Hauptstudie

Anomalie Detection in Netzwerken

Ives Schneider

Index

1. Management Summary

2. Erhebung

2.1. Netzwerk

TODO

2.1.1. Baseline

TODO

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 erkennen könnte.

3.3. Kill-Chain

3.3.1. Reconnaissance

Beschreibung

In dieser Phase werden weitere Informationen über das Ziel beschaft.
Während dieser Phase, befindet sich der Angreiffer meist noch ausserhalb des zu angreifenden Netzwerkes.
Allerdings ist Reconnaissance eine wiederholende Phase, welche während des gesamten Angriffs vortbeständig durchgeführt wird.

Erkennung

Solange sich der Angreiffer ausserhalb des Netzwerkes befindet, gibt es nur eine limitierte Anzahl an Erkennungsmassnahmen.
Da der Scope der Applikation 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 Angreiffer 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. 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 die Applikation 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 Angreiffer. 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.

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

Die Applikation spezialisiert sich eher auf Erkennung und nicht auf Verhinderung.
Allerdings wird mithilfe des Portscanners die Binding shell erkannt werden.
Dropper + Reverse Shell können leider nicht erkannt werden.

3.3.5. Installation

Beschreibung

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

Erkennung

Es gibt viele massnahmen, wie man ein Backdoor erzeugen kann. Die Applikation 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

Beschreibung

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

Erkennung

Siehe Phase "Installation"

3.3.7. Actions on Objective

Beschreibung

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

Erkennung

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

3.4. Einstufung

Die Einstufung erfolgt anhand mehrerer Sicherheitsstufen mit zusätzlichen Unterstufen.
Je nach Einstufung werden verschiedene Massnahmen getroffen.

3.4.1. Event

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

NOTE	High Bandwith: {IP}

3.4.2. Alert

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

CAUTION	New device found: {IP} {MAC}

3.4.3. Incident

Anomalien welche auf lateral movement hinweisen könnten. Beispiel:

WARNING	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

Falls der ARP-Request Sensor einen neuen Host im Netzwerk finden sollte, wird folgender Workflow ausgelöst.

Die Geschwindigkeit der Erkennung ist Netzwerkgrössen abhängig.

3.5.2. New open Port

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

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.

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.

4. Baseline

Die Baseline wird mithilfe von PRTG erstellt.
TODO

4.1. Netzwerk

TODO

4.2. Load

TODO

5. Applikation

Die Applikation welche unter dem Namen "Nidhogg" entwickelt wird, soll als Anlaufstelle für Anomalieerkennungen 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.

5.1. Technology

Die Applikation 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 die Applikation möglichst erweiterbar und ressourcenschonend zu schreiben.

5.2. 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 werde sollte.
Allerdings besteht bereits ein Programm von welchem ich die Funktionalität wiederverwenden kann.

First Tier

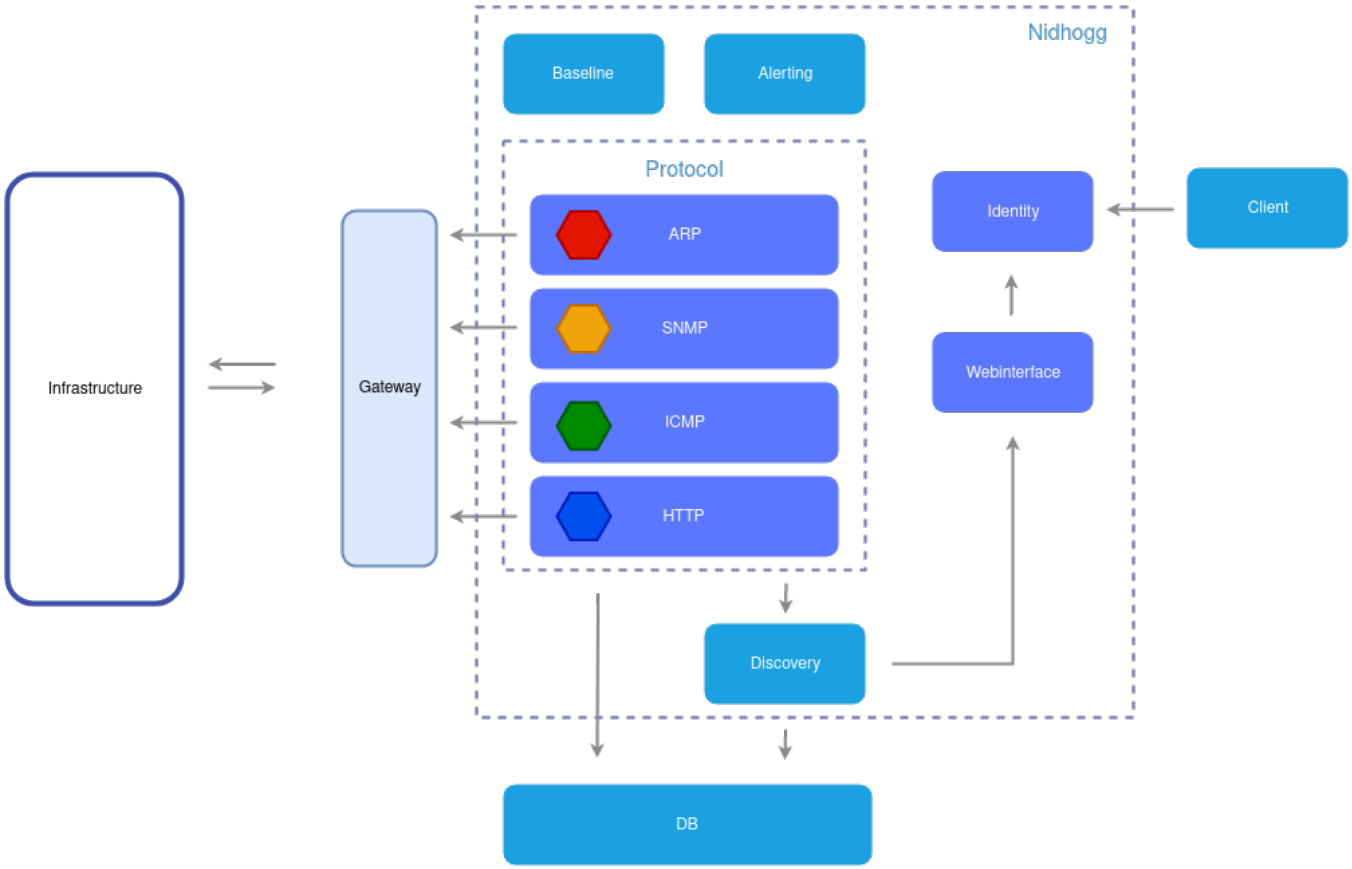
Webinterface

Second Tier

jProgramm Logik

Third Tier

SQLite



5.3. Diagramme

TODO

5.4. Implementation

TODO

6. Weiteres vorgehen

TODO

6.1. Installation

Um die Installation möglichst einfach zu halten, soll Nidhogg einfach über CLI installiert werden können.
Unter Linux wird dies mithilfe des .deb Formates zustandegebracht.
Windows Systeme werden eine portable Binary erhalten.

6.2. Konfiguration

Einstellungen werden via eines YAML Files vorgenommen.
Um möglichst flexibel zu bleiben, werden Default Einstellungen mit der Applikation mitgebracht.

config.yml

```
webserver:
    address: "0.0.0.0"
    port: "8080"

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

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: "~/Repos/nidhogg/portspecs.yml"
    mappings: "~/Repos/nidhogg/mappings.xml"
    timeout: 500

arpscan:
    interface: "eno1"
    db: "/tmp/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"
```

Mappings.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": ["10.0.0.33"],
            "name": "artoria",
            "portspec": "artoria"
        },
        {
            "hostname": "splunk",
            "id": "i-1",
            "ips": ["10.0.0.36"],
            "name": "splunk",
            "portspec": "splunk"
        }
    ]
}
```

porspec.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
  - name: splunk
    ports:
      - id: 22
        state: open
      - id: 8080
        state: open
```

7. Controlling

TODO

7.1. Testing

TODO

8. Kosten

Die Kosten belaufen sich auf die aufgewendete Arbeistzeit.

Da die Entwicklung mit OpenSource 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.

9. Reflextion

TODO

10. Freigabe

TODO <<<

11. Anhang

11.1. Installationsdokumentation

TODO

11.2. Wartungsdokumentation

TODO