>Erfolgreich</b>	Ja



#### 4.8. Case #8

<b>Title</b>	PRTG
<b>Beschreibung</b>	<p>PRTG wird eine Meldungen senden, dass sich ein port geändert hat. Dadurch wird ein Mail aktiviert, welches die letzte Nachricht von dem Host in Splunk einbetten sollte.</p>
<b>Config</b>	<p>Es wird ein notification Trigger auf nginx gelegt. Sobald der State sich ändern sollte, wird wird die URL: nidhogg.hosts.i-401.xyz mit den Sensor informationen via GET aufgerufen.</p>
<b>Eingabe</b>	<pre>systemctl stop nginx</pre>
<b>Check</b>	Nginx wird manuell gestoppt.
<b>Soll</b>	nidhogg sollte ein Mail mit Portscan inhalt senden.
<b>Ist</b>	<p>Host: nginx changed sensor: HTTP to state: UP Please investigate!</p> <p>Pass: 0 Fail: 1 Analysis: IP: 10.0.0.201 Result: [ALERT] Port: 22 - State: ClosedButOpen</p>
<b>Erfolgreich</b>	Ja





#### 4.9. Case #9

<b>Title</b>	Splunk
<b>Beschreibung</b>	Der Output der letzten Splunk info wird via Mail gesendet, sobald PRTG einen Host spezifischen Sensor meldet.
<b>Config</b>	Es wird ein notification Trigger auf nginx gelegt. Sobald der State von Load sich ändern sollte, wird die URL: nidhogg.hosts.i-401.xyz mit den Sensor informationen via GET aufgerufen.
<b>Eingabe</b>	<pre>curl nidhogg.hosts.i-401.xyz:8080/sensor/nginx/load/up</pre>
<b>Check</b>	n/a
<b>Soll</b>	nidhogg sollte ein Mail mit Splunk content senden.
<b>Ist</b>	Host: nginx changed sensor: Load to state: up Please investigate!  Last Splunk messages: [2019-11-17 17:23:50.000 UTC] Nov 17 17:23:50 nginx sudo: pam_unix(sudo:session): session closed for user root
<b>Erfolgreich</b>	Ja