Hochschule Wismar
University of Applied Sciences
Technology, Business and Design
Fakultät für Ingenieurwissenschaften

Bachelor-Thesis

Detektion von Command and Control Datenverkehr mittels Open Source IDS-/IPS-Systemen

Von: Andreas Barttels

Betreuer: Prof. Dr. rer. nat. Nils Gruschka
Zweitbetreuer: Prof. Dr. Olaf Hagendorf

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Dass Unternehmen sich mit Cyberangriffen beschäftigen müssen, zeigt ein Statistikreport von Statista, welcher aktuelle sowie historische Daten darstellt[26]. Alleine in einer Meinungsumfrage, bei der es darum geht, wie sich Cyberangriffe wohl künftig entwickeln, zeigt eindrücklich das Stimmungsbild über die letzten Jahre gesehen. Hier war 2011 noch 35 % der Führungskräfte deutscher Unternehmen der Meinung, dass Cyberangriffe sich stark oder leicht zurückentwickeln. Weitere 60 % der Meinung, dass lediglich eine leichte Steigung erfolgt, nur 5 % vertraten die Meinung, dass es hier zu einem starken Anstieg kommt. Zahlen aus 2023 zeigen ein komplett anderes Bild, sodass niemand mehr davon ausgeht, dass ein Rückgang von Cyberangriffen erfolgt. 46 % gehen von einer leichten Steigung aus, ganze 54% von einer starken Steigung[26, S. 25]. Dieses Meinungsbild wird von den polizeilich erfassten Fällen von Cyberkriminalität gestützt. 2007 lag die Anzahl der erfassten Fälle bei 34.180, 10 Jahre später, 2017, mit 85.960 erfassten Fällen mehr als verdoppelt und 2022 136.865 erfasste Fälle[26, S. 4].

In 2023 gaben 58% befragter Unternehmen an, einen Cyberangriff gehabt zu haben[26, S. 29]. Als Cyberangriff werden Vorfälle bezeichnet wie z. B. Phishing (31 %) aber auch Ransomware (23%)[26, S. 28].

Konkrete Hinweise auf eine Datenexfiltration gab es unter den Befragten im Jahre 2015 zu 14 %. In 2017 haben sich die Zahlen bereits gewandelt, sodass 16 % mindestens von einem Datenabfluss betroffen waren, weitere 28 % sogar mehrfach betroffen. Die Zahlen haben sich seitdem um nur wenige Prozentpunkte verändert[26, S. 30].

Bei Cyberangriffen kommt häufig auch auf Systemen ausgeführte oder installierte Malware zum Einsatz, welche dann unterschiedliche Zwecke erfüllen kann. Ein bekanntes Beispiel sind Systeme die dann als `Zombies` bezeichneten werden. Diese führen auf Anweisung sog. Distributed Denial of Service (DDoS) Angriffe aus, die oft dazu führen, dass ein Dienst oder Webseite überlastet wird und entweder langsamer oder gar nicht mehr antwortet. Ein weiteres Beispiel sind *Information-Stealer*, wo

im Rahmen einer Datenexfiltration gespeicherte Passwörter oder E-Mails abfließen. Oft persistiert sich derartige Malware, unabhängig vom verfolgten Zweck. Auch sind diese oftmals Modular aufgebaut, sodass in Abhängigkeit gewisser Faktoren z. B. Module nachgeladen werden und andere Zwecke erfüllen kann. Hierzu gehört auch eine Art der Trojaner-Funktionen, auch Backdoor genannt, sodass ein Angreifer jederzeit auf das infizierte System zugreifen kann.

Die meisten dieser Malware-Typen haben eins gemeinsam, sie halten Kommunikation (sog. `Beacons`), laden Aktualisierungen herunter oder führen Befehle aus, die der Command and Control (C2)-Server ihnen zugesandt hat. All dies passiert über C2-Kommunikation, um dessen Erkennung es in der vorliegenden Bachelor-Thesis geht.

1.2 Ausgangssituation

Die zuverlässige Erkennung von C2-Kommunikation ist nach der *Defense in Depth* Strategie eine weitere wichtige Komponente. Neben der Tatsache, dass oben beschriebene Malware ggf. durch eine installierte Antivirus (AV)- oder Endpoint Detection and Response (EDR)-Lösung rechtzeitig erkannt und isoliert wird, spielt neben der hostbasierten Erkennung auch die netzwerkbasierte Erkennung eine tragende Rolle. Selbst in homogenen IT-Infrastrukturen existieren Ausnahmen, welche Lücken offen lassen, die Angreifer ausnutzen könnten. Auch die sogenannte Schatten-IT könnte solche Schlupflöcher öffnen. In Bildungseinrichtungen, wie z. B. Hochschulen oder Universitäten, können hostbasierte Sicherheitsmechanismen zum Teil nicht installiert werden, obgleich ein identisches Risiko für die dortige IT-Infrastruktur besteht. Die Gründe hierfür sind unterschiedlich, stehen meist aber unter dem Einfluss der Forschungsfreiheit. In solchen Fällen ist eine netzwerkbasierte Erkennung umso wichtiger.

Für die Erkennung von C2-Kommunikation sind gesonderte Systeme vonnöten. Hierbei handelt es sich um Intrusion Detection System (IDS) oder Intrusion Prevention System (IPS). Eventuell wird eine solche Funktion aber auch durch eine Next-Generation Firewall (NGFW) bereitgestellt. Da die vorliegende Thesis jedoch ausschließlich Open Source IDS-/IPS-Systeme behandelt, wird der Umstand an dieser Stelle lediglich der Vollständigkeit halber aufgeführt.

Um bei einem Beispiel über die aktuelle Situation zu geben, sind öffentlich verfügbare Informationen zu gängigen IDS-/IPS-Systemen und möglichen Regeln zur Detektion von C2-Kommunikation gesichtet worden. Hierbei fallen unterschiedliche

Probleme auf. So werden teilweise Fehler im Protokoll erkannt, inwiefern dies auch möglicherweise auf missbräuchlich veränderten Pakete zutrifft, ist fraglich. Einige Regeln beziehen sich auf Malware, welche jedoch und ob diese nicht ggf. schon veraltet und gar nicht mehr verwendet wird, ist nicht festzustellen. Auch beinhalten die Beschreibungen selten genauere Details darüber, was einen Alarm hat auslösen lassen.

Eine entsprechende qualitative Auswertung ist nicht verfügbar. Daher ist es IT-Verantwortlichen nicht ohne weiteres Möglich, ohne ein Proof of Concept (PoC) Aussagen darüber zu treffen, wie gut eine Erkennung ist.

1.3 Zielsetzung

Ein Ziel dieser Bachelor Thesis ist es einem Überblick über aktuelle IDS-/IPS-Systeme zu geben, um die folgende Forschungsfrage zu beantworten:

Forschungsfrage: Wie gut ist die Detektion von C2-Datenverkehr durch Open Source IDS-/IPS-Systeme? In der vorliegenden Bachelor-Thesis werden geeignete IDS-/IPS-System anhand von definierten Kriterien ausgewählt. Die Detektionsraten der Systeme werden anhand von simulierter C2-Kommunikation betrachtet. Für die Simulation der C2-Kommunikation wird sich eines Open Source C2-Frameworks bedient, sodass keine echte Malware zum Einsatz kommt. Ziel ist es, den dabei entstehende Datenverkehr zu untersuchen, um individuelle Detektionsregeln erstellen zu können.

Neben der Frage, wie IDS-/IPS-Systeme funktionieren, wird eine Auswahl an Systemen einer Bewertung anhand definierter Kriterien unterzogen.

Anschließend wird ein geeignetes IDS-/IPS-System einem zweiteiligen Test unterzogen. Im ersten Teil wird mittels eines Open Source C2-Frameworks C2-Kommunikation erzeugt, um die folgenden Fragen zu beantworten:

- Welche Protokolle können missbraucht werden, um C2-Kommunikation zu tarnen?
- Besitzt das System Regeln, um unterschiedliche Methoden der C2-Kommunikation zu erkennen?
- Welche für C2-Kommunikation genutzte Protokolle werden erkannt, bzw. wie gut ist die Detektionsrate?

Im zweiten Teil des Tests wird für ein Protokoll eine individuelle Detektionsregel erstellt, um exemplarisch die folgenden Fragen zu beantworten:

- Wie wird die C2-Kommunikation in dem Protokoll verborgen?
- Wie gut lassen sich individuelle Regeln zur Detektion implementieren?
- Können für die durch das C2-Framework verwendete Methoden neue Detektionsregeln implementiert werden?
- Wie gut ist die Detektionsrate der individuell implementierten Regel?

Vorliegende Arbeiten beschäftigen sich mit verschiedenen Aspekten von IDS-/IPS (Anomalieerkennung, Evaluierung unterschiedlicher Aspekte, sichere Konfiguration), oder der praktischen Verwendung von C2. Wenn spezielle Systeme betrachtet werden, handelt es sich in der Regel um Open Source Systeme.

Viele der vorgefundenen Arbeiten sind veraltet, sodass kein technologischer Bezug mehr bestehen kann. Insbesondere, weil durch Open Source eine ständige Weiterentwicklung sowohl der Anwendungen (IDS-/IPS) als auch der bestehenden Regelwerke zu erwarten ist.

Die vorliegende Arbeit soll hier einen neuen Blickwinkel liefern, welcher vor allem für operatives IT-Personal aus Netzwerk- oder Security-Fachbereichen von Relevanz sein kann. Die im späteren Verlauf vorgestellten Untersuchungen könnten auch als Ausgangspunkt für weitere Forschungsfragen genutzt werden. Auch weitere praktische Aspekte wie z. B. die Erstellung eines Frameworks zur (halb-)automatischen Erkennung von C2-Kommunikation und entsprechender Regelerstellung zur Detektion könnten daraus folgern.

1.4 Vorgehen

Kapitel 1 wird als Einleitungskapitel die Inhalte des Exposé behandeln.

Kapitel 2 dient als Grundlagenkapitel zur Erläuterung der Netzwerktechnik und behandelt darauf aufbauend die in der Bachelorarbeit thematisierten Sicherheitseinrichtungen, insbesondere Open Source IDS/IPS-Systeme. Außerdem wird Malware behandelt, hier primär der von Malware ausgehende C2-Datenverkehr. Im Speziellen werden die unterschiedlichen Methoden, also die missbräuchliche Verwendung unterschiedlicher Protokolle, betrachtet, welche benutzt werden, um die Kommunikation zu verbergen.

Kapitel 3 wird im Rahmen eines Vergleichs unterschiedliche Open Source IDS-/IPS-Systeme betrachten und diese bewerten. Hierbei werden Faktoren wie z. B. der Anwendungsbereich (hostbasierte-, netzwerkbasierte Erkennung) oder die Erweiterbarkeit eine Rolle spielen. Ziel ist es, ein geeignetes Systeme zu identifizieren, welches für das nachfolgende Experiment verwendet wird.

Kapitel 4 wird den für das Experiment benötigten Versuchsaufbau und die Konfiguration der einzelnen Bestandteile beschrieben. Im Rahmen des Praxistests wird durch ein C2-Framework C2-Kommunikation erzeugt und die Detektionsraten der geeigneten IDS-/IPS-Systeme bewertet. Hierbei werden unterschiedliche Methoden zum Verschleiern des Datenverkehrs getestet, um einen besseren Gesamtüberblick zu erhalten.

In Kapitel 5 wird der erzeugte C2-Datenverkehr genauer untersucht, um Möglichkeiten für die Regelerstellung zu identifizieren. In Abhängigkeit der praktischen Relevanz werden für die identifizierten Merkmale individuelle Regeln verfasst. Die Detektionsrate der IDS-/IPS-Systeme werden auf Basis der neu erstellten Regel erneut bewertet.

Im letzten Kapitel, Kapitel 6, wird ein Fazit über die gesamte Bachelor-Thesis gezogen. Unter anderem wird es um die Funktionen der IDS-/IPS-Systeme sowie die Detektionsraten gehen. Außerdem wird die Möglichkeiten der Regelerstellung und Erweiterbarkeit und die identifizierten Vor- oder Nachteile, welche sich aus dem Experiment ergeben, behandelt. Zuletzt wird ein Ausblick auf mögliche nachfolgende Arbeiten gegeben werden.

Nachfolgend wird die *kursive Schreibweise* dazu verwendet, englische Wörter zu markieren oder ein für den Absatz wichtigen Begriff hervorzuheben. Die Schreibweise in `dieser Schriftart` wird dazu verwendet, technische Elemente oder (technische) Eigennamen kenntlich zu machen.

2 Grundlagen der Netzwerktechnik und Sicherheit

2.1 Netzwerktechnik

Das nachfolgende Kapitel dient nicht einer umfassenden Aufarbeitung des Themas Netzwerktechnik, sondern eine kurze Wiederholung der Themenkomplexe. Themen, die für das weitere Verständnis der Arbeit notwendigen sind, werden eingehender beleuchtet.

2.1.1 Standards und Netzwerkprotokolle

Im Bereich der Netzwerktechnik existieren eine Vielzahl an Standards und Protokollen, welche implementiert sein müssen, um eine reibungslose Kommunikation zu gewährleisten. So ist die Interoperabilität zwischen Netzwerkkomponente verschiedener Hersteller, z. B. `Cisco` und `Juniper` oder verschiedener Betriebssystemen (Client: Windows, Server: Linux) gewährleistet.

Datenpakete, auch *Frames* genannt, sehen je nach dem vorliegenden Standard unterschiedlich aus. Der Ethernet 2 Standard ist vor allem in moderneren Local Area Network (LAN) gegenüber dem älteren `IEEE 802.3` dominierender Standard. Wie ein Ethernet Datenpaket aussieht, kann der nachfolgenden Abbildung 1 entnommen werden.[42, S. 226 f.]

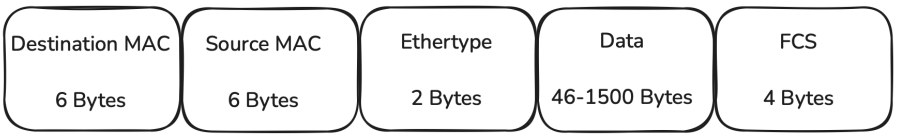

Abbildung 1: Ethernet II Frame

- Destination Media Access Control (MAC): MAC-Adresse des Ziels
- Source MAC: MAC-Adresse der Quelle
- Ethertype: Bezeichner für die in `Data` gespeicherten Daten, z. B. Internet Protocol (IP)- oder Address Resolution Protocol (ARP)-Paket

- Data: Nutzdaten der höheren Schichten wie z. B. Transport Control Protocol (TCP)/IP
- Frame Check Sequence (FCS): Checksumme zur Validierung auf Datenfehler

Die beiden relevantesten Transportprotokolle (OSI Layer 4) sind TCP und User Datagram Protocol (UDP). Die folgende Tabelle 1 gibt einen Überblick über die Kernfunktionen der beiden Protokolle[42, S. 37 ff., S. 50].

Tabelle 1: Vergleich TCP und UDP

Merkmal	**TCP**	**UDP**
Art des Protokolls	Verbindungsorientiert	Verbindungslos
Verbindungsaufbau	3-Wege-Handshake	Keiner
Fehlererkennung	Prüfsummen und 3-Wege-Handshake	Prüfsummen
Fehlerkorrektur	Automatische Retransmission	Nein, Paket wird ggf. erneut gesendet
Geschwindigkeit	Geringer, da Overhead im Paket und 3-Wege-Handshake	Schneller, wenig Overhead
Persistente Sitzung	Ja	Nein

Einige der gängigen Anwendungsprotokolle sind in den folgenden Unterabschnitten erläutert.

2.1.2 DNS

Domain Name System (DNS) ist ein essenzieller Netzwerkdienst, der z. B. die Auflösung von Domänennamen wie `www.google.de` in die entsprechenden IP-Adressen ermöglicht, unter denen die Webserver tatsächlich erreichbar sind. DNS löst Namen zu IP-Adressen auf. Es existieren unterschiedliche Resource Record (RR), sodass je nach Zweck IP-Adressen aufgelöst werden können. Die gängigsten RRs sind die folgenden:

- **Address (A) Record** Löst Domänennamen zu IP-Adresse auf
- **AAAA Record** Löst Domänennamen zu IPv6-Adresse auf
- **Canonical Name (CNAME) Record** Alias eines Domänennamen
- **Mail Exchange (MX) Record** Löst die IP-Adresse des E-Mail-Servers auf

- **Name Server (NS) Record** Löst autoritative NS für eine Zone auf
- **Pointer (PTR) Record** Reverse DNS, zeigt Domänenname(n) zu einer IP-Adresse auf

Bei DNS handelt es sich um ein hierarchisches System, bei welchem autoritative NS für eine *Zone* verantwortlich sind und DNS-Anfragen für diese beantworten. Eine Zone ist ein logischer Verwaltungsbereich unter welchem sich eine oder mehrere Domänen (z. B. `beispiel.de`) und ggf. deren Subdomänen (z. B. `portal.beispiel.de`) befinden.[55]

Eine typische DNS-Anfrage sieht wie folgt aus[58, S. 695 f.]:

1. Ein Client sendet seine DNS-Anfrage an den im System eingetragenen *Resolver*, meist gleichzeitig auch das Standardgateway oder ein öffentlicher DNS Server
2. Der Resolver prüft, ob die Anfrage mit den Informationen des eigenen Cache beantwortet werden kann
3. Ist dem nicht so, sendet der Resolver eine Anfrage an den NS Root-Server für die zuständige Top-Level-Domain (TLD) (z. B. `.de`)
4. Der NS Root-Server gibt Adresse des autoritative NS zurück an den Resolver
5. Der Resolver fragt den autoritative NS zu der gewünschten Domäne an, dieser liefert den angeforderten Eintrag zurück
6. Der Resolver speichert die Informationen für die Zeitdauer der Time-to-Live (TTL) in seinem Cache und beantwortet die DNS-Anfrage des Clients

In Form einer DNS-Response sieht dies wie in Abbildung 2 dargestellt aus.

```
Ethernet II, Src: VMware_c9:71:06 (00:0c:29:c9:71:06), Dst: VMware_c0:9b:39 (00:0c:29:c0:9b:39)
Internet Protocol Version 4, Src: 8.8.4.4, Dst: 10.0.0.30
User Datagram Protocol, Src Port: 53, Dst Port: 61100
Domain Name System (response)
    Transaction ID: 0xe49e
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > www.google.com: type A, class IN
  v Answers
    v www.google.com: type A, class IN, addr 142.250.181.228
        Name: www.google.com
        Type: A (1) (Host Address)
        Class: IN (0x0001)
        Time to live: 149 (2 minutes, 29 seconds)
        Data length: 4
        Address: 142.250.181.228
    [Request In: 34]
    [Time: 0.012740000 seconds]
```

Abbildung 2: Exemplarische Darstellung einer DNS-Query für einen A-Record zu google.de und dazugehörige DNS-Response

2.1.3 HTTP

Hypertext Transfer Protocol (HTTP) ist ein für den Abruf von Webseiten verwendetes Protokoll. Je nach Anwendung oder Aktion existieren unterschiedliche Methoden zur Anfrage von Daten bzw. zur Übertragung von Daten. Die gängigsten HTTP Methoden sind `GET`, um eine Ressource anzufordern und `POST`, um Daten zu Senden bzw. eine Ressource auf dem Server zu erstellen[29].

```
1 GET /index.html HTTP/1.1
2 Host: www.example.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: keep-alive
```

Listing 1: Exemplarischer HTTP *Request Header*

2.1.4 HTTPS und TLS

Hypertext Transfer Protocol Secure (HTTPS) funktioniert im Kern ähnlich wie HTTP, jedoch sind die eigentlichen Nutzdaten verschlüsselt. Dies passiert typischerweise über Transport Layer Security (TLS). Die TLS-Verbindung ist auch ähnlich wie die TCP-Verbindung als virtueller Tunnel zwischen Client und Server zu sehen. TLS selber verfügt über mehrere Teilprotokolle um alle nötigen Funktionen zu erfüllen.[61]

- **Handshake Protocol** Verbindungsaufbau, Austausch und Aushandlung der unterstützten kryptografischen Parameter
- **ChangeCipherSpec Protocol** Anwendung ggf. neuer kryptografischen Parameter
- **Alert Protocol** Übertragung von Warnungen oder Fehlern
- **Application Data Protocol** Sicherstellung der Verschlüsselung der Anwendungsdaten über das *Record Protocol*
- **Record Protocol** Stellt die eigentliche Verschlüsselung zur Verfügung

Ein vollständiger Vorgang zum Aufruf einer HTTPS-Webseite würde wie folgt aussehen:[58, S. 441 ff.]

- Aufbau einer TCP-Verbindung zwischen dem anfragenden Client und dem Webserver
- Aufbau einer verschlüsselten TLS-Verbindung
- Übertragen der eigentlichen HTTPS-Nutzdaten

2.2 Netzwerksicherheit

Wie bereits in der Motivation beschrieben besteht ein immer größerer Bedarf nach IT-Sicherheit. Im Folgenden wird zunächst ein Überblick gegeben, wie Sicherheitsmechanismen aussehen können und Implikationen beim Mitschneiden von Datenverkehr (*sniffing*). Zuletzt wird die Technologie von IDS-/IPS bzw. Network Intrusion Detection System (NIDS) näher betrachtet, bevor in Kapitel 3 konkrete Systeme evaluiert werden.

2.2.1 Sicherheitsüberwachung

Um all den oben genannten Gefahren zu begegnen, ist eine Sicherheitsüberwachung des Netzwerks wichtig. Hierbei spielen häufig Security Information and Event Management (SIEM) Systeme eine Rolle. Hier werden großflächig Logdaten unterschiedlicher Systeme wie Netzwerkkomponenten sowie Server-, Clientsysteme oder Protokolle anderer Dienste zusammengeführt. Durch die Korrelation vieler Ereignisse unterschiedlicher Quellen können Angriffe, Anomalien oder Sicherheitsverletzungen frühzeitig detektiert werden. Grundsätzlich lassen sich zwei verschiedene Ansätze für die Erstellung von Alarmen beschreiben. Im ersten Fall soll schadhaftes Verhalten erkannt und alarmiert werden. Der zweite Fall funktioniert eher anders herum, hierbei soll legitimes Verhalten in Form eines Normalzustands-Baselinings erkannt und Abweichungen davon erkannt und alarmiert werden.

Neben der Überwachung der lokalen Aktivitäten auf einem System lässt sich auch das Netzwerk überwachen. Eine passive Methode wäre die Erfassung von `NetFlow`-Daten, um ein Bericht darüber zu erhalten, welche Systeme miteinander kommuniziert haben. Auch so würden sich ohne direkten Eingriff in den Netzwerkverkehr passiv Alarme auf gewisse Anomalien definieren lassen. Die eher aktive Methode ist der Einsatz einer IDS-/IPS oder einer NGFW, die genaueren Methoden werden im folgenden Unterabschnitt 2.2.2 näher erläutert werden.

2.2.2 Sniffing

Um den Datenverkehr im Netzwerk umfassend analysieren zu können, muss dieser an einer geeigneten Stelle aufgezeichnet werden, sogenanntes *sniffing*. Es existieren unterschiedliche technische Methoden, den Datenverkehr aufzuzeichnen, welche unterschiedliche Implikationen mit sich bringen[14]:

- **Hub** Vorläufer eines Netzwerkswitches, bei dem alle Pakete zu allen Netzwerkports weitergeleitet werden, einfach installiert, hohe Paketverluste bei Last
- **Mirrorport** Eine Netzwerkschnittstelle an einem Switch wird auf eine weitere gespiegelt, Paketverluste bei Last
- **MitM** Methode sich ungewollt in die Kommunikation zweier Endpunkte zu platzieren, hoher Paketverlust
- **Inline** Das Gerät, welches den Datenverkehr mitschneidet, ist gleichzeitig auch Router, sodass der gesamte Datenverkehr über diesen geht. Alternativ kann auch eine transparente Bridge eingesetzt werden, allerdings besteht dann nicht mehr die Möglichkeit in den Datenverkehr einzugreifen, ist dafür aber auch nicht für einen Dritten festzustellen, dass sie existiert.
- **Netzwerk-TAP** Dedizierte Hardware, welche wie die transparente Bridge nicht festzustellen ist und ähnlich wie ein Mirrorport funktioniert, jedoch ohne Paketverluste

Die vorgestellten Methoden eignen sich nicht alle, um Netzwerkdatenverkehr zu überwachen. Außerdem bestehen speziell aus der IT-forensischen Sicht Unterschiede. Ein Mirrorport durch die Konfiguration eines Switches jederzeit angelegt werden, ein Inline Router impliziert jedoch ggf. Änderungen am lokalen Routing was möglicherweise Ausfallzeiten nach sich zieht. Geht man davon aus, dass sich ein Angreifer im Netzwerk befindet, welcher bereits aktiv den Netzwerkverkehr beobachtet, wären größere Ausfälle im Routing oder neue Gateways auffällig. Auch eignen sich nicht alle Methoden gleich gut, um die mitgeschnittenen Pakete sinnvoll analysieren zu können. Werden Informationen im Paket oder dessen Datenfluss verändert, spricht man von einer Kontamination. Bei einem Router wäre dies z. B. durch die Network Address Translation (NAT), da das Paket nicht mehr den originalen Sender enthält. Von einer deutlich stärkere Kontamination kann man bei Man-in-the-Middle (MitM) sprechen, da z. B. durch Manipulation des ARP-Cache des Opfers der Paketfluss manipuliert wird, sodass dieser über den Angreifer geht. Dieser ersetzt zusätzlich Quell-/Ziel-IP oder nimmt innerhalb des Pakets weitere Manipulationen vor. In der nachfolgenden Tabelle 2 werden IT-forensische Implikationen unterschiedlicher

Methoden des *sniffings* im Netzwerk noch einmal speziell aus dieser Perspektive betrachtet[14].

Tabelle 2: IT-forensische Perspektive auf unterschiedliche Sniffing-Verfahren

Verfahren	Sniffer detektierbar für Dritte	Kontamination	Verbindungsunterbrechung bei Installation	Paketverluste
Hub	Ja	Gering	Kurz	Bei Last
Mirrorport	Nein	Gering	Nein	Bei Last
MitM	Ja	Ja	Ja	Ja
Inline	Router: Ja Bridge: Nein	Router: Gering Bridge: Nein	Router: Ja Bridge: Kurz	Nein
Netzwerk-TAP	Nein	Nein	Kurz	Nein

Anhand der Zusammenfassung aus Tabelle 2 lässt sich feststellen, dass die sinnvollsten Methoden zum Mitschneiden und analysieren von Netzwerkdatenverkehr Mirrorports, Inline oder Netzwerk-TAPs sind. Hier entscheidet sich die Methode eher nach dem konkret verfolgten Usecase. Für eine kurzfristige Fehlersuche ist ein Mirrorport schnell konfiguriert. Soll der Netzwerkverkehr durch eine NIDS überwacht werden, passiert dies eher Inline oder als gesonderte Netzwerk-TAP. Hier spielt vor allem der Paketverlust bei hoher Netzwerklast eine Rolle. Außerdem handelt es sich bei den Paketen eines Mirrorports immer um eine explizit angelegte Kopie durch den Switch. Hierdurch kommt es zu unterschieden im Timing, sodass Analysen über Antwortszeiten oder ähnliches verfälscht werden. Hinzu kommt, dass bei Mirrorports defekte Pakete verworfen werden, sie werden also niemals ausgeleitet. Zuletzt ist außerdem relevant, dass den Paketen die Informationen über Virtual Local Area Network (VLAN) verloren gehen, da diese nicht an die Paketkopie angefügt werden[14].

2.2.3 IDS/IPS

Ab diesem Kapitel wird NIDS vereinfachend als Synonym für IDS-/IPS verwendet. Obwohl eine Network Intrusion Prevention System (NIPS) erweiterte Funktionen

bietet, werden NIDS und NIPS hier synonym behandelt. Sollte eine Intervention durch eine NIPS relevant sein, wird dies im Text ausdrücklich erwähnt.

Das Ziel einer NIDS besteht darin, Angriffe, die Ausnutzung von Schwachstellen oder Sicherheitsverletzungen zu detektieren. Im Kern hat sich an einer abstrakten Beschreibung der Arbeits- und Funktionsweise seit 2005[27, S. 207 ff.] nichts geändert.

Vor allem praktische Aspekte und technische Details werden in Abschnitt 3.1 näher erläutert. Um Dopplungen zu vermeiden, wird in Abbildung 3 zunächst eine NIDS grafisch dargestellt und anschließend die Grundzüge einer NIDS erläutert werden.

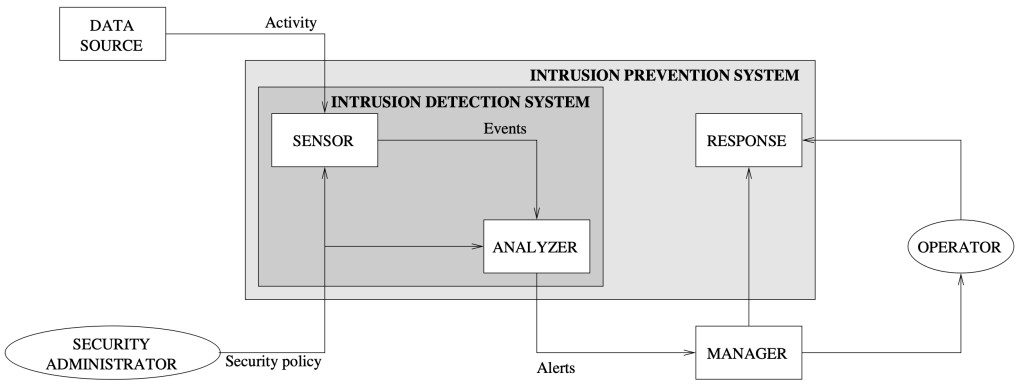

Abbildung 3: Grafische Darstellung der in Unterabschnitt 2.2.3 erläuterten Bestandteile[27, S. 208]

Intrusion Detection Sensors

Eine NIDS verfügt über eine Sensorkomponente. Diese dient der Überwachung der Netzwerkaktivität und bereitet die Daten so vor, dass diese innerhalb der NIDS weiterverarbeitet werden können.

Als weitere Funktion liegt ein Analysemodul vor, welches die Ereignisse der Sensorkomponente untersuchen und ggf. einen Alarm für detektiertes maliziöses verhalten auslösen kann.

Eine Dritte Komponente erhält den Alarm des Analysemoduls und stellt die Informationen einem Analysten bereit. Je nach technischer Implementation können die drei erwähnten Funktionen auch durch ein einzelnes Modul erfüllt werden.

Datenquellen

IDS können sowohl als Host-based intrusion detection system (HIDS) als auch NIDS oder in hybriden Mischformen vorkommen. Bei einer IDS handelt es sich um einen generischen Sammelbegriff, unter welchem HIDS und NIDS fallen bzw. konkretere Ausprägungen dessen sind. HIDS sind nicht mit AV- oder EDR-Lösungen zu vergleichen, hierbei spielt vor allem auch die operative Ebene über Aktivitäten auf dem System eine Rolle. Das wäre die Vorstufe zu einem auf dem System erkannten Virus o. ä. darstellen.

Bei der NIDS verhält sich das etwas anders. Im Gegensatz zu Computern oder Servern existiert im Netzwerk und bei einer NIDS kein wirklich definierter Zustand. Netzwerkdaten sind im permanenten Fluss, und Gefahren müssen zur Laufzeit und nach Möglichkeit in Echtzeit erkannt werden. Diese Arbeit konzentriert sich im Speziellen auf NIDS.

Methoden zur Erkennung

Das NIDS benötigt unterschiedliche Detektionsmechanismen, um Gefahren zu erkennen. Neben der Erkennung von bekannten Angriffsmustern erkennt es auch eine Abweichung vom Normalverhalten als Anomalie.

Informationsmanagement

Erfasste oder gespeicherte Informationen werden zu einer zentralen Stelle weitergeleitet, bei welcher diese ggf. auch weiterverarbeitet werden. Die dafür nötigen Grundlagen ist die Möglichkeit, die auf der NIDS erfassten Informationen zu Filtern, zu Aggregieren und vor allem auch zu normalisieren. Eine NIDS verfügt auch Schnittstellen zur Echtzeitüberwachung, sodass z. B. der Datenverkehr mitgeschnitten oder in Echtzeit betrachtet werden kann.

Alarme sind mit einer Priorisierung oder Kritikalität versehen, um diese durch einen Analysten sinnvoll bearbeiten zu können.

Eine NIDS stellt für Alarme relevante Kontextinformationen bereit, nach Möglichkeit werden die Informationen auch an der zentralen Stelle mit weiteren Details angereichert werden.

2.3 Cybersecurity

Im Folgenden Unterabschnitt 2.3.1 wird es spezifischer um existierende Gefahren gehen. Danach werden in Unterabschnitt 2.3.2 verfolgte Zwecke den jeweiligen Malware-Typen zugeordnet. Zuletzt wird es in Unterabschnitt 2.3.3 konkreter um C2 Datenverkehr gehen und wie dieser verschleiert wird.

2.3.1 Arten von Gefahren

Gefahren sind unterschiedlich ausgeprägt, haben oft unterschiedliche Ursachen und meist auch unterschiedlich schwer in ihrer Auswirkung. MITRE stellt mit dem `ATT&CK` Framework einen Rahmen dar, um *Tactics* (Taktiken) und *Techniques* (Techniken) zu kategorisieren. Diese Art der Darstellungsweise wird auch beispielsweise dazu verwendet, um komplexere Cyberangriffe darzustellen. Dies ist vor allem für Cybersicherheitskräfte von Relevanz, da es häufig darum geht, bestimmte Gefahren abstrakter abwehren oder geeignete Gegenmaßnahmen treffen zu können. Um ein konkretes Beispiel zu nennen, ist für die Abwehr eines Phishing-Angriff irrelevant, ob sich in der E-Mail ein sogenannter Phishing-Link befindet oder sich dieser in einem Anhang verbirgt, welcher aufgerufen werden soll. Abstraktere Gegenmaßnahmen wie beispielsweise ein Webproxy, der den Datenverkehr überwacht und die aufgerufene Uniform Resource Locator (URL) detektiert, hat den Angriff in diesem Moment unterbunden. Außerdem wird durch das Unterbinden einer bestimmten Technik meistens andere Techniken be- oder verhindert. Im Folgenden werden Begriffe der Literatur[58] erläutert und nach Möglichkeit dem *MITRE ATT&CK-Framework*[23] zugeordnet.

Hackerangriffe

Dies ist ein Oberbegriff dafür, dass unberechtigte Personen sich Zugriff zum Netzwerk verschaffen wollen. Neben unterschiedlichen Motivationen (Anerkennung, finanzielle Motivation, Neugierde oder Zerstörungswut) unterscheidet sich auch, wer für einen Angriff oder Angriffsversuch verantwortlich ist. Neben Cybersicherheitsexperten wie Penetrationstester kommen hier auch IT-Terroristen, meist in Verbindung mit politischen- oder gesellschaftlichen Zielen, staatlich-finanzierten Hacker, meist mit politischen Zielen wie z. B. *Cyberwar* auch Hacker aus dem Bereich der organisierten Kriminalität infrage[58, S. 43 f.]. Die spezielle Ausprägung des Hackerangriffs in Form von Ransomware aus dem Bereich der organisierten Kriminalität wird in Unterunterabschnitt 2.3.1 näher ausgeführt werden. Der genaue

Sinn und Zweck eines Hackerangriffs lässt sich manchmal erst bei einer näheren IT-forensischen Untersuchung bestimmen.

Cyberkriminalität und Ransomware

Bei Cyberkriminalität handelt es sich wie bei dem Hackerangriff um einen Oberbegriff. Cyberkriminalität ist oft, aber nicht unbedingt zwangsläufig, durch eine gewisses finanzielles Interesse im Bereich der organisierten Kriminalität geprägt[11]. *Ransomware* ist einer der dominanteren Vertreter der Cyberkriminalität. Die bei *Ransomware* bezahlten Lösegeldsummen lagen in 2023 bei einem Allzeithoch von 1,1 Millarde Dollar, verglichen dazu 567 Millionen Dollar in 2022 und 983 Millionen Dollar in 2021[56]. Der bei Ransomware verfolgte Modus-Operandi sowie der zugrunde liegende Verschlüsselungsmalware werden in Unterunterabschnitt 2.3.1 und Unterunterabschnitt 2.3.2 näher erläutert. *Ransomware* umfasst z. B. Gruppen wie `LockBit`, die Ransomware as a Service (RaaS) zu einem Geschäftsmodell entwickelt haben[3]. Innerhalb dieses Geschäftsmodells ist es nicht mehr nötig, dass ein potenzieller Angreifer über tiefer gehendes Wissen in IT-Security verfügen muss. Alle nötigen Dokumente und Tools werden zur Verfügung gestellt, wird die Lösegeldsumme (*ransom*) bezahlt, erhält der RaaS-Anbieter einen Teil der Summe. Zum Teil fällt Cyberkriminalität in Verbindung mit staatlicher Unterstützung auch in den Bereich des *Cyberwar*, wie der wahrscheinlich berühmteste Vertreter `Stuxnet`, bei welchem iranische Urananreicherungsanlagen physischen beschädigt wurden[58, S. 23 f.].

Social Engineering und Phishing

Phishing ist die Täuschung einer Person, die dazu verleitet werden soll, ihre Zugangsdaten preiszugeben oder maliziöse Anhänge auszuführen. Hierfür wird z. B. vorgegeben, sich mit einem Dienst o.ä. zu authentifizieren, wobei es sich hierbei um Nachbildungen eines legitimen Dienstes handelt. Social Engineering unterstützt Phishing dahingehend, den Angriff gezielter zu gestalte, um mehr Authentizität vorzutäuschen. Hierbei wird auch von *Spear-Phishing* gesprochen[58, S. 197 f.]. Die Taktik Phishing fällt unter die Technik *Initial Access*[21], welche den sprichwörtlichen Fuß in der Tür darstellt. Sobald legitime Zugangsdaten vorliegen, kommt es in der Regel danach zu einem unautorisierte Zugriff, siehe auch Unterunterabschnitt 2.3.1. Je nachdem, welches Ziel ein Angreifer erfolgt, werden die erlangten Zugangsdaten zunächst veräußert, ohne, dass weitere Aktivitäten erfolgen.

Unautorisierter Zugriff

Hierunter können unterschiedliche Techniken gefasst werden. Grundsätzlich geht es darum, dass jemand, egal ob Angreifer oder Mitarbeiter sich Zugriff zu einem System oder ein Privileg verschafft, zu dem normalerweise kein Zugriff besteht. In der Taktik *Initial Access*[21] fällt hierunter der Zugriff mit den durch Phishing erlangte Zugangsdaten auf im Internet erreichbare Anwendungen oder Remote Services wie Virtual Private Network (VPN) oder virtuelle Desktops. Die Taktik *Privilege Escalation*[25] spielt sich ausschließlich um den Punkt der Erweiterung der verfügbaren Privilegien, sodass z. B. ein höher privilegiertes Benutzerkonto verwendet wird, welches erweiterte Rechte besitzt.

Sniffing und MitM

Sniffing und MitM gehen oftmals gemeinsam einher. Sniffing beschreibt das mitlesen des Datenverkehrs, MitM wird es dann genannt, wenn der Datenverkehr durch den Angreifer hindurch geht. Sniffing erfolgt Grundsätzlich passiv, es wird der lokale Datenverkehr mitgeschnitten, um beispielsweise durch Broadcasts Informationen über Teilnehmer im LAN zu erhalten. Bei MitM hat der Angreifer die Möglichkeit den Datenverkehr zwischen zwei Systemen mit lesen zu können. Da der Netzwerkverkehr in diesem Fall über den Angreifer geht, hat dieser auch zusätzlich die Möglichkeit diesen z. B. zu entschlüsseln, um vertrauliche Informationen zu erlangen oder den Inhalt zu manipulieren. Beides setzt voraus, dass sich ein Angreifer bereits innerhalb des Netzwerkes befindet. *Sniffing* fällt als Technik unter die beiden Taktiken *Credential Access*[16] und *Discovery*[18]. *MitM* fällt als Technik unter die beiden Taktiken *Credential Access*[16] und *Collection*.

Sicherheitslücken / Schwachstellen

Hierbei handelt es sich um Schwachstellen in einer Anwendung oder dem Betriebssystem selber. Diese können ausgenutzt werden um, in Abhängigkeit der konkreten Schwachstelle, Aktivitäten außerhalb der eigentlichen Sicherheitsvorkehrungen ausführen zu können. Dies kann von *Information Leak*, der ungewollten Ausgabe von Daten, bis hin zur *Code Execution*, dem Ausführen von beliebigem Code, reichen[58, S. 6 f.]. Die Ausnutzung solcher *Exploits* als Technik ist mehreren Taktiken zugeordnet: *Initial Access*[21], *Execution*[19], *Privilege Escalation*[25], *Defense Evasion*[17], *Credential Access*[16] und *Lateral Movement*[22]. Sicherheitslücken sind gleichzeitig ein gutes Beispiel dafür, dass das proaktive Suchen nach Schwachstellen sowie das

zeitnahe schließen neu bekanntgewordener Schwachstellen an mehreren Stellen einen möglichen Angriff erschweren zu können.

Ransomware

Mit Ransomware wird umgangssprachlich ein Vorfall bezeichnet, in welchem meist Firmen, Opfer eines eben solchen Angriffs werden. Ransomware kommt im Kontext der organisierten Kriminalität vor, um durch eine Lösegelderpressung Geld zu erwirtschaften. Bei einem Ransomware-Angriff werden großflächig Nutzdaten wie Office-Dokumente, Datenbanken oder die gesamte Festplatte von Virtuelle Maschines (VMs) verschlüsselt. Oft werden Backups gezielt gesucht und auch verschlüsselt oder gelöscht, um eine Rücksicherung der Daten zu verhindern. Häufig geht damit einher, dass ein Subset der Daten vor der Verschlüsselung exfiltriert worden sind. Dies dient dem Zweck, gleichzeitig mit einer Veröffentlichung der Geschäftsgeheimnisse drohen zu können, sollte das Lösegeld, die *ransom*, nicht bezahlt werden. Man spricht hierbei auch von *Double Extortion*, also einer doppelten Erpressung in Bezug auf die verschlüsselten Daten sowie der Veröffentlichung der exfiltrierten Daten[11]. Die zugrunde liegende Malware vom Typ Ransomware wird in Unterunterabschnitt 2.3.2 näher beschrieben.

2.3.2 Typen von Malware

Malware ist ein Oberbegriff für unterschiedliche Ausprägungen von Schadsoftware wie z. B. Viren, Würmer, Trojaner oder Spyware. Wie in Unterabschnitt 2.3.1 beschrieben, fließt die Grenze zwischen unterschiedlichen Malware-Typen, sodass eine Beschreibung aufgrund der Funktion der Malware nicht immer zielführend ist. Malware besitzen unterschiedliche Funktionen, sodass die Grenzen zwischen der zuvor beschriebenen Sichtweisen fließend sind. In diesem Abschnitt wird ein Überblick darüber gegeben, welches Ziel oder Zweck mit der jeweiligen Malware verfolgt wird.

Downloader / Dropper

Downloader oder Dropper sind meist die erste Stufe, die zur Infektion eines Systems führt. Bei diesem Mechanismus handelt es sich oft um sehr einfache Skripte, welche die nächste Stufe der Malware herunterladen und ausführen. Dies ist für Angreifer vorteilhaft, da sich Skripte relativ schnell neu obfuskieren, also verschleiern, lassen

und so ein neuer Hashwert der Datei entsteht. Hinzu kommt, dass die darin ausgeführten Befehle selten bis gar nicht als schadhaft klassifiziert werden, was z. B. eine Zustellung über E-Mail vereinfacht. Vorteilhaft ist außerdem, dass sich diese Skripte Living Off The Land Binaries (LOLBins) bedienen, also Anwendungen oder Programme, die auf einer Vielzahl von Systemen standardmäßig vorinstalliert sind. Hierzu gehört auch die Windows Kommandozeileneingabe (`cmd`) zur Ausführung von Befehlen oder die Zertifikatsverwaltung (*certutil*), welche zum Download von Dateien missbraucht werden kann. Welche Malware heruntergeladen werden soll, lässt sich Anhand des Downloaders nicht feststellen, da diese oder abgewandelte Formen des beschriebenen Mechanismus häufig zum Einsatz kommt. Dropper persistieren sich typischerweise nicht auf dem infizierten System. [59, S. 143 ff.]

Cryptominer

Cryptominer nutzen die Rechenkapazität des infizierten Systems, um Kryptowährung zu schürfen. Bei Blockchain-basierten Kryptowährungen validiert das System durch mathematische Berechnungen in der Blockchain befindliche Transaktionen. Hierfür wird eine Entlohnung ausgeschüttet. Da die mathematischen Berechnungen häufig rechenintensiv sind, entsteht hierdurch eine höhere CPU-/GPU-Auslastung auf dem System, somit findet eine unrechtmäßige Nutzung der Rechenkapazität statt. Hier geht eine höhere Stromaufnahme mit einher, sodass von einem Energiediebstahl gesprochen werden kann. Cryptominer persistieren sich auf dem infizierten Windows-System über z. B. geplante Aufgaben oder Dienste. [43]

Information Stealer

Information Stealer stehlen und exfiltrieren Informationen von dem System, auf welchem sie sich befinden. Eine gängige Alternativbezeichnung wäre auch Spyware. Zu den Informationen die gestohlen werden gehören Zugangsdaten, Kreditkarteninformationen oder Cookies zur Authentifizierung aus dem Browser, lokale Passwortsafes wie z. B. KeePass oder E-Mails. Information Stealer persistieren sich häufig, um eine regelmäßige und somit längerfristige Exfiltration von Daten zu gewährleisten. [12]

Bots

Bots sind Teilnehmer in einem Botnetz, sie werden auch manchmal als `Zombies` bezeichnet. Bots führen die Befehle aus, die ihnen durch das Botnetz bzw. dem

Botmaster mitgeteilt werden. Abstrakt ausgedrückt kann der Botmaster über die Ressourcen der Bots verfügen. Häufig geht es darum, gemeinsam einen DDoS Angriff auszuführen. Solche Angriffe werden dazu verwendet, Webseiten oder Dienste im Internet zu überlasten, sodass diese nicht mehr erreichbar sind oder eine hohe Antwortzeit hat. Bots halten C2 Kommunikation und persistieren sich, um eine höhere Verfügbarkeit im Botnetz selber zu gewährleisten. [59, S. 144] [58, S. 310 ff., 817 f.]

Initial Access Malware

Initial Access Malware, Alternativ auch als Trojaner, Remote Access Trojan (RAT) oder Backdoor zu bezeichnen, ist ein Typ Malware, welche den digitalen *Fuß in der Tür* des Netzwerks darstellt. Sie besitzen häufig auch Eigenschaften eines Information Stealers und sammeln außerdem speziellere Informationen über das lokal erreichbare Netzwerk (LAN). Hierzu gehört auch z. B. Informationen über die Domäne oder IP-Adressinformationen. Initial Access Malware führt ähnlich wie ein Bot Befehle aus, hierzu wird C2-Kommunikation gehalten. Sie spielt häufig in *Ransomware*-Vorfällen eine Rolle[58, S. 7 ff.] und persistiert sie sich, um einen längerfristigen Zugriff auf das System zu erhalten[12][59, S. 144 ff.].

Ransomware

Bei Ransomware handelt es sich wie bereits in Unterunterabschnitt 2.3.1 beschrieben um die eigentliche Malware, welche die Verschlüsselung der Daten ausführt. Aus der Perspektive von Malware handelt es sich bei Ransomware um die eigentliche Verschlüsselungsmalware (Technik *Data Encrypted for Impact*), welche der Taktik *Impact*[20], also der Auswirkung zuzuordnen ist. Moderne Ransomware verschlüsselt je nach Dateigröße nur Teilbereiche der Datei, was den kompletten Vorgang der Verschlüsselung um ein vielfaches beschleunigt. Dies soll zum einen verhindern, mittels Statistik über Dateiänderungen die Verschlüsselung zu erkennen aber auch zum anderen die Möglichkeit nehmen, aufgrund der hohen Geschwindigkeit der Verschlüsselung durch z. B. abschalten des Systems intervenieren zu können[54].

Ransomware persistiert sich nicht und löscht sich in der Regel nach der Ausführung.

Wiper

Wiper verfolgen einen etwas anderen Ansatz als die in Unterunterabschnitt 2.3.2 beschriebene Ransomware, sie Löschen die Nutzdaten anstatt sie zu verschlüsseln. Zum Teil werden die Blöcke des Speichermediums genullt, was zur Folge hat, dass die darauf befindlichen Daten auch IT-forensisch nicht mehr wiederherzustellen sind. Die hierbei verfolgten Ziele sind meist destruktiver und verfolgen eher den Zweck einer digitalen Kriegsführung wie z. B. 2022, wo der Wiper *AcidRain* Modems des Herstellers Viasat unbrauchbar machte. Dies wiederum stand im Kontext des Russisch-Ukrainischen Kriegs[40].

Wiper persistiert sich nicht und löscht sich in der Regel nach der Ausführung. Abschließend sei angemerkt, dass auch nach dieser Aufteilung hybride Mischformen von Ransomware und Wipern existieren. Um z. B. nach einer Ransomware-Ausführung die Wiederherstellung von Daten zu verhindern, werden Teile gezielt durch Wiper-Funktionen gelöscht.

2.3.3 Command and Control

Wie in Unterabschnitt 2.3.2 beschrieben, halten einige Typen von Malware C2 Kommunikation. Da die konkrete Einbettung in Kapitel 5 ausführlich beschrieben wird, erfolgt an dieser Stelle eine Übersicht sowie eine Darstellung der Vor- und Nachteile des jeweiligen Protokolls.

Neben den nachfolgend aufgeführten und die mit gängigsten Protokollen, existieren auch weitere Möglichkeiten, die C2-Kommunikation zu verschleiern. Grundsätzlich eignen sich viele Protokolle, wenn bestimmte Voraussetzungen gegeben sind. So sollte für den gewünschten Kanal eine native Windows-Application Programming Interfaces (API) existieren, sodass Anfragen oder Antworten über diese getunnelt werden können. Das Protokoll sollte in einer Bi-direktionalen Kommunikation Daten senden können. Falls nicht, sollte es sich um ein Peer-to-Peer (P2P) Protokoll handeln. In diesem Kontext ist es auch relevant, unter welchen Privilegien die API bzw. das Protokoll genutzt werden kann. Hier ist der User-Kontext am günstigsten, da keine vorgelagerte Privilegieneskalation passieren muss[44].

DNS

DNS, siehe Unterabschnitt 2.1.2, kann dazu missbraucht werden, die C2-Kommunikation in den DNS-Queries und DNS-Responses zu verbergen. Konsequenterweise muss der Angreifer hierfür über eine Domäne verfügen, die für diese Zwecke missbraucht werden kann. Damit die DNS-Request zu dem durch den Angreifer kontrollierten C2-Server gelangen, ist ein NS-Eintrag für die Domäne erforderlich. Für größere Datenmengen kann der RR `TXT` missbraucht werden, da sich hier einfacher größere Datenmengen einbetten lassen.

Als *Defense Evasion* Mechanismus kommen die folgenden Methoden zum Einsatz: Mittels eines Domain Generation Algorithm (DGA) werden randomisiert Domänennamen generiert, was eine Erkennung oder die Blockierung einer einzelnen Domäne erschwert. Als weitere Methode wird mittels *Fast Flux* DNS regelmäßig die IP-Adresse ausgewechselt, sodass eine Zuordnung zu einer gewissen Geolokation erschwert wird. Mittels *Domain Shadowing* werden Subdomänen unter einer legitimen Domäne erstellt, welche speziell für die C2-Kommunikation genutzt wird. Besitzt die legitime Domäne eine gute Reputation, vererben die Subdomänen diese, was eine generelle Erkennung erschwert. [50] [72]

Neben der Nutzung einer Domäne bzw. eines autoritativen NS, welche auf der Auflösung innerhalb des hierarchischen DNS-Systems basiert, besteht noch die Möglichkeit, eine Punkt-zu-Punkt-Verbindung über DNS aufzubauen. Die Funktionsweise ist hier weitestgehend identisch, jedoch deutlich auffälliger. Direkte DNS-Anfragen werden ggf. schon durch die Firewall unterdrückt, sodass keine Kommunikation zustande kommen kann.

Die Nutzung von DNS als C2-Protokoll hat den Vorteil, dass DNS ein zwangsläufig benötigtes Protokoll ist, was eher selten durch eine Firewall gefiltert wird. Außerdem besteht durch die hierarchische Auflösung von DNS-Anfragen die indirekte Möglichkeit, aus stark eingeschränkten Netzbereichen ein-/ausgehend kommunizieren zu können. Der Nachteil liegt darin, dass DNS kein Protokoll ist, welches die Nutzdaten verschlüsselt, eine NIDS also das gesamte Paket überprüfen kann. Die für die C2-Kommunikation verwendeten Daten sind jedoch oftmals obfuskiert oder verschlüsselt, was bei der Analyse eines solchen Pakets auffällig ist.

HTTP / HTTPS

HTTP, siehe Unterabschnitt 2.1.3, und HTTPS, siehe Unterabschnitt 2.1.4, können dazu missbraucht werden, die C2-Kommunikation in HTTP/HTTPS-Anfragen und HTTP/HTTPS-Antworten zu verbergen[47].

Die Nutzung von HTTP und HTTPS als C2-Protokoll hat den Vorteil, dass diese selten durch eine Firewall eingeschränktes Protokoll ist, welches für den Aufruf von Webseite genutzt wird. Vor allem Clients können häufig Internetseiten direkt aufrufen, was von Vorteil für die Nutzung von C2 via HTTP ist.

Der große Vorteil von HTTPS gegenüber von HTTP liegt darin, dass HTTPS die Nutzdaten verschlüsselt, eine NIDS also nicht das gesamte Paket überprüfen kann. Hier müsste eine Entschlüsselung des Datenverkehrs passieren, bevor die eigentlichen Nutzdaten überprüft werden können.

Der Nachteil von HTTP liegt darin, dass die Nutzdaten unverschlüsselt sind, eine NIDS also das gesamte Paket überprüfen kann. Außerdem ist die Nutzung von HTTP im Vergleich zu HTTPS eher auffällig. Die für die C2-Kommunikation verwendeten Daten sind jedoch oftmals obfuskiert oder verschlüsselt, was bei der Analyse eines solchen Pakets auffällig ist.

ICMP

Internet Control Message Protocol (ICMP) kann dazu missbraucht werden, die C2-Kommunikation in ICMP Echo-Request und Echo-Reply zu verbergen[52].

Die Nutzung von ICMP hat den Vorteil, dass es sich hierbei um ein häufig durch Administratoren verwendetes Protokoll handelt, welches auch gewisse Kontrollinformationen im Netzwerk austauscht. Es ist eher seltener explizit durch eine Firewall gefiltert. Der Nachteil liegt darin, dass ICMP die Nutzdaten nicht verschlüsselt, eine NIDS also das gesamte Paket überprüfen kann. Die für die C2-Kommunikation verwendeten Daten sind jedoch oftmals obfuskiert oder verschlüsselt, was bei der Analyse eines solchen Pakets auffällig ist.

Server Message Block (SMB)

SMB kann dazu missbraucht werden, die C2-Kommunikation in den *Named Pipes* zu verbergen. *Named Pipes* werden bei SMB unter anderem dazu genutzt, um einen Austausch von Daten zwischen Server und Client im Rahmen der Interprozess-

kommunikation zu ermöglichen. Hierbei handelt es sich um eine Punkt-zu-Punkt-Verbindung[48].

Der Vorteil liegt darin, dass SMB innerhalb eines Netzwerkes häufig freigegeben ist, sodass Teilnehmer auf z. B. SMB-Freigaben zugreifen können. Ein großer Nachteil ist, dass Ein-/Ausgehender SMB-Datenverkehr häufig an einer Firewall blockiert wird, sodass diese Methode eigentlich nur innerhalb eines Netzwerks verwendet werden kann.

Mutual TLS (mTLS) und VPN

mTLS und VPN kann analog zu HTTPS dazu missbraucht werden, die C2-Kommunikation zu verbergen. Anders als die bisher vorgestellten Protokolle kann man an dieser Stelle nicht mehr von einem verschleiern sprechen, da die Protokolle entweder die Nutzdaten nicht verschlüsseln oder aber eine Entschlüsselung zur Überprüfung möglich war. Dies ist bei mTLS oder anderen VPN-Protokollen anders. Die Nutzdaten sind verschlüsselt und können auch nicht zur Untersuchung entschlüsselt werden. Eine Untersuchung von Sophos zeigte für das erste Quartal 2021 bereits, dass 46 % der festgestellten Malware C2 via TLS kommunizierte.

Der Vorteil liegt darin, dass keine Überwachung der C2-Kommunikation erfolgen kann. Der Nachteil liegt darin, dass je nach Regelwerk der Firewall ausgehende VPN-Verbindungen nicht erlaubt sind.

Sonstige

Die nachstehende Auflistung umfasst eher unkonventionelle Methoden zur Umsetzung von C2-Kommunikation. Diese Methoden erfüllen möglicherweise nicht die üblichen Anforderungen für eine ordnungsgemäße Implementierung und stellen eine eher exotische Sammlung dar, die nicht weiter erläutert wird. C2 via Gmail[37], C2 via OneDrive oder Dropbox[33] oder C2 via Discord[34].

2.4 C2-Frameworks

Im folgenden Unterabschnitt 2.4.1 wird es um die Begriffsdefinition und gängige Begrifflichkeiten von C2-Frameworks gehen. Zuletzt wird es in Unterabschnitt 2.4.2 um den Funktionsumfang gängiger Produkte oder Open-Source-Projekte gehen, die zu den C2-Frameworks zählen.

2.4.1 Begriffsdefinition und Begrifflichkeiten

C2-Frameworks sind das Handwerkszeug von Penetrationstestern. Aufgrund ihres Funktionsumfanges und dem inzwischen hohen Grad an Automatisierung werden diese häufig auch zu illegitimen Zwecken verwendet. Im Regelfall passiert dies im Rahmen von Ransomware-Vorfällen im Kontext der organisierten Kriminalität.

Eine lehrbuchartige Definition zu C2-Frameworks gibt es nicht, im Rückgriff auf übliche Funktionen, welche durch ein Framework bereitgestellt werden, geben eine eher praktische Definition. Im Kern ist ein C2-Framework dazu da, Remote auf zu steuernde Systeme zugreifen zu können. Dies passiert in einem Client-Server-Modell, wobei der Operator mit dem C2-Server auf das System zugreift. Die Erstellung des Beacons, mitunter als Implant bezeichnet und der Konfiguration sind hierbei einer der Hauptbestandteile eines C2-Frameworks[51].

Neben der Fernsteuerung bieten Frameworks unterschiedliche Funktionen, welche im folgenden sowie Unterabschnitt 2.4.2 näher erläutert werden.

Stager und Loader

Stager und Loader sind zwei Komponenten ganz zu Beginn einer Infizierung eines Systems. Der Stager ist in der Regel minimalistischer Code und dient meist ausschließlich dazu, eine Verbindung zum C2-Server herzustellen um die nächste Stufe, also den Loader, herunterzuladen und auszuführen. Der Loader erfüllt weitere, meist komplexere Zwecke, wie z. B. lokale Schutzmaßnahmen außer Kraft zu setzen oder diese zu umgehen. Danach wird durch den Loader der eigentliche Payload heruntergeladen und ausgeführt[59, S. 143 ff.]. Manchmal erfüllt auch eine einzelne Stufe, also ein hybrid aus Stager und Loader, die Aufgaben beider Komponente.

Beacon

Beacon oder auch Implant ist das Stück Software, welches die Kommunikation zum C2-Server hält und die von dort empfangenen Instruktionen ausführt. Das Beacon wird mit einer Konfiguration versehen, welches sein Verhalten steuert.

So kann z. B. festgelegt werden, in welchem Abstand es sich beim C2-Server melden soll, aber auch das zu verwendende Protokoll oder den zu verwendenden *Fallback-Channel*. Sollte beispielsweise C2 via VPN nicht funktionieren, kann auf ein anderes Protokoll zurückgegriffen werden. [45]

Persistenz

Persistenz bezeichnet nach MITRE `ATT&CK` die Taktik (siehe auch Unterabschnitt 2.3.1) von Angreifern, einen permanenten Zugriff auf das System zu haben. Permanent bedeutet in diesem Falle nicht, dass das System nicht mehr ausgeschaltet werden kann, vielmehr, dass auch nach einem Neustart der Beacon wieder Kontakt zum C2-Server herstellt. Dies auch idealerweise unabhängig vom angemeldeten Benutzer. Auf Windows-Systemen werden dazu häufig *Run*-Keys in der Registry angelegt, Dienste installiert oder *Scheduled Tasks* angelegt, welche den Beacon wieder ausführen[24].

Post-Exploitation

Post-Exploitation ist die Phase eines Angriffs, welche nach der erfolgreichen, initialen Kompromittierung eines Systems kommt. Das ist der Zeitpunkt, nach welchem sich der Beacon erfolgreich beim C2-Server melden konnte. Würde man diesen Begriff nach MITRE `ATT&CK` einzuordnen versuchen, stellt man fest, dass die Aktivitäten unter verschiedene Taktiken fallen. In dieser Phase werden meist zu Beginn Informationen über das Netzwerk gesammelt, *Discovery*[18], die verfügbaren Privilegien erweitert, *Privilege Escalation*[25] und sich innerhalb des Netzwerkes weiterbewegt, *Lateral Movement*[22]. Besonders die kommerziellen Tools haben hier einen technologischen Fortschritt gegenüber der Open Source C2-Frameworks, mehr dazu in Unterabschnitt 2.4.2

Lateral Movement

Lateral Movement bezeichnet die Taktik, sich innerhalb des Netzwerkes fortzubewegen. Dies kann z. B. über Remote Desktop Protocol (RDP) erfolgen, indem man sich schlicht zu den Zielsystemen verbindet. RDP selber ist eines der häufigst verwendeten Methoden, wenn es darum geht, sich Remote mit einem Server zu verbinden. Diese Methode wird selten gefiltert, oder großzügig freigegeben, da die Administratoren der Infrastruktur diese Zugriffe selber benötigen. Etwas unauffälliger, jedoch nicht ähnlich Funktional kann auch auf dem Zielsystem ein Dienst installiert werden, welcher wiederum den Beacon startet, sodass der Angreifer direkten Zugriff hat. *Lateral Movement* setzt häufig voraus, dass bereits weitere Zugangsdaten oder Passworthashes vorliegen, um sich authentifizieren zu können.[22]

Abgrenzung

Von der im Kern vorhandenen Funktion eines C2-Frameworks, also dem erstellen eines Beacons und der Kommunikation zwischen Beacon und C2-Server unterscheidet sich im ersten Augenblick wenig von RATs oder unterschiedlichen Arten von Shells. Hier wären z. B. *Webshell* oder *Reverse Shell* zu erwähnen, welche häufig bei der Ausnutzung von offenen Schwachstellen von im Internet erreichbaren Servern platziert werden. Diese stellen dabei meist, jedoch nicht immer, den sprichwörtlichen Fuß in der Tür dar.

Tatsächlich ist eine Grenze zwischen einer Shell, einem RAT und einem C2-Beacon fließend. Shells verfügen meist über ein limitiertes Set an Funktionen, und dienen eher dazu, dass nicht immer wieder die offene Schwachstelle ausgenutzt werden muss, um einen einzelnen Befehle auf dem System auszuführen. Erweiterte Funktionen wie z. B. eine komplexere Befehlsausführung via PowerShell lassen diese häufig vermissen und stellen eher die erste Stufe der Kompromittierung dar. Je nach Shell-Variante unterscheidet sich der Kommunikationsfluss nicht von einem C2-Beacon, so kommunizieren z. B. Reverse Shells analog. Andere Shell-Varianten wie z. B. Webshells, welche es aktiv erforderlich machen, dass eine gewisse URL aufgerufen wird, gehören eher nicht dazu. Die Implementation von einfachen Shells ist oftmals auch eher simpel gehalten, sodass komplexere Befehlsketten nicht korrekt übersetzt werden, was wiederum dazu führt, dass der Befehl auf dem Zielsystem nicht erfolgreich ausgeführt werden kann. RATs hingegen unterscheidet relativ wenig von einem C2-Beacon, sodass eigentlich erst nach einer Untersuchung der Malware eine Attributierung passieren kann[67, S. 232 ff.].

Eine Abgrenzung zu Bots oder Botnetzen lässt sich anhand der Funktionsweise beschreiben. Bots werden typischerweise gemeinsam als ein Objekt kontrolliert, sodass z. B. die Masse eine bestimmte Interaktion ausführt. Beacons kommen bei gezielten Angriffen zum Einsatz und werden eher einzeln oder in Gruppen kontrolliert[67, S. 234].

2.4.2 Überblick Funktionsumfang C2-Frameworks

Im Bereich der kommerziellen Anwendungen wären Cobalt Strike von Fortra[46] und Brute Ratel C4 von Dark Vortex[80] zu erwähnen. Im Rahmen dieser Bachelor-Thesis ist Fortra für eine kostenfreie Lizenz von Cobalt Strike angefragt worden, diese lehnten allerdings ab. Team Cymru Threat Research veröffentlichte im Februar

2023 auf X eine Grafik, siehe Abbildung 4, zur Verteilung aktueller Offensive Security Tools (OST).

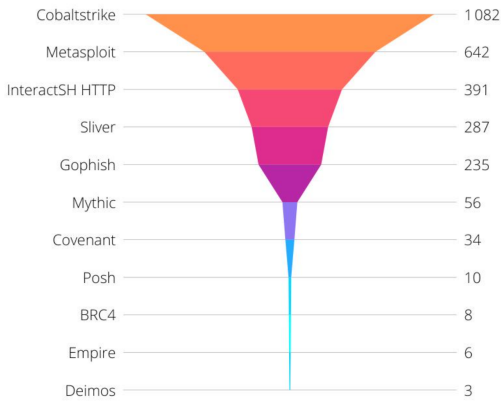

Abbildung 4: Verteilung OST im Februar 2023[74]

Hier wird umso deutlicher, wie intensiv Cobalt Strike für illegitime Zwecke missbraucht wird.

Im Bereich der Open Source Anwendungen wäre Sliver[5] und Havoc[35] zu nennen. Auch erwähnenswert ist auch das OST Metasploit[38], welches eine Sammlung diverser, häufig für Angriffe verwendete Tools, bereitstellt. Außerdem existieren kleinere und weniger aufwändig gestaltete Projekte, welche z. B. die C2-Kommunikation über ein bestimmtes Protokoll umsetzen, der Funktionsumfang jedoch deutlich geringer ist.

Die nachfolgenden Informationen stammen aus den offiziellen Dokumentationen von Sliver[6], Havoc[1] und Cobalt Strike[49].

Sowohl kommerziellen als auch nicht-kommerzielle Tools unterstützen die Generierung von Beacons anhand einer gewissen Konfiguration. So kann das Protokoll gewählt und innerhalb dieses Protokolls gewisse Anpassungen zur *Defense Evasion*[17] vorgenommen werden. Für den Beacon-Timer, manchmal auch *sleep* genannt, also die Zeit, in der der Beacon nicht aktiv mit dem C2-Server kommuniziert, kann von Sekunden bis Stunden variieren. Zusätzlich kann zu diesem Zyklus auch ein Jitter, also eine bewusste zeitliche Abweichung, definiert werden, um eine größere Varianz im Kommunikationsfluss zu erzeugen und somit eine Anomalieerkennung zu erschweren. Um ein weiteres Beispiel zu nennen, kann der zu verwendende `UserAgent` festgelegt werden, um ihn an den durch das System verwendeten `UserAgent` anzupassen. Auch lassen sich zum Teil Aktivitätszeiten definieren, sodass der Beacon

nur im vorgegeben Zeitraum Kontakt zum C2-Server aufbaut. Auch dies dient der Erschwerung einer Anomalieerkennung bzw. zur *Defense Evasion*[17].

Sliver besitzt keine GUI, sondern ist eine Konsolenanwendung, jedoch ist die Ausgabe grafisch gestaltet. Die WebGUI von Cobalt Strike, Brute Ratel C4 und Havoc ähneln sich sehr und ermöglichen nahezu dieselben Interaktionen mit den infizierten Systemen.

Sliver und Havoc besitzen keine Funktionen aus dem Bereich *Post Exploitation*, hier unterscheiden sich die kommerziellen Produkte deutlich. So kann z. B. eine Privilegienerweiterung mittels *Token impersonation* zu `SYSTEM` per Befehl ausgeführt werden. Hierfür wären bei Sliver oder Havoc eigene Tools auf dem infizierten System notwendig.

3 Vergleich Open Source IDS-/IPS-Systeme

In diesem Kapitel wird ein für die im nachfolgenden Kapitel 4 zum Einsatz kommende NIDS ausgewählt.

Zunächst werden in Abschnitt 3.1 die ausgewählten Leistungsmerkmale erläutert, diese in Abschnitt 3.2 auf mögliche Systeme angewendet und abschließend in Abschnitt 3.3 ein Fazit gezogen.

3.1 Merkmale

Die nachfolgend beschriebenen Merkmale, über die eine NIDS verfügen sollte stammen neben der Berufserfahrung des Autors aus den Funktionsbeschreibungen der Fachliteratur von Hervé Debar und Jouni Viinikka[27, S. 207 ff.], Umesh Hodeghatta Rao und Umesha Nayak[59, S. 225 ff.] sowie einem Leitfaden zur Einführung von IDS des BSI[65].

3.1.1 Anwendungsbereich

Ziel ist es, Angriffe, die Ausnutzung von Schwachstellen oder Sicherheitsverletzungen wie Datenuploads zu Erkennen. Hierfür sollte das System mindestens über eine geeignete NIDS-Komponente verfügen. Reine HIDS sind aufgrund der Aufgabengestaltung ausgeschlossen.

3.1.2 Architektur und Ressourcennutzung

Die NIDS überwacht den Netzwerkverkehr eines einzelnen Systems oder ganzer Netze. Idealerweise erfolgt die Implementation über einen oder mehrere Sensoren, sodass bei heterogenen und wachsenden Netzen die NIDS entsprechend mitwachsen kann. Dies passiert meist auch in Verbindung mit einer Risikobewertung etwaiger betroffener Netze. Gleichzeitig sollte das System in der Lage sein, Datenverkehr aufzeichnen zu können, *sniffing*, um eine nachfolgende Analyse zu ermöglichen.

Die Ressourcennutzung des Systems sollte in einem kalkulierbaren Rahmen liegen. So sollte z. B. das Mitschneiden des Datenverkehrs nicht dazu führen, dass Teile des Datenverkehrs ohne Untersuchung weitergeleitet werden (*fail-open*) oder Pakete womöglich verworfen werden (*drop*). CPU sowie Arbeitsspeicher sollten nach ihren Möglichkeiten effizient genutzt werden.

3.1.3 Betriebsmodus

Das System muss über eine Detektionslogik (NIDS) sowie einer Intervention (NIPS) verfügen. Das System kann über weitere Schnittstellen verfügen um, eine Intervention auszulösen, beispielsweise über einen Datenaustausch zur Firewall, welche ein System blockieren kann.

3.1.4 Management-Zugriff

Der Zugriff auf die Konfiguration des Systems sollte über eine Kommandozeilenbasierte Command Line Interface (CLI), Webschnittstelle oder API-Schnittstelle verfügen. Durch den Anbieter zur Verfügung gestellte Anwendungen für den Zugriff sollten Aufgrund der Interoperabilität vermieden werden.

3.1.5 Pre-/Postprocessing

Im folgenden Punkt wird auf eine detaillierte technische Beschreibung einer speziellen Technologie verzichtet, damit auf die zu erfüllenden Funktionen eingegangen werden kann.

Preprocessing

Im Preprocessing geht es darum, dass mit bekannten sowie unbekannten Protokollen umgegangen werden kann und diese auch so geparst werden, sodass ein Zugriff auf die Datenfelder des Protokolls besteht und auf diese Detektionen verfasst werden können. Protokolle sollten unabhängig vom verwendeten Standardport erkannt werden. Bei bekannten Protokollen sollten im besten Falle Datenfelder auch interpretiert, sowie mögliche Nutzdaten extrahiert werden, um eine nachträgliche Analyse zu beschleunigen. Eine Verarbeitung der Daten sollte sich nicht ausschließlich auf eine paketweise Analyse beschränken, sondern auch komplette Datenströme analysieren können.

Postprocessing

Dies umfasst die Art und Weise wie die Metadaten im Falle einer Detektion vorliegen oder ausgegeben werden. Außerdem umfasst es im umgekehrten Fall, wie etwaige Metadaten oder Network Monitoring Systems (NMS)-Daten vorliegen oder ausgegeben werden. Das Postprocessing sollte anpass- sowie erweiterbar sein, sodass die Daten des Systems weiterverarbeitet werden können. Die Datenausgabe der NIDS sollte ohne die Einbindung zusätzlicher Drittanbietersysteme dazu geeignet sein, in z. B. einem SIEM System weiterverarbeitet zu werden.

3.1.6 Detektionsmechanismen

Die Detektion zur Angriffserkennung sollte vielseitig gestaltet sein, um ein möglichst hohes Schutzniveau zu erlangen. Vielseitig meint in diesem Fall die abgedeckten Methoden zur Detektion, um keinen Negativeffekt durch mögliche Nachteile einzelner Detektionsmechanismen zu erhalten. Die NIDS sollte Angriffsmuster erkennen, Anomalien erkennen und nach Möglichkeit Meta- bzw. NMS-Daten korrelieren können.

Unter der Erkennung von Angriffsmustern lässt sich eine klassische Signatur fassen. Signaturen sind vordefinierte Muster, die maliziöses Verhalten oder spezielle Merkmale innerhalb der Netzwerkpakete erkennen. Dies können Bytefolgen für die Ausnutzung einer Schwachstelle sein, aber auch gewisse Informationen innerhalb eines Pakets. Informationen innerhalb eines Pakets wären z. B. bekannt-schadhafte `UserAgents` innerhalb eines `HTTP-GET`, oder eine Quell-IP-Adresse.

Unter die Anomaliedetektion fällt alles das, was nicht der Norm entspricht, also Abweichungen von der Definition eines bestimmten Protokolls. Vorteilhaft wäre es, wenn außerdem eine Abweichung vom Normalzustand erkannt werden würde. Dies wäre z. B. ein hohes Datenvolumen ausgehend von einem System, welches normalerweise kaum Daten überträgt.

Unter die Datenkorrelation fällt das Zusammenfassen von vielen einzelnen, möglicherweise nicht zwangsläufig alarmierungsbedürftigen Ereignisse. Hierunter würde z. B. ein regelmäßiger Portscan der DMZ fallen, welcher über einen Zeitraum einer Woche einige Male aufgetreten ist.

3.1.7 SSL/TLS Entschlüsselung

Da immer mehr Datenverkehr standardmäßig verschlüsselt ist, besteht prinzipiell auch nicht mehr die Möglichkeit auf Protokollanomalien oder Signaturen zu prüfen, da die Nutzdaten nicht mehr im Klartext vorliegen. Deshalb sollte das System über die Möglichkeit verfügen, den Datenverkehr zu entschlüsseln und diesen zu analysieren.

3.1.8 Logging und Ausgabe

Das System sollte hinsichtlich der Speicherung von Daten oder Metadaten konfigurierbar sein. Dies betrifft sowohl Pakete oder Datenströme, die einen Alarm ausgelöst haben, sowie den weitergeleiteten Netzwerkverkehr in Form eines Paketmitschnitts (Packet Capture (pcap)).

Die hierbei anfallenden Daten sollten in einem möglichst homogenen Datenformat vorliegen und auf ein Drittsystem weitergeleitet werden können, wie z. B. ein SIEM-System oder einem Logmanagement.

3.1.9 Alert Management

Das System sollte ein Alert Management bereitstellen. Innerhalb dieses Moduls werden Detektionen als Alarm dargestellt, die als Arbeitsaufträge zu interpretieren sind. Jeder Alarm enthält detaillierte Informationen zur Detektion sowie relevante Metadaten. Ein Analyst hat die Möglichkeit, sich den Alarm zuzuweisen, zusätzliche Anmerkungen hinzuzufügen und den Alarm nach Bearbeitung als gelöst zu kennzeichnen.

Der Alarm sollte mindestens die Regel, gegen welche verstoßen wurde sowie die Rahmendaten Protokoll, Quell-/Ziel-IP-Adresse und Quell-/Ziel-Port beinhalten. Außerdem sollte in den Alarmdetails genügend Informationen vorhanden sein wie z. B. Metadaten des Pakets welche für die Detektion möglicherweise relevant waren.

3.1.10 Threathunting

Die NIDS sollte über NMS-Informationen ein Threathunting ermöglichen. Dies bezieht sich nicht ausschließlich auf Alarme, sondern soll auch ein proaktives Threathunting über die gespeicherten Metadaten für Pakete oder Datenströme ermöglichen.

3.1.11 Reporting

Das Reporting schließt organisatorisch an das Alert Management, dem Threathunting und der Ausgabe an. Ziel eines Reportings kann sein, Statistiken im Bereich des Threathunting anzufertigen, um mögliche neue Bedrohungen identifizieren zu können. Ein weiteres Ziel wäre die Aufarbeitung gewisser Metriken im Rahmen eines Management Reports.

3.1.12 Ruleset Management

Unter das Ruleset Management fallen mehrere Teilaspekte.

Es muss möglich sein, individuelle Regeln zu definieren und bestehende Regeln oder Regelwerke zu deaktivieren. Optional sollte es möglich sein, bestehende Regeln individuell anzupassen.

Regeln bzw. Regelwerke müssen im-, sowie exportierbar sein. Das Regelwerk sollte möglichst einfach zu aktualisieren sein. Optional ist es von Vorteil, Drittanbieter-Regelwerke einzubinden und diese analog verwalten zu können.

Zuletzt sollten sich externe Informationsquellen wie Cyber Threat Intelligence (CTI) einbinden lassen, um auf dessen Basis Regeln verfassen zu können.

3.1.13 Organisatorische Aspekte

Ein letzter Punkt beschreibt mehrere unterschiedliche organisatorisch Aspekte einer NIDS.

Es sollte möglich sein, über ein kryptografisch sicheres und etabliertes Verfahren an der NIDS zu authentifizieren. Es kann möglich sein, dass die durchgeführten Logins im Rahmen eines Audits überprüft werden können.

Die NIDS sollte über ein Berechtigungskonzept verfügen, sodass es nicht zu ungewollten oder nicht autorisierten Modifikationen innerhalb des Systems kommt.

Die NIDS sollte über fälschungssichere System-, sowie Anwendungs- oder Ruleset Updates verfügen.

3.2 Bewertung der ausgewählten NIDS

Die hier betrachteten Systeme sind vorab selektiert worden. Bei einer initialen evaluierung über mögliche NIDS, fallen neben kommerziellen NIDS und NGFW häufig dieselben Namen: Snort, Suricata, Zeek (ehem. Bro), OSSEC, Wazuh, Security Onion oder Sguil.

OSSEC und Wazuh sind ausschließliche HIDS und deshalb nicht weiter betrachtet worden.

Zeek und Sguil sind keine echten NIDS in diesem Sinne, eher NMS. Derartige Systeme sind darauf angewiesen, dass eine Anomaliedetektionslogik in einem an ihr angeschlossenen System funktioniert oder durch manuelles anlegen von Alarmen innerhalb des Systems eine Detektion stattfindet[75]. Im Falle von Sguil kommt hinzu, dass der letzte `Commit` auf Version 1.0.0 im März 2018 stattgefunden hat, was gegen eine regelmäßige Pflege spricht[32].

Security Onion ist ein Bundle aus verschiedenen Softwareprodukten, welches unter anderem auch eine NIDS beinhält. Da hierbei auf Suricata zurückgegriffen wird und alle weiteren Funktionen nicht Gegenstand dieser Arbeit sind, ist auch Security Onion nicht weiter berücksichtigt worden.

Näher betrachtet werden deshalb Snort und Suricata.

Bei Snort handelt es sich um eine NIDS, welche je nach Konfiguration auch als NIPS eingesetzt werden kann. Snort existiert bereits seit 1998 und wird seit 2013 durch Cisco Systems aktiv gepflegt und weiterentwickelt. Das für Snort hauptsächlich verwendete Regelset wird durch Cisco Talos regelmäßig gepflegt. Aufgrund der bereits lange zurückreichenden Existenz des Projektes ist Snort ein weit verbreitetes NIDS. Als Teil eines Geschäftsmodells besteht die Möglichkeit neben einem kostenlosen Regelset (`Community Ruleset`) durch ein kostenpflichtiges Abonnement auf ein erweitertes Regelset (`Subscriber Ruleset`) zugreifen zu können[69].

Bei Suricata handelt es sich um eine NIDS, welche je nach Konfiguration auch als NIPS eingesetzt werden kann. Erster Code für Suricata ist bereits 2007 geschrieben worden, ein erstes Release gab es 2009. Es wird durch die Open Information Security Foundation (OISF) aktiv gepflegt und weiterentwickelt. Trellix (ehem. FireEye), ein Anbieter für diverse IT-Security Produkte, verwendet Suricata in Teilen seiner Produkte, was der OISF finanziell zugutekommt. Aufgrund der bereits lange zurückreichenden Existenz des Projektes ist auch Suricata ein weit verbreitetes NIDS. Das für Suricata hauptsächlich verwendete Regelset (`Emerging Threats`)

wird durch Proofpoint gepflegt. Ähnlich zu Snort existiert auch hier ein kostenfreies Regelwerk (`ETOpen Ruleset`), sowie ein kostenpflichtiges Regelwerk (`ETPro Ruleset`)[41][77].

Bei beiden Systemen handelt es sich um lange etablierte NIDS, sodass der Funktionsumfang selber auch nahezu identisch ist. Deshalb ist von einem tabellarischen Vergleich für das Vorhandensein von gewissen Funktionen Abstand genommen worden. Vielmehr ist während einer genaueren Betrachtung aufgefallen, dass ausgewählte Merkmale auf beiden Systemen vorliegen und sich die Unterschiede im Detail ergeben.

Deshalb wird im Folgenden ein qualitativer Vergleich beider Systeme anhand der in Abschnitt 3.1 beschriebenen Merkmale erfolgen. Die Informationen hierzu stammen aus der offiziellen Dokumentation von Snort[70] und Suricata[71].

3.2.1 Anwendungsbereich

Bei Snort sowie Suricata handelt es sich um reine NIDS. Sowohl Snort als auch Suricata können dazu verwendet werden, um den Netzwerkverkehr des lokalen Systems selber zu überwachen.

3.2.2 Architektur und Ressourcennutzung

Sowohl Snort als auch Suricata arbeiten als sogenannte Sensoren, was es auch wachsenden Netzen ermöglicht diese flexibel zu implementieren. Beide Systeme können die beteiligten Netzwerkkarten im *Promiscous Mode* betreiben um den Netzwerkverkehr aufzeichnen zu können, *sniffing*.

Bis zu Snort Version 3 (2021), fehlte eine Multi-Threading bzw. Multi-Core Unterstützung, über welche Suricata schon in ihrem ersten Release verfügte. Bis zu diesem Release, also Snort Version 2 und älter, war Suricata ungeschlagen, was es unterschiedliche Metriken wie z. B. CPU- und Arbeitsspeicherauslastung oder Paketverlust anging. Ein mit Snort 3 und Suricata 6.0 durchgeführtes Experiment zeigte eine ähnliche CPU- und Arbeitsspeicherauslastung bei Netzwerklast, jedoch einen etwas höheren Paketverlust bei Snort. Die aus dem Paketverlust resultierende Falsch-Negativ-Rate war bei Suricata besser[8].

3.2.3 Betriebsmodus

Sowohl Snort als auch Suricata können als IDS und IPS betrieben werden.

3.2.4 Management-Zugriff

Sowohl Snort als auch Suricata sind primär über eine Kommandozeilen-basierte CLI zu administrieren. [70, S. 3 f.] [71, S. 33 f.]

Beide verfügen standardmäßig über keine Webschnittstelle, es existieren jedoch GitHub-Projekte, die je nach Anwendungszweck erlauben auf die Daten des jeweiligen Systems per Browser zuzugreifen. Exemplarisch wären `BASE` für Snort oder `evebox` für Suricata zu erwähnen.

Im Gegensatz zu Suricata bietet Snort für bestimmte Funktionen eine API-Unterstützung, jedoch nicht in Form einer vollwertigen API. Beide Systeme werden durch verschiedene GitHub-Projekte erweitert, die API-ähnliche Funktionen bereitstellen.

3.2.5 Pre-/Postprocessing

Snort erkennt einige ausgewählte Protokolle wie HTTP, File Transfer Protocol (FTP) oder Simple Mail Transfer Protocol (SMTP) über den jeweiligen Protokoll Preprocessor. Existiert kein Preprocessor oder ein Plugin für ein zu untersuchendes Protokoll, ist keine nähere Untersuchung der Datenpakete oberhalb der Transportschicht (OSI-Layer 4) möglich. Ausnahmen hiervon bilden Signatur-basierte Regeln, welche auf den reinen Paketinhalt, in diesem Fall Bytefolgen, abzielen. Preprocessoren sowie Plugins lassen sich durch den Benutzer erweitern. Diese Funktion der Protokolldetektion kann auch durch Alarmregeln erweitert werden, sodass z. B. Datenverkehr über TCP auf Port 21 als `Cleartext FTP or Telnet Detected` alarmiert wird[70, S. 26 ff.].

Suricata identifiziert Protokolle über eine automatische Protokollerkennung, welche den Inhalt des Pakets analysiert und anhand dessen versucht, das Protokoll zu bestimmen. Suricata unterstützt mehr und modernere Protokolle als Snort. Für einige ausgewählte Protokolle wie z. B. SMB oder FTP existiert ein Datei-Extraktions-Feature, sodass die Anfrage, Antwort oder übertragene Dateien automatisch zusammengesetzt werden. Die automatische Protokollerkennung ist erweiterbar. Anders als bei Snort kann bei Suricata vom Standardport abgewichen werden und das Protokoll wird nach wie vor erkannt. Erkennt die Protokollerkennung das Protokoll

nicht, verhält es sich wie bei Snort, es ist keine nähere Untersuchung möglich. Auch hier gilt die Ausnahme von Signatur-basierten Regeln[71, S. 39 ff.].

Snort und Suricata unterstützen beide sowohl die Detektion auf Basis einzelner Pakete sowie eines ganzen Datenstroms.

Snort verwendet für die Speicherung von Metadaten oder Alarmen standardmäßig das Unified2-Format. Hierbei handelt es sich um ein proprietäres Datenformat. Außerdem ist die Ausgabe in Form von Syslog sowie Paketmitschnitte möglich. Die Ausgabe lässt sich durch `barnyard2` modifizieren. Hierbei handelt es sich um einen proprietären Spooler für die Unified2-Daten, welcher auf GitHub allerdings archiviert wurde. Archivierte Projekte werden nicht mehr gepflegt oder weiterentwickelt. Weitere Modifikationen am Postprocessing oder dem Output lässt sich über Output-Plugins realisieren[70, S. 7, 23 f.].

Suricata verwendet für die Speicherung von Metadaten oder Alarmen standardmäßig das Extensible Event Format JSON (EVE JSON) Format. Hierbei handelt es sich um ein für Suricata entwickeltes Format, welches als Basis jedoch JavaScript Object Notation (JSON) verwendet. JSON ist in dem Kontext von SIEM oder anderen Logmanagement-Systemen ein Standardformat[71, S. 321 ff.].

3.2.6 Detektionsmechanismen

Sowohl Snort als auch Suricata unterstützen die Signatur- und Anomaliebasierte Detektion. Da Suricata deutlich mehr Protokolle abdeckt, fällt die Anomaliedetektion auch besser aus.

Beide Systeme unterstützen eine zeitliche Komponente in ihren Alarmen. So lassen sich Schwellwerte (*thresholds*) für das mehrfache Auftreten eines gewissen Ereignisses über einen zu definierenden Zeitraum festlegen. Es ist unbekannt, wie sehr die Systeme durch mehrerer solcher Regeln für Zeiträume von Tagen oder Wochen ausgelastet werden. Eine echte Datenkorrelation lässt sich so höchstwahrscheinlich nicht abdecken. Dies ist nur über ein entsprechend angeschlossenes SIEM oder Logmanagement-System möglich. Im Postprocessing hat sich Suricata durch die Verwendung von Standardformaten (EVE JSON) als deutlich zukunftsweisender gezeigt, sodass eine Einbindung in ein Drittsystem weniger aufwand darstellen würde[70, S. 13 ff.][71, S. 39 ff.].

3.2.7 SSL/TLS Entschlüsselung

Beide Systeme unterstützen nicht standardmäßig die Dekodierung von verschlüsseltem Datenverkehr wie z. B. HTTPS. Der Datenverkehr muss über Drittsysteme vorher entschlüsselt werden und dann anschließend durch eines der beiden Systeme weitergeleitet werden[71, S. 469].

3.2.8 Logging und Ausgabe

Beide Systeme lassen sich im Hinblick auf die aufzuzeichnenden Details anpassen.

Suricata lässt sich in vielerlei Hinsicht anpassen. Neben einem festzulegenden Detailgrad etwaiger Metadaten wie z. B. TLS-Informationen, lässt sich unter anderem auch der Detailgrad des Datenverkehrs oder von `Flow`-Daten konfigurieren. Unter `Flow`-Daten fasst man die Parameter einer einzelnen Verbindung, im weiteren Sinne einer *Session*, wie z. B. Quell-/Ziel-IP, Quell-/Zielport, Protokoll, Übertragene Pakete/Bytes oder Start-/Endzeitpunkt[71, S. 321 ff.].

Die Aufzeichnung von sogenannten `Flow`-Daten ist bei Snort jedoch nur über Umwege möglich und nicht direkt implementiert[70, S. 78].

Die Ausgabeformate und damit einher gehenden Implikationen werden im nachfolgenden Unterabschnitt 3.2.9 (Alert Management) beschrieben.

3.2.9 Alert Management

Beide Systeme verfügen über kein Alert Management. Für beide Systeme muss eine Drittsoftware installiert werden, hierfür würde z. B. `Kibana`, eine Datenalysesoftware des Elastic Stack, infrage kommen. Der Elastic Stack, auch ELK Stack genannt, ist eine aus mehreren Komponenten bestehende Plattform, mit welcher Daten erfasst, verarbeitet und analysiert werden können[28].

Die Integration des Alert Management von Snort in Kibana ist jedoch nicht so ohne weiteres möglich. Snort gibt wie bereits oben beschrieben Ereignisse in Form des proprietären Unified2-Formats aus, welches nicht ohne weiteres durch Kibana verarbeitet oder visualisiert werden kann. Hier ließe sich an beiden Enden ansetzen, entweder werden die Ereignisse über ein entsprechendes Output-Plugin in Snort so aufbereitet, dass eine Integration möglich ist, oder aber das Parsing wird auf `Logstash`, einer dem ELK Stack zugehörigen Datenverarbeitungslösung, ausgelagert.

Die Integration des Alert Management von Suricata in Kibana hingegen stellt sich einfacher dar. Da die Ausgabe der Ereignisse von Suricata bereits in dem an JSON angelehnten EVE JSON erfolgt, ist die Integration einfacher. Elastic stellt hierfür bereits einsatzfähige Module zur Verfügung, welche die Daten parsen können.

3.2.10 Threathunting

Je nach Konfiguration speichern beide Systeme Metadaten bzw. NMS-Informationen. Die gespeicherten Logs lassen sich auf Snort sowie Suricata lokal in der CLI analysieren[71, S. 463 ff.].

Zusätzlich ist es möglich, die Ereignisse in `Kibana` zu analysieren. Die dabei auftretenden Implikationen sind im Bereich des Unterabschnitt 3.2.9 (Alert Management) beschrieben worden. Suricata ist interoperabler als Snort und würde sich einfacher einbinden lassen.

3.2.11 Reporting

Ein Reporting unterstützt keines der beiden Systeme. Dieses muss über eine Drittlösung wie z. B. `Kibana` erfolgen.

Die damit einher gehenden Implikationen sind im Bereich des Unterabschnitt 3.2.9 (Alert Management), Unterabschnitt 3.2.8 (Logging und Ausgabe) und Unterabschnitt 3.2.10 (Threathunting) beschrieben worden.

3.2.12 Ruleset Management

Grundsätzlich erfüllen Snort sowie Suricata die Anforderungen. Es lassen sich individuelle Regeln definieren und bestehende Regeln können angepasst oder deaktiviert werden.

Da das Regelwerk aus eingebundenen Dateien besteht, lassen sich diese auch im-/exportieren. Ein automatisches Update des Regelwerks ist bei Snort allerdings nicht ohne Drittanbieter Software wie z. B. `PulledPork` möglich[70, S. 6]. Suricata hat für eine Aktualisierung des Regelwerks einen gesonderten Update-Mechanismus[71, S. 191 ff.].

Die Einbindung von CTI wie z. B. Malware Information Sharing Platform & Threat Sharing (MISP) ist in Snort nicht direkt vorgesehen. MISP ist einer der größten

Open Source Anbieter für den gemeinschaftlichen Austausch von Bedrohungsinformationen (CTI). Hier wäre auch entweder die Integration von `PulledPork` möglich, alternativ müssten Indicator of Compromise (IoC)-Listen bezogen und angepasst werden, sodass diese der Form des Regelwerks entsprechen. Der Prozess zur Einbindung von CTI ist in Suricata und dem gesonderten Update-Mechanismus für `SSLBL` (Blockliste für schädliche SSL-Zertifikate) und `abuse.ch` (Anbieter für Blocklisten) implementiert[71, S. 192].

Ähnlich wie CTI lassen sich auch weitere Quellen für Regelwerke für Snort analog hinzufügen. Es sind zur Automatisierung des Prozesses eigene Skripte oder Erweiterungen wie `PulledPork` nötig.

Suricata hat neben dem `Emerging Threats` Regelwerk auch weitere optionale Regelwerke wie z. B. eines der OISF. Diese lassen sich über den Update-Mechanismus aktualisieren. Weitere öffentliche Regelwerke lassen sich in diesen Update-Mechanismus hinzufügen, wenn diese in dem richtigen Format vorliegen. Analog lassen sich auch Drittquellen einbinden, müssen allerdings auch erst in die korrekte Form gebracht werden. Der vorteilhafte Update-Mechanismus müsste an dieser Stelle dann durch eigene Skripte zur Aktualisierung erweitert werden.

3.2.13 Organisatorische Aspekte

Die Authentifizierung, ein Login-Audit, ein Berechtigungskonzept sowie fälschungssichere System-/Anwendungs-/Ruleset-Updates lassen sich nur bedingt abdecken.

Die Authentifizierung sowie das Login-Audit lassen sich auf Betriebssystem-Ebene lösen. Snort und Suricata besitzen kein Feature, welches dieses umsetzt.

Ein Berechtigungskonzept besitzen beide Systeme nicht. Auch dies müsste über die Betriebssystem-Ebene umgesetzt werden, innerhalb der Anwendung selber existieren dann aber keine feineren Berechtigungsstrukturen mehr.

System- und Anwendungsupdates auf debian-basierenden Linux-Betriebssystemen sind durch den `apt` Update Mechanismus so gestaltet, dass sowohl die Übertragung der Daten als auch eine Integrität der Updatepakete sichergestellt ist.

Die Regelupdates selber werden im Transport verschlüsselt heruntergeladen. Die digitalen Signaturen der Rulesets werden anschließend validiert.

3.3 Fazit

Obwohl Snort sowie Suricata auf den ersten Blick einen nahezu identischen Funktionsumfang besitzen, hat sich Suricata als geeignetes System ergeben. Dies hängt vor allem damit zusammen, dass Suricata insgesamt moderner und deutlich interoperabler ist.

Dies beginnt mit einer größeren Bandbreite an unterstützten Protokollen. Ohne Deep Packet Inspection (DPI) lassen sich keine sinnvollen Regeln erstellen. Ein Rückgriff auf CTI scheint nicht sinnvoll und dient eher, die sprichwörtlichen *niedrig hängende Früchte* zu ernten. Damit ist gemeint, dass z. B. bekannte IP-Adressen von denen eine schadhafte Aktivität ausgeht durch die CTI zu erkennen.

Das von Snort verwendete Unified2 proprietäre Datenformat zeigt sich an mehreren Stellen, z. B. für die Einbindung in ein SIEM System als ungünstige Designentscheidung. Die Tatsache, dass eine hierfür relevante Komponente, `barnyard2`, inzwischen nicht mehr aktiv weiterentwickelt wird, ist zusätzlich negativ zu bemerken. Positiv hingegen ist zu bemerken, dass Suricata mit dem auf JSON basierenden EVE JSON Format zukunftsweisender ist.

Das Alert Management ist zwar in beiden Systemen nicht standardmäßig enthalten, jedoch bietet Suricata aufgrund seiner technischen Voraussetzungen eine bessere Integrationsmöglichkeit in ein solches System.

Ein im operativen Bereich relevanter Punkt ist das Management der Rulesets bzw. Regeln. Der gesamte Prozess erfordert bei Suricata selten ein manuelles eingreifen. Bei Problemen wäre eine Fehlersuche bei Snort aufgrund diverser selbst erstellter Skripte zur Automation umso schwieriger. Positiv ist außerdem hervorzuheben, dass Suricata mit der `Emerging Threat Open`[76] Liste aktuell über 50.000 Regeln verfügt, was deutlich mehr ist als `Snort Community Rules`, ca. 4.000[73]. Auch die Tatsache, dass CTI verhältnismäßig niederschwellig eingebunden und aktuell gehalten werden kann, verringert den operativen Aufwand.

Da die oben aufgeführten Gründe für Suricata sprechen, wird im nachfolgenden ausschließlich Suricata zum Einsatz kommen.

3.3.1 Betrachtung des Regelwerks ET Open

ET Open ist nicht strukturiert, was eine genauere Aufarbeitung des Regelwerks erschwert. Auch ist der *classtype* einer Regel nicht immer konsequent gesetzt, sodass auch z. B. *Trojan-Activity* gesetzt sein kann, aus der Beschreibung der Regel aber deutlich wird, dass C2 erkannt werden soll. Es existieren in dem Regelwerk vom 09.08.2024 insgesamt 8.303 Regeln zu den Stichwörtern *classtype:command-and-control*, *Command and Control*, *CnC*, *c2 communication* und *Beacon*. Hiervon sind 1.747 Regeln auskommentiert, was zur Folge hat, dass diese nicht aktiv sind. Die verbleibenden 6.556 Regeln umfassen unterschiedliche Protokolle:

- HTTP: 2.351 und HTTP 1.1: 408
- DNS: 2.277
- TLS: 746

Eine vollständige Auflistung der Häufigkeitsverteilung der Protokolle, für die eine Detektionsregel existiert, ist Listing 4 in Anhang A zu entnehmen. Eine qualitative Betrachtung erfolgt aufgrund der Anzahl der Regeln nur stichprobenartig und dient eher dazu, einen Überblick zu gewinnen.

Die verfügbaren Regeln für HTTP lassen sich wie folgt kategorisieren:

- ET MALWARE: 2.564
- ET MOBILE_MALWARE: 74
- ET ADWARE_PUP: 74
- ET CURRENT_EVENTS: 22

Die meisten Regeln zielen auf Malware ab und untersuchen auf viele, der im HTTP-Protokoll verfügbaren Suricata-Keywords. Zum Teil wird auch mit Perl Compatible Regular Expressions (PCRE) nach gewissen schadhaften Elementen gesucht. Die Regeln für HTTP wirken alles in allem sehr konkret.

Die verfügbaren Regeln für DNS lassen sich wie folgt kategorisieren:

- ET MALWARE: 1.911
- ET MOBILE_MALWARE: 341
- ET PHISHING: 16
- ET ADWARE_PUP: 4

Auch hier zielen die meisten Regeln auf Malware ab, bei der Durchsicht dieser Regeln konnten keine Detektionsregeln festgestellt werden, die auf allgemein ungewöhnliche Parameter im DNS-Protokoll abzielen. In den meisten Regeln wird ausschließlich auf schadhafte oder teile schadhafter Domänen gesucht. Inwiefern dies im allgemeinen erfolgsversprechend ist, ist außerhalb von aktuellen Kampagnen in Zweifel zu ziehen, da ein häufiger wechsel von verwendeten Domänen üblich ist.

Die verfügbaren Regeln für TLS lassen sich wie folgt kategorisieren:

- ET MALWARE: 695
- ET MOBILE_MALWARE: 42
- ET JA3: 7

Etwa die Hälfte der Regeln zielen auf gewisse (Teil-)Namen im TLS Server Name Indication (SNI) ab und eine weitere Hälfte auf das `Subject`-Feld im TLS-Zertifikat.

Abschließend lässt sich festhalten, dass der Erwartung entsprechend durchaus Regeln vorliegen, welche die C2-Kommunikation detektieren könnten. Dies wird im Rahmen des nachfolgenden Experiments in Kapitel 4 einem praktischen Test unterzogen.

4 Experiment zur Bewertung der Detektionsraten

Dieses Kapitel wird sich mit dem Experiment zur Bewertung der Detektionsraten von C2 Kommunikation auseinandersetzen.

Abschnitt 4.1 wird sich mit der für das Experiment festgelegten Methode beschäftigen. Abschnitt 4.2 wird das Experiment in seiner konkreten Gestaltung und Konfiguration beschreiben. Abschnitt 4.3 wird die Durchführung und die dabei festgestellten Verhaltensweisen beschreiben. Zuletzt wird in Abschnitt 4.4 ein Fazit über das durchgeführte Experiment gezogen werden.

4.1 Methode

Die Tatsache, dass eine Kombination aus NIDS, HIDS und sonstiger (Netzwerk-)Sicherheitseinrichtungen oder lokalen Härtungsmaßnahmen im Rahmen einer *Defense in Depth* Strategie wahrscheinlich die sinnvollste Lösung wäre, wird an der Stelle ausgeklammert, da sie nicht Fokus der vorliegenden Arbeit ist. Die Ausgangslage ist so gestaltet, dass davon auszugehen ist, dass lediglich das Netzwerk an einem zentralen Knotenpunkt in Richtung des Internet-Zugangs überwacht werden kann. In diesem Fall ist davon auszugehen, dass z. B. Fremdgeräte Netzwerkzugriff haben, Schatten-IT vorherrscht oder andere Dogmen wie Freiheit der Wissenschaft, Forschung und Lehre gemäß Artikel 5 des Grundgesetzes gegen eine restriktive Verwaltung der Netzwerkteilnehmer spricht.

Für das Experiment ist in Kapitel 3 zunächst Suricata als geeignete NIDS identifiziert worden. Neben zeitlichen Gründen macht es aus operativer Sicht an dieser Stelle wenig Sinn, zusätzlich Snort zu testen, welches z. B. weniger Protokolle abdeckt. Zusätzlich zum Standardregelwerk von Suricata (`ET Open`) auch das Snort native Regelwerk (`Snort Community Rules`) eingebunden. Dieses ist in Teilen zu Suricata kompatibel, da in Regeln Bestandteile verwendet worden sind, welche in Suricata nicht existieren.

Diese NIDS wird im Rahmen eines fiktiven Angriffs daraufhin geprüft, ob unterschiedliche Methoden zum Verschleiern des C2 Datenverkehrs im Rahmen des

Standardregelwerks einen Alarm auslösen. Wie in Unterabschnitt 2.3.3 beschrieben, existieren unterschiedliche Möglichkeiten den C2 Datenverkehr zu verschleiern. Die gängigen Methoden sind, auch wie in Abschnitt 2.4 beschrieben, von diversen C2-Frameworks implementiert, sodass davon ausgegangen werden muss, dass theoretisch jede Methode angewandt werden könnte.

Deshalb werden im Rahmen des Experiments unterschiedliche Protokolle bedient, um die Detektionsrate überprüfen zu können und eine realistische Erwartungshaltung in Bezug auf das Standardregelwerk von Suricata `ET Open` sowie die `Snort Community Rules` geben zu können. Hierfür werden einzelne der in Unterabschnitt 2.3.3 beschriebenen C2-Frameworks oder Tools angewendet. Ausgewählte C2-Protokolle sind:

- DNS mittels `dnscat2`[36]
- HTTP mittels `Sliver`
- HTTPS mittels `Sliver`
- SMB und HTTP mittels `Havoc`
- mTLS mittels `Sliver`
- VPN (WireGuard) mittels `Sliver`

Grund für diese Auswahl war unter anderem eine Übersicht von Sophos zu verwendeten C2-Protokollen aus 2021, siehe Abbildung 5.

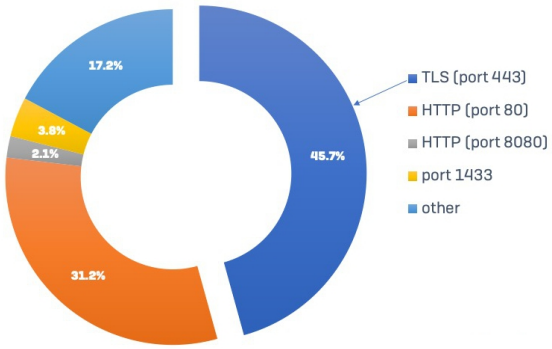

Abbildung 5: Aufschlüsselung der von Malware verwendeten C2-Protokollen für das erstes Quartal 2021[30]

Festgestellte C2-Protokolle waren dort TLS, HTTP, HTTPS, ein nicht näher beschriebenes Protokoll über Port `1433` und 17,2 % andere Protokolle. Überraschenderweise besaß HTTP zu diesem Zeitpunkt 31,2 % Anteil, obwohl der Anteil von C2 via TLS von 2020, 23 %, auf 45,7 % in 2021 angestiegen ist. Im Jahr 2022 machten

Remote Access Tools (26,5 %) und Web-Protokolle (67,6 %) laut "The DFIR Report"den Großteil der C2-Methoden aus[4]. Web-Protokolle beschreiben den Fall, unabhängig vom verwendeten Port, HTTP, HTTPS und WebSocket Datenverkehr[15]. Aktuellere Zahlen, die einen Trend zur Verdrängung von HTTP bestätigen würden, haben sich nicht ermitteln lassen. Allerdings gibt es auch in 2024 Veröffentlichungen, die über C2 via HTTP berichten z. B. Palo Alto[62]. Dies deckt sich mit der Erfahrung des Autors, dass wenn Cobalt Strike bei Ransomware-Vorfällen genutzt wurde, es sich hierbei um HTTP Beacons handelte.

In Bezug auf die Relevanz der verwendeten OST bzw. C2-Frameworks kommt Sliver auch in der Realität zum Einsatz, siehe Abbildung 4. Havoc selbst wird in dieser Erhebung zwar nicht aufgelistet, wird aber auch bei Cyberangriffen verwendet, wie z. B. der Blog von Zscaler zeigt[64]. Hinzu kommt, dass die Implementation von C2 via SMB von Havoc und Cobalt Strike deckungsgleich sind, Cobalt Strike aus der zuvor aufgeführten Erhebung den ersten Platz einnimmt[48][2]. dnscat2 findet zwar auch keine Erwähnung, jedoch finden wir hier auch eine identische Implementierung vor, wie es bei Sliver der Fall ist[7].

Um das Experiment innerhalb eines ungefährlichen Rahmens betreiben zu können, wird sich keiner aktiven Malware bedient, sondern über die aufgelisteten C2-Frameworks oder Tools der Datenverkehr erzeugt. Dies hat neben einem zu vermeidenden Informationsabfluss auch praktische Gründe, da Malware Samples je nach Alter nicht mehr auf noch aktive C2-Server zugreifen können. Des Weiteren wäre so auch nicht das Verhalten des Beacons zu steuern. Gerade Interaktion zwischen dem Beacon und dem C2-Server schlagen sich möglicherweise anders in den übertragenen Paketen nieder. Ein weiterer Grund hierfür ist, dass die spezifische Konfiguration des Beacons auch angepasst werden kann. So kann z. B. das Intervall, manchmal auch *sleep* genannt, des Beacons nach Belieben konfiguriert werden oder weitere *Defense Evasion* Mechanismen ausgetestet werden.

Sowohl der C2-Server als auch das System, auf welchem der Beacon ausgeführt sind, befinden sich unter der eigenen Kontrolle. So sind Schutzmaßnahmen wie beispielsweise der Windows Defender explizit deaktiviert worden. Dies hat einerseits den Grund, dass die Umgehung von Schutzmaßnahmen ähnlich reeller Bedingungen explizit nicht Bestandteil des Experiments ist. Außerdem wird so sichergestellt, dass die Client-Server-Kommunikation störungsfrei gewährleistet ist.

Da der C2 Datenverkehr nachfolgend in Kapitel 5 analysiert wird, muss dieser auch aufgezeichnet werden. Das passiert aus praktischen Gründen direkt auf dem System,

auf welchem sich der Beacon befindet.

Durch die Verwendung der häufig verwendeten NIDS Suricata, wird ein realistisches Bild über die Erwartung an das verfügbare Standardregelwerk `Emerging Threats` gegeben. Dadurch, dass der C2-Server und das System, auf welchem die Beacons ausgeführt werden, sich unter der eigenen Kontrolle befinden, kann die Konfiguration nach eigenem Ermessen erfolgen. So können unterschiedliche C2-Protokolle sowie mögliche *Defense Evasion* Mechanismen ausgetestet werden. Die ausgewählten Protokolle entsprechen außerdem einem zu erwartenden Rahmen. Die nachfolgende Analyse des Datenverkehrs wird außerdem aufzeigen, inwiefern etwaige Lücken in der Detektion geschlossen werden können.

4.2 Vorgehensweise

In diesem Abschnitt wird die genaue Ausgestaltung des Experiments beschrieben. Unterabschnitt 4.2.1 beschreibt die zugrunde liegende Virtualisierungsumgebung. Unterabschnitt 4.2.2 beschreibt das innerhalb der Virtualisierungsumgebung konfigurierte Netzwerk. Unterabschnitt 4.2.3 wird auf die VMs eingehen, vorwiegend C2-Server sowie den Client, auf welchem der Beacon ausgeführt wird. Zuletzt wird in Unterunterabschnitt 4.2.3 beschrieben, wie genau die Suricata NIDS konfiguriert ist. Unterabschnitt 4.2.5 wird auf die Erstellung der verwendeten Beacons eingehen.

4.2.1 Grundlagen und Virtualisierungsumgebung

Das durchzuführende Experiment wird größtenteils in einer vollständig virtualisierten Umgebung stattfinden. Die zugrunde liegende Hardware ist ein MacBook Pro des Herstellers Apple mit dem Betriebssystem 14.6.1 (Codename Sonoma) vom 07.08.2024. Bei der verwendeten Virtualisierungsumgebung handelt es sich um VMware Fusion Professional in der Version 13.5.2 vom 15.05.2024.

Das Netzwerk ist Teil von VMware Fusion, wo verschiedene virtuelle LANs konfiguriert und den Netzwerkadaptern der VMs zugewiesen werden können. Der Host der Virtualisierungsumgebung, hier das MacBook, dient dabei als Internetzugang für die VMs. Die detaillierte Netzwerkstruktur wird in Unterabschnitt 4.2.2 beschrieben.

Für die realitätsnahe Nachbildung der hierarchischen DNS-Auflösung, im weiteren Sinne C2 via DNS ist eine Second-Level Domäne erworben worden. Da der dazugehörige Server eine öffentliche IPv4-Adresse benötigt, ist eine VM bei dem Hosting

Provider Digital Ocean installiert worden. Die weitere Beschreibung dessen erfolgt in Unterabschnitt 4.2.4.

4.2.2 Konfiguration des Netzwerks

Die physische Netzstruktur wurde begrenzt auf das Notwendigste und in der Virtualisierungsumgebung VMware Fusion nachgebildet, siehe Abbildung 6.

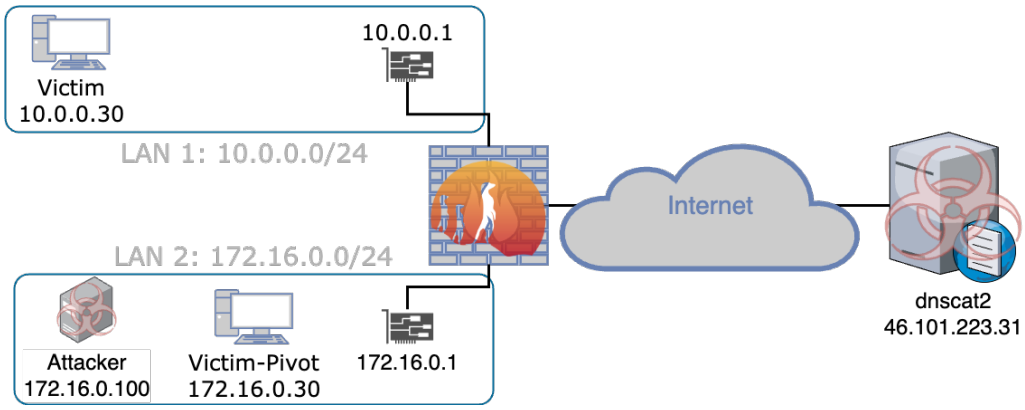

Abbildung 6: Grafische Darstellung des konfigurierten Netzwerkes

Die NIDS Suricata ist gleichzeitig zentraler Router für das gesamte virtuelle LAN. Sie verfügt über die drei folgenden Netzwerkkarten:

- Bridge zum Hypervisor (MacBook), gleichzeitig Internet Breakout für das virtuelle Netzwerk
- Gateway für LAN 1 (`10.0.0.0/24`)
- Gateway für LAN 2 (`172.16.0.0/24`)

Den beteiligten VMs sind die zutreffenden Netzwerkkarten des jeweiligen LANs zugewiesen worden. So können die Systeme der unterschiedlichen LANs über Suricata als NIDS/Router kommunizieren.

Um IP-Adresskonflikte zu vermeiden, ist dem LAN 2 ein privater IP-Adressraum zugewiesen worden.

Der Autoritative NS besitzt eine durch Digital Ocean zugewiesene, öffentliche IPv4-Adresse, `46.101.223.31`

4.2.3 Konfiguration der VMs

Im Rahmen des Experiments kommen insgesamt 5 VMs zum Einsatz, welche im Folgenden näher beschrieben werden. Um eine Zuordnung zu erleichtern werden die folgenden Aliasse eingeführt:

1. **`Victim`** VM in LAN 1, auf welcher die Beacons ausgeführt werden
2. **`Victim-Pivot`** VM in LAN 2 auf welcher speziell der SMB Beacon ausgeführt wird
3. **`Attacker`** VM in LAN 2 auf welcher die beiden C2-Frameworks Sliver und Havoc ausgeführt werden
4. **`NS`** VM gehostet bei Digital Ocean, auf welcher die C2 via DNS Anwendung dnscat2 ausgeführt wird
5. **`Suricata`** VM auf welcher die NIDS Suricata installiert wurde, um den Datenverkehr zu überwachen

Victim

Hierbei handelt es sich um eine VM, auf welcher Windows 10 Home in der Version 22H2 vom 18.10.2022 installiert worden ist. Für den Datenaustausch wird ein durch VMware Fusion geteilter Ordner verwendet und das Clipboard geteilt.

Zusätzlich ist die Software `Wireshark` in Version 4.2.6 vom 10.07.2024 installiert worden, welche dazu benötigt wird, den Datenverkehr der Netzwerkkarte mitschneiden zu können. Wireshark setzt bei Aufzeichnung den Modus der Netzwerkkarte automatisch in den *promiscuous mode*, sodass alle Netzwerkpakete aufgezeichnet werden können. Außerdem Visual Studio 2022 in Version 17.11 vom 13.08.2024 installiert worden, um den dnscat2 Beacon kompilieren zu können, siehe Unterabschnitt 4.2.5.

Um die Windows Defender Real-Time Überwachung zu deaktivieren, ist der folgende Registry-Key angelegt worden[63]: `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\WindowsDefender\Real-TimeProtection`
In dem Key `Real-Time Protection` ist der Value `DisableRealtimeMonitoring` vom Typ `REG_DWORD` mit dem Wert 1 angelegt worden.

Victim-Pivot

Hierbei handelt es sich um einen Klon der VM `Victim`, sodass die Konfiguration in ihrer Grundlage identisch ist.

Nötig war dies, da der Havoc C2 Beacon (SMB) nicht direkt zwischen `Victim` und dem C2-Server kommunizieren kann. SMB kann in diesem Fall nur dazu benutzt werden, um zwei Beacons miteinander zu verbinden, sodass der C2-Server dann indirekt mit dem nachgelagerten System kommunizieren kann. Deshalb befindet sich `Victim-Pivot` auch im selben LAN wie `Attacker`, denn so wird der SMB-Datenverkehr zwischen `Victim` und LAN 1 über Suricata zu `Victim-Pivot` in LAN 2 geleitet. Eine grafische Darstellung dessen ist Abbildung 7 zu entnehmen.

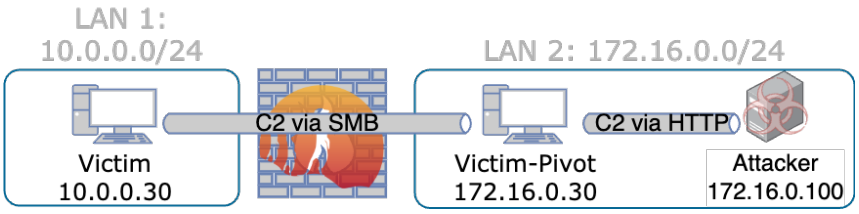

Abbildung 7: Kommunikationsfluss für C2 SMB-Pivot

Zusätzlich ist noch für den Registry-Key `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa` der Value `LimitBlankPasswordUse` vom Typ `REG_DWORD` mit dem Wert 0 angelegt worden[60]. Dies war nötig, da sonst die Erstellung der *Named Pipe* fehlschlägt. Die Dokumentation von Havoc stellte keine Informationen dahingehend zur Verfügung, inwiefern eine tatsächliche Authentifizierung an dem 2. System über SMB möglich ist. Die Erstellung einer sogenannten `NULL Session` wäre ohne die Konfiguration des Registry Keys unterbunden worden.

Attacker

Hierbei handelt es sich um die auf Debian basierende Linux-Distribution `ParrotOS` in Version 6.0 vom 24.01.2024. Für den Datenaustausch wird ein durch VMware Fusion geteilter Ordner verwendet und das Clipboard geteilt.

Dieses System agiert als C2-Server. Dafür sind über GitHub die beiden Anwendungen Sliver[5] vom 31.07.2024 (`127caf5`) und Havoc[35] vom 06.12.2023 (`ea3646e`) installiert worden.

Name Server

Wie in Unterabschnitt 4.2.1 beschrieben, soll die hierarchische DNS-Auflösung realitätsnahe funktionieren. Dies innerhalb des Netzwerkes zu simulieren, würde den nötigen Aufwand übersteigen, weshalb der NS als VM bei dem Hosting Provider Digital Ocean gehostet worden ist.

Durch den Installationsassistenten von Digital Ocean ist Ubuntu 22.04 installiert worden. Außerdem ist durch Digital Ocean ein öffentlicher SSH Schlüssel auf dem System hinterlegt worden, sodass sich über Public-Key-Authentifizierung angemeldet werden kann.

Auf dem Server selbst ist die GitHub Anwendung dnscat2[36] vom 03.01.2022 (`42f8d78`) installiert worden. Für dnscat2 ist Ruby als installierte Abhängigkeit notwendig, welche in Version 3.2.3 vom 18.01.2024 installiert wurde.

Sobald die C2-Server Anwendung gestartet wird, antwortet diese auf gewisse DNS-Queries bzw. auf die Queries des Beacons. Die hierfür infrastrukturellen Voraussetzungen werden in Unterabschnitt 4.2.4 beschrieben.

Suricata

Hierbei handelt es sich um eine VM, auf der die Linux-Distribution Ubuntu 24.04 installiert worden ist. Für den Datenaustausch wird ein durch VMware Fusion geteilter Ordner verwendet und das Clipboard geteilt.

Um die nachfolgende Konfiguration besser nachzuvollziehen, erfolgt an dieser Stelle eine Auflistung der Netzwerkinterface-Bezeichnung:

1. `ens33` Bridge
2. `ens34` `172.16.0.1/24` LAN 2
3. `ens37` `10.0.0.1/24` LAN 1

Suricata in Version 7.0.3 ist gemäß Installationsleitfaden installiert worden[71]. Version 7.0.3 deshalb, weil es die beste Kompatibilität für das ET Open Regelwerk besitzt. Das Snort3 Regelwerk ist nur in kleinen Teilen zu Suricata kompatibel. So sind nur etwa 5 % der insgesamt etwas über 4.000 Regeln der Snort Community Rules erfolgreich geladen worden. Die Konfiguration von Suricata erfolgte in der `/etc/suricata/suricata.yaml`, worin für den korrekten Betrieb erforderlichen Änderungen vorgenommen worden sind:

- Setzen der Variable `HOME_NET` in der Sektion `vars` auf den Wert `10.0.0.0/8`
- Setzen der Variable `interface` in der Sektion `af-packet` auf den Wert `ens37`
- Aktivieren des Regelwerks ET Open (`suricata.rules`) vom 09.08.2024 in der Sektion `rule-files`
- Aktivieren des Regelwerks Snort Community Rules (`snort3-community.rules`) vom 17.08.2024 in der Sektion `rule-files`

Damit die Kommunikation zwischen den LANs möglich ist, ist über die Firewall Anwendung `iptables` die folgenden Regeln angelegt worden, siehe Listing 2.

```
1 Chain FORWARD
2 target     prot opt in     out     source               destination
3 ACCEPT     0    --  ens34  ens33   172.16.0.0/24        0.0.0.0/0
4 ACCEPT     0    --  ens37  ens33   10.0.0.0/24          0.0.0.0/0
5 ACCEPT     0    --  ens37  ens34   10.0.0.0/24          172.16.0.0/24
6 ACCEPT     0    --  ens34  ens37   172.16.0.0/24        10.0.0.0/24
```

Listing 2: `iptables` Firewall-Regeln auf dem System Suricata

Sowie die folgenden NAT-Regeln, siehe Listing 3.

```
1 Chain POSTROUTING
2 target      prot opt in     out     source               destination
3 MASQUERADE  0    --  *      ens33   0.0.0.0/0            0.0.0.0/0
```

Listing 3: `iptables` NAT-Regeln auf dem System Suricata

4.2.4 Autoritative NS

dnscat2 soll über die hierarchische DNS-Auflösung, also kein DNS-Tunnel, agieren. Damit die Anwendung auf dem NS-Server funktionieren kann, waren die nachfolgenden Konfigurationen notwendig.

Zunächst ist bei dem Domänen-Registrar namecheap die Domäne `see-two.xyz` erworben worden. In den Einstellungen der Domäne sind zwei NS-Einträge für `ns1.see-two.xyz` und `ns2.see-two.xyz` gesetzt worden. Beide NS-Einträge zeigen auf die öffentliche IP-Adresse der bei dem Hosting Provider Digital Ocean gehosteten VM, `46.101.223.31`.

Da auf dem NS keine DNS-Zonenkonfiguration vorliegt, ist keine DNS-Query möglich, siehe Listing 5. Dass die Konfiguration aber so Funktional ist und die DNS-Queries an den C2-Server weitergeleitet werden, lässt sich davon ableiten, dass im selben Moment eine Fehlermeldung im Anwendungsprotokoll von dnscat2 erzeugt worden ist, siehe Listing 6. Eine zum selben Zeitpunkt aufgezeichneter Paketmitschnitt, in welchem die DNS-Pakete erscheinen, bestätigt dies.

4.2.5 Erzeugung der Beacons

Wie in Abschnitt 4.1 beschrieben, wird nun die Erzeugung des jeweiligen Beacons erläutert werden.

Sliver

Die Beacons im C2-Framework Sliver lassen sich über die interaktive Kommandozeile der C2-Serveranwendung erzeugen. Im einfachsten Falle lautet diese:
`generate beacon --[Protokoll] [C2-Server]:[Port] --save[Pfad]`
Eine vollständige Auflistung der Befehle zur Erzeugung der Beacons mittels Sliver ist Anhang C zu entnehmen.

Damit der Beacon zum C2-Server kommunizieren kann, muss ein entsprechender Listener gestartet werden. Dieser wird über eine Eingabe des C2-Protokolls, z. B. `http` in der interaktiven Kommandozeile gestartet.

Havoc

Havoc verfügt über eine grafische Benutzeroberfläche, in welcher die Konfiguration vorgenommen wird. Bevor ein Beacon erzeugt werden kann, muss zunächst ein Listener für das entsprechende Protokoll angelegt werden. Der Listener wird an ein Protokoll und an eine IP-Adresse gebunden. Danach kann eine Payload generiert werden. Hierbei können analog zu Sliver diverse Parameter modifiziert werden, in diesem Fall sind die Standardwerte beibehalten worden.

Die Konfiguration des HTTP- und SMB-Beacons unterschieden sich lediglich im ausgewählten Listener. Die Konfiguration der beiden Beacons kann exemplarisch anhand des HTTP-Beacons dem Anhang D entnommen werden.

4.2.6 dnscat2

dnscat2 ist wie Sliver eine Konsolenanwendung. Allerdings muss der Beacon manuell kompiliert werden. Hierzu ist auf dem System `Victim` mittels Visual Studio 2022 die vorgefertigte Projektdatei kompiliert worden. Anders als bei Sliver oder Havoc wird die Konfiguration des Beacons nicht bei der Kompilierung mit in die Binary eingebunden. Die Anwendung muss mit Parametern ausgeführt werden, welche dann die Konfiguration des Beacons steuern. Soll der Beacon z. B. mittels Autoritative NS bzw. einer vollständigen DNS-Auflösung betrieben werden, muss dieser wie folgt gestartet werden:
`dnscat2.exe [Domäne]`
Also:
`dnscat2.exe see-two.xyz`
Der Kommandozeilenaufruf sowie die Etablierung der Sitzung befindet sich in Abbildung 10 des Anhang F.

4.3 Durchführung

In den nachfolgenden Unterabschnitten wird die Detektion durch Suricata für die einzelnen C2-Protokolle beschrieben. Um eine größtmögliche Varianz in dem Netzwerkverkehr zwischen `Victim` und `Attacker` zu erhalten ist auf unterschiedliche Art mit dem Beacon interagiert worden. So ist sichergestellt, dass möglicherweise auch unterschiedliche Detektionsregeln ausgelöst werden könnten. Mit jedem Beacon ist dasselbe Set an Interaktion ausgeführt worden:

- Etablierung der Sitzung
- Keine Interaktion über mehrere Sleep-Intervalle hinweg
- Ausführen von mehreren Befehlen mit unterschiedlicher Antwortgröße wie `pwd`, `dir` oder `pslist`
- Upload zweier Test-Dateien (5 MB und 10 MB) zu `Victim`
- Beenden der Sitzung

4.3.1 Detektion von DNS

Der Beacon wurde wie in Abschnitt 4.1 und Unterabschnitt 4.2.5 mittels des Tools dnscat2 erzeugt und anschließend auf `Victim` ausgeführt.

Der Datenverkehr zwischen `Victim` und dem C2-Server ist durch Suricata nicht detektiert worden.

4.3.2 Detektion von HTTP

Da zwei unterschiedliche HTTP-Beacons ausgeführt worden sind, werden diese getrennt voneinander betrachtet.

Sliver

Der Beacon wurde wie in Abschnitt 4.1 und Unterabschnitt 4.2.5 mittels des C2-Frameworks Sliver erzeugt und anschließend auf `Victim` ausgeführt. Sobald die Sitzung initiiert wird, gibt es durch Suricata die Detektion *SURICATA Applayer Mismatch protocol both directions* mit der Regel SID `2260000`, siehe Listing 7 in Anhang E. Die Nachricht der Detektion zeigt an, dass eine Anomalie, nicht zwangsläufig schadhafter C2-Datenverkehr, erkannt wurde. Der Schalter `applayer_mismatch_protocol_both_directions` weist darauf hin, dass die Protokollidentifizierung von Suricata nicht erfolgreich war, bzw. der Datenverkehr für die zwei Richtungen nicht übereinstimmten. Dies kann auf einen Fehler oder einen Missbrauch des Protokolls hinweisen. Warum dieser Schalter ausgelöst worden ist, konnte nicht ermittelt werden.

Havoc

Der Beacon wurde wie in Abschnitt 4.1 und Unterabschnitt 4.2.5 mittels des C2-Frameworks Havoc erzeugt und anschließend auf `Victim` ausgeführt. Sporadisch kam es zu einer Detektion mit der SID `2045270`, siehe Listing 8 in Anhang E. Die Nachricht der Detektion, *ET MALWARE Havoc Framework Header in HTTP Response*, macht eindeutig klar, dass die Verwendung von Havoc erkannt worden ist. Die Regel selber sucht im Kern nach dem Auftreten eines HTTP X-Headers mit dem Namen `x-havoc`.

4.3.3 Detektion von HTTPS

Der Beacon wurde wie in Abschnitt 4.1 und Unterabschnitt 4.2.5 mittels des C2-Frameworks Sliver erzeugt und anschließend auf `Victim` ausgeführt.

Der Datenverkehr zwischen `Victim` und dem C2-Server ist durch Suricata nicht detektiert worden.

4.3.4 Detektion von SMB

Der Beacon wurde wie in Abschnitt 4.1 und Unterabschnitt 4.2.5 mittels des C2-Frameworks Havoc erzeugt und anschließend auf `Victim-Pivot` ausgeführt. Parallel dazu ist auf `Victim` ein HTTP-Beacon auf `Victim` ausgeführt worden, siehe auch Unterunterabschnitt 4.2.3 zur Funktionsbeschreibung der Pivot-Funktion.

Der SMB-Datenverkehr zwischen `Victim` und `Victim-Pivot` ist durch Suricata nicht detektiert worden.

4.3.5 Detektion von mTLS

Der Beacon wurde wie in Abschnitt 4.1 und Unterabschnitt 4.2.5 mittels des C2-Frameworks Sliver erzeugt und anschließend auf `Victim` ausgeführt.

Der Datenverkehr zwischen `Victim` und dem C2-Server ist durch Suricata nicht detektiert worden.

4.3.6 Detektion von VPN (WireGuard)

Der Beacon wurde wie in Abschnitt 4.1 und Unterabschnitt 4.2.5 mittels des C2-Frameworks Sliver erzeugt und anschließend auf `Victim` ausgeführt.

Der Datenverkehr zwischen `Victim` und dem C2-Server ist durch Suricata nicht detektiert worden.

4.4 Fazit

Die Erkennungsrate der Beacons der C2-Frameworks Sliver (HTTP, HTTPS, mTLS und VPN), Havoc (HTTP, SMB) und dnscat2 ist als gering einzustufen. Bezugnehmend auf die Verteilung unterschiedlicher C2-Protokolle von Sophos[30], kann von schätzungsweise 33 % Erkennungsrate gesprochen werden. Diese Detektionsrate ist nur erreichbar, wenn ausschließlich Havoc für C2 über HTTP verwendet wird. Unter Realbedingungen scheint dies eher unwahrscheinlich, sodass eine geringere Detektionsrate erzielt werden würde.

Bezieht man den Fakt mit ein, dass über 6.000 Regeln zu dem Stichwort C2 vorliegen und vielfältig mit den Beacons interagiert worden ist, scheint dies ernüchternd. Insgesamt scheinen die Regel eher speziell auf gewisse Malware(-Familien) abzuzielen, weniger auf generische Alarme über ungewöhnliche Parameter oder verstöße innerhalb eines Protokolls. Da nur ein sehr kleiner Teil der Snort Community Rules in Suricata importiert werden konnten, lässt sich an dieser Stelle keine Aussage über die Detektionsrate dieses Regelwerks treffen.

Von insgesamt sieben Beacons ist nur einer erkannt worden, bei einem zweiten Beacon kam es zu einer irreführenden, eher auf operative Probleme hindeutende Detektion. Einzig die Detektion von C2 via HTTP mit Havoc war zutreffend. Die genauen Umstände, die dazu führten, dass der detektierte HTTP X-Header vorhanden war, konnten im Rahmen des Experiments nicht bestimmt werden. Unabhängig davon, ob oder wie mit dem Beacon interagiert wurde, kam es nur sporadisch dazu, dass der HTTP X-Header vorhanden war und somit eine Detektion ausgelöst wurde. Damit verfügt die Regel zwar über eine sehr hohe Konfidenz, wird aber nur sporadisch ausgelöst. Unter praktischen Aspekten betrachtet wäre die zweite Detektion wahrscheinlich aufgrund einer erfolgten Priorisierung nicht weiter verfolgt worden.

5 Regelerstellung zur Detektion des C2-Datenverkehrs

Dieses Kapitel wird sich mit der Analyse des C2-Datenverkehrs auseinandersetzen.

In Abschnitt 5.1 wird der mitgeschnittene Datenverkehr näher betrachtet werden, um Ansätze für die Detektion zu identifizieren. Abschnitt 5.2 wird sich mit der praktischen Umsetzung der Detektionsmöglichkeiten beschäftigen. Zuletzt wird in Abschnitt 5.3 ein Fazit über die Implementation der neu erstellten Detektionsregeln gezogen werden.

5.1 Analyse des Datenverkehrs

In den nachfolgenden Unterabschnitten wird der C2-Datenverkehr genauer betrachtet werden. Dieser ist im Rahmen des Experiments, siehe Abschnitt 4.3, mitgeschnitten worden. Ziel ist es an dieser Stelle nicht, den Datenverkehr zwischen C2-Server und dem Beacon durch Reverse Engineering zu entschlüsseln oder zu dekodieren, vielmehr wiederkehrende oder auffällige Muster zu identifizieren, um in Abschnitt 5.2 Regeln für die Detektion erstellen zu können.

5.1.1 Analyse C2-DNS

Zunächst einmal ist festzuhalten, dass das DNS-Protokoll an sich normal verwendet wird. Das System bzw. der Beacon sendet DNS-Queries an seinen konfigurierten DNS-Server. Die eigentliche IP-Adresse des C2-Servers, `46.101.223.31`, tritt zwischen dem Beacon und dem DNS-Server nicht auf. Durch den Beacon werden die folgenden RRs für die Domäne `see-two.xyz` angefragt: TXT, MX und CNAME.

Auffällig ist die Häufigkeit der DNS-Queries. In einem Zeitraum von etwa 600 Sekunden sind über 1.300 DNS-Queries durch den Beacon gesendet worden. Das sind etwas über zwei Queries in der Sekunde, was eigentlich für kurze Peaks nicht auffällig wäre, in der Totale und in Bezug auf eine einzige angefragte Domäne jedoch schon. Je nach Aktivität lassen sich bis zu acht Queries innerhalb einer Sekunde feststellen,

was wiederum auffällig ist. Die nachfolgende Abbildung 8 zeigt sehr eindeutig, in welcher Zeit vermehrt mit dem Beacon interagiert worden ist.

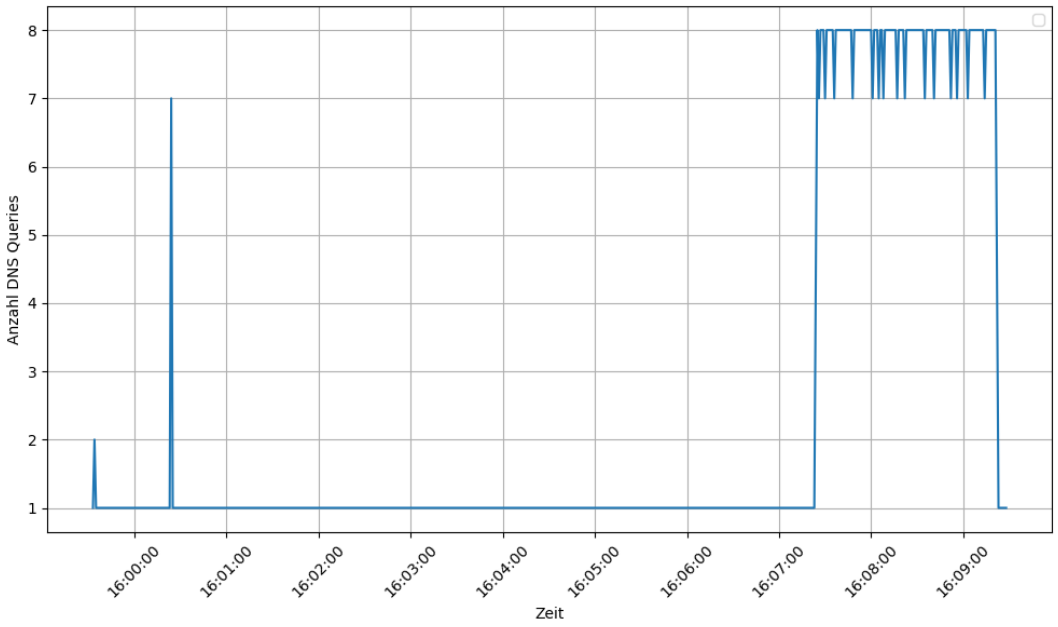

Abbildung 8: Plot der Häufigkeit von DNS Queries durch Victim an see-two.xyz

Auch ist eine auffällige Verteilung der Query Types festzustellen. In dem selben Zeitraum liegt eine nahezu gleiche Verteilung zwischen den RRs TXT, MX und CNAME vor, was eher gegen menschliche Interaktion spricht. Bei dieser wäre von einer eindeutigen dominanz von A-Records festzustellen, welche hier nicht vorliegt.

Bei der Betrachtung der Query Name fällt auf, dass die Subdomänen wechseln und kein lesbarer Name erkennbar ist, z. B.:
`3f05015bf078164cb4fb73048929d9bfb7.see-two.xyz`. Vielmehr scheint die Subdomäne aus hexadezimalen Text zu bestehen, ob zum Informationsaustausch oder nur, um ein caching des DNS-Servers zu vermeiden ist nicht klar.

Eine weitere Auffälligkeit ergibt sich aus der dazugehörigen Query Response. In dem Teil der Response, in welchem die angefragten Informationen enthalten sein sollten (3), befindet sich auch hexadezimaler Text. Dabei wird die Form der Response eingehalten, sodass auf z. B. einen TXT-Query (1) auch eine scheinbar valide TXT-Response (2) zurückgesendet wird, wie in Abbildung 9 zu sehen.

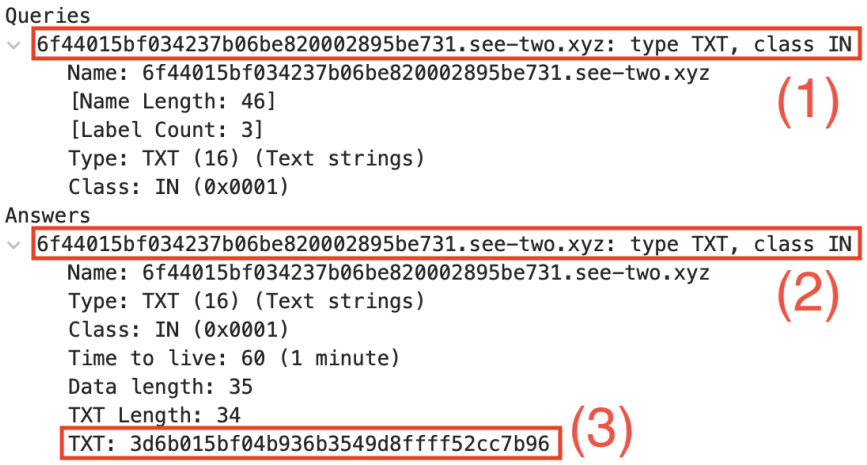

Abbildung 9: Exemplarische DNS Query Response

Abschließend lässt sich noch feststellen, dass die Datenfeld-Länge häufig über dem zu erwartenden Standard liegt. Queries zu TXT-Einträgen, bei denen die Antwort über 200 Zeichen lag, waren im Rahmen des Experiments festzustellen. Ähnlich verhält es sich bei z. B. dem RR CNAME, auch hier hatte die dazugehörige Response zum Teil über 200 Zeichen. Man erkennt hier eine Limitierung des Protokolls, da die RRs auf 255 Zeichen beschränkt sind, was die maximal mögliche Datenmenge pro Paket begrenzt[55, S. 9 f.].

5.1.2 Analyse C2-HTTP

Sliver

Zunächst einmal ist festzuhalten, dass das HTTP-Protokoll an sich normal verwendet wird. Es sind hauptsächlich HTTP `GET` und `POST` festzustellen. Die Endpunkte der angeforderten oder zu übertragenden Daten variieren, sodass es so aussieht, als wenn tatsächlich die Interaktion mit einer Website stattfinden würde. Auch die Tatsache, dass der verwendete Content-Type zum Teil `application/x-gzip` ist, ist durchaus üblich bei der Übertragung größerer Datenmengen.

Ungewöhnlich jedoch ist, dass im Dateninhaltsbereich des HTTP-Pakets fast immer, offenbar kodierter Text enthalten ist. Der angegebene Content-Type ist `text/plain` und dem Charset UTF-8 angegeben. Festzustellen waren die drei folgenden Typen:

- Hexadezimale Zahlen
- Buchstaben, Zahlen und Symbole, Entropie von etwa 5,9
- Passphrase-artig englische Wörter

Die Passphrase-artigen Wörter waren auch in den `POST`-Requests ausgehend vom Beacon festzustellen. Dies passierte sporadisch, aber immer in Verbindung mit der Tatsache, dass der Beacon Befehle ausgeführt hat und Informationen zurück an den C2-Server überträgt. Laut einem Artikel aus 2023[10], handelt es sich um unterschiedlichen Encoding-Methoden. Deshalb ist die Entropie des Textes mit 5,9 relativ hoch und liegt etwa zwischen normalem Text (4-5) und komprimierten (7-8) oder verschlüsselten Inhalten (8). Im Falle der Klartext Wörter handelt es sich um eine hart kodierte Liste, bestehend aus 1420 englischen Wörtern.

Neben einer Häufung gewisser HTTP Pfade waren außerdem unterschiedliche Cookie-Header festzustellen. Darunter `PHPSESSID`, `SSID` und `AWSALBCORS`. Dieselben und weitere durch Sliver verwendete Namen im Cookie-Header sind auch dem Artikel zu entnehmen[10]. Allerdings handelt es sich bei allen festgestellten Namen auch um real existierende Cookie-Namen, welche im Rahmen legitimer Anwendungen vorkommen könnten. `PHPSESSID` bei PHP-Webanwendungen, `SSID` häufig in Verbindung mit Google-Diensten und `AWSALBCORS` bei Amazon Web Services (AWS). Inwiefern eine Regel auf diese Namen zu False-Positive-Alarmen führen würde, müsste in einer realen Umgebung festgestellt werden.

Havoc

Anders als es bei Sliver der Fall gewesen ist, fallen bei der Havoc-Implementation von C2 via HTTP gleich mehrere Punkte auf.

Bei einer normalen Interaktion mit einer Webseite sind unterschiedliche HTTP Methoden wie `GET` und `POST` zu erwarten. In diesem Fall existierten ausschließlich `POST`-Requests, welche widerum mit einem HTTP `200` (OK) bestätigt werden. Das im Paketmitschnitt festgestellte Verhalten änderte sich auch nicht bei der Interaktion mit dem Beacon.

Eine weitere Auffälligkeit ist die Tatsache, dass ungewöhnliche Standardwerte durch den Beacon verwendet werden, wenn der Listener nicht speziell konfiguriert wurde. Hier können Header-Parameter sowie Uniform Resource Identifier (URI) frei definiert werden und könnten somit legitimen und unauffälligen Parametern entsprechen. Passiert dies nicht, fallen die beiden folgenden HTTP Request Parameter auf:

- HTTP `POST` an `/`
- HTTP `POST` Content-Type `*/*`

5.1.3 Analyse C2-HTTPS

Im Rahmen der Untersuchung des HTTPS-Datenverkehrs haben sich keine ungewöhnlichen Merkmale erkennen lassen. Die Kommunikation sieht, wie zu erwarten aus: Aufbau des TCP-Tunnels, Aufbau des TLS-Tunnels, TLS verschlüsselter Datenverkehr.

Sliver verwendet TLS 1.3, sodass anders, wie bei TLS 1.2 und älter die Zertifikate nicht mehr im Klartext übertragen werden.

5.1.4 Analyse C2-SMB

Im Rahmen der Untersuchung des SMB-Datenverkehrs haben sich keine ungewöhnlichen Merkmale erkennen lassen. Die Kommunikation entspricht dem SMB-Standard.

Nachdem die Sitzung initiiert wurde, wird ein `Tree Connect Request` auf die Freigabe `\\172.16.0.30\IPC$` versendet. Diese Freigabe ist für unter anderem die Verwendung von *Named Pipes* genutzt. Danach wird ein `Create Request File` versendet, aus diesem geht der durch den Angreifer gewählten Name der Named Pipe hervor, welcher hierdurch erstellt wird. Im Rahmen des Experiments wurde der Name `smbicious` gewählt. Danach findet die zu erwartende Interprozesskommunikation (IPC) in Form von Write-Requests und -Responses sowie Ioctl Requests und Responses statt im Kontext der Pipe statt.

Ein eher generischer Ansatz würde sich auf die Detektion für die Verwendung der Freigabe `IPC$` beziehen.

Als zweite oder weitere Möglichkeit könnte sich auf den Namen der Pipes beziehen.

Beide Methoden wären kein eindeutiger Hinweis auf die Verwendung von C2 via SMB, aber mögliche Detektionen für zu überprüfende Aktivitäten innerhalb des Netzwerks.

5.1.5 Analyse C2-mTLS

Ähnlich wie bei der Untersuchung des HTTPS Datenverkehrs waren auch hier keine ungewöhnlichen Merkmale festzustellen.

5.1.6 Analyse C2-VPN (WireGuard)

Die mitgeschnittenen Pakete haben keine Hinweise auf die Verwendung von WireGuard gegeben. Wireshark interpretiert diese aufgezeichneten Pakete als DNS-Pakete, jedoch mit dem Flag *malformed*, also stark vom Standard abweichend oder gar den Standard missachtend. Gemeinsam ist, dass als Transportprotokoll UDP verwendet wird, die aufgezeichneten Pakete jedoch ähneln nicht denen einer klassischen WireGuard VPN Sitzung. Offenbar wird das WireGuard-Protokoll innerhalb eines direkten Punkt-zu-Punkt DNS-Tunnels getunnelt. Informationen, die diese These stützen würden, war der offiziellen Dokumentation von Sliver nicht zu entnehmen. Da die vermeintlichen DNS-Queries stark abweichen, fallen deshalb gleich mehrere Punkte auf, welche zur Detektion verwendet werden könnten.

Die DNS Query ID variiert nur zwischen einigen wenigen Zahlen, `0100`, `0200` und `0400`. Typischerweise verändert sich die Transaction ID, um zuordenbar zu sein. Auch ist die Wahrscheinlichkeit sehr gering, dass in einer zeitlichen nähe von wenigen Sekunden dieselbe ID wiederverwendet wird.

Der DNS Query Count bzw. `QDCOUNT` enthält sehr große Werte. Dem DNS Standard nach ist dies zwar ein 16-Bit Integer (0 - 65535), aus der Praxis heraus wären mehrere Tausend Einträge in der nachfolgenden Question Sektion sehr ungewöhnlich[55, S. 27]. Gleiches gilt im Antwortbereich für den DNS Answer Count, welcher ungewöhnlich hohe Werte enthält.

Der DNS Query Name enthält in Kombination mit der Query ID `0400` ausschließlich den Wert `<Root>`. Möglicherweise handelt es sich dabei um eine Fehlinterpretation von Wireshark, da nur ein Byte den Wert `00` hat. In diesem Feld sollte eigentlich der Name oder Fully Qualified Domain Name (FQDN) stehen, der angefragt werden soll. `Root` bzw. ein leeres Byte ist als äußerst ungewöhnlich zu betrachten.

Die DNS Query Type enthält den Wert `00 00` und entspricht somit keinem der zu erwartenden Werte, nämlich 1 - 17[55, S. 11].

Die DNS Query Class enthält andere Werte als 1 (`IN`). Dem DNS Standard entsprechend existieren zwar noch drei weitere Werte, welche in der Praxis jedoch eher irrelevant sind. Eine DNS Query Class die von 1 abweicht ist somit auffällig[55, S. 12].

5.2 Regelerstellung

In den nachfolgenden Unterabschnitten werden mögliche Regeln für die identifizierten Anomalien aus Abschnitt 5.1 beschrieben und deren Implementation dargestellt.

Alle Regeln sind analog zur ersten Durchführung in Abschnitt 4.3 erneut getestet worden, um sicherzustellen, ob diese korrekt erstellt worden sind. Die nachfolgend neu verfassten Detektionsregeln sind also gegen die identische C2-Kommunikation getestet worden.

Die Regeln für DNS und VPN (WireGuard) sind im Rahmen einer 10-minütigen Sitzung, in welcher normales Internet Browsing auf `Victim` simuliert wurde, getestet worden. Im Rahmen dieser Sitzung sind unterschiedliche Webseiten aufgerufen worden, unter anderem die Kategorien: Shopping, News und Blogs, Streaming.

Für die Regeln für die eine eigene Simulation nicht möglich war, wird eine Abschätzung zur *false positive* Rate der Regel getroffen werden.

5.2.1 Detektion von C2-DNS

Wie in Unterabschnitt 5.1.1 beschrieben, werden in den DNS-Queries und -Answers genutzt, um den C2-Datenverkehr zu tunneln. Das Protokoll wird hierbei wie vorgesehen genutzt, zeigte aber jedoch Auffälligkeiten, welche gegen eine legitime Verwendung sprechen.

Query-Count

Die häufige Verwendung von DNS-Queries ist ein guter Indikator für die Verwendung von C2 via DNS. Da die Datenmenge verglichen zu anderen Protokollen relativ gering ist, wird dies über eine gewisse Häufigkeit der Datenübertragung kompensiert. Eine relativ einfache Implementierung, welche auf die Häufigkeit der Queries eines einzelnen Systems abzielt, kann Listing 9 in Anhang F entnommen werden. Diese Regel könnte sich ggf. noch durch Lua-Scripting dazu erweitern lassen, die Queries pro Domäne näher analysieren zu können.

In einer wirklichen Umgebung müsste die eigene Domäne vermutlich von dieser Regel explizit ausgeschlossen werden, um False Positives zu vermeiden. Terminal-Server sollten ggf. auch von dieser Regel ausgeschlossen werden, da das Vielfaches eines einzelnen Nutzers ans DNS-Queries zu erwarten ist. Im Rahmen einer 10-minütigen

Sitzung, in welcher normales Internet Browsing simuliert wurde, ist die Regel nicht ausgelöst worden.

DNS-Query Length

Eine Statistik, auf die sich auch in den letzten Jahren, wenn auch zum Teil ungenannt, noch unterschiedliche Quellen beziehen, ist inzwischen aus 2009[79]. Diese besagt, dass die durchschnittliche Domänenlänge der Top 1.000.000 Millionen Webseiten 10,1 Zeichen lang ist. Inwiefern es um Domänennamen oder Subdomänen geht, wird leider nicht deutlich. Aus der Security-Perspektive ist dies aber eine wichtige Information, um einschätzen zu können, wie lang DNS-Queries möglicherweise sein können. Splunk hat hierzu eine Regel verfasst und geht dabei von der zweifachen Standardabweichung aus, also 25 Zeichen[57]. Die dazugehörige Suricata Detektionsregel kann Listing 10 in Anhang F entnommen werden.

Aufgrund der oben dargestellten Statistik, ist davon auszugehen, dass sich die False Positive Rate, auf ein Minimum beschränken wird. Die Konfidenz der Regel ist relativ hoch, entweder handelt es sich um C2-Kommunikation oder aber auch möglicherweise Datenexfiltration. Im Rahmen einer 10-minütigen Sitzung, in welcher normales Internet Browsing simuliert wurde, ist die Regel nicht ausgelöst worden.

DNS-Query Response Length

Analog zur DNS Query Length waren die durch den C2-Server gesendeten DNS Responses Auffälligkeiten in Bezug auf die Größe der Antwort. Antworten unter 100 Bytes konnten im Rahmen des Experiments nicht festgestellt werden, dieser Wert wird in einer ähnlichen Detektion durch Splunk verwendet[78].

Mangels Suricata Keywords für die DNS Response, musste auf das allgemeinen Payload Keyword `dsize` zurückgegriffen werden. Hierbei handelt es sich nicht nur um die reine Response, sondern die Größe des gesamten DNS-Teils innerhalb des Paketes. Der Wert ist somit nicht akkurat und müsste ggf. nach oben korrigiert werden.

Die Implementation der Suricata Regel kann Listing 11 in Anhang F entnommen werden.

Die False Positive Rate dürfte sich auf ein Minimum beschränken. Wie beschrieben wird der gesamte Payload des Pakets betrachtet, nicht nur explizit die Response, daher ist wahrscheinlich, dass dieser Wert noch einmal angehoben werden müsste, um auch längere DNS Responses nicht zu erfassen. Um diesen Effekt abzufedern

ist auch hier eine Threshold gesetzt worden, da durch die zyklische Kommunikation eine zufällige Häufung eher unwahrscheinlich scheint. Die Konfidenz ist relativ hoch, dass es sich hierbei um C2-Kommunikation handelt, auch deshalb, weil es um eingehende Pakete geht. In diesem Kontext kann eine Datenexfiltration nicht zutreffen. Im Rahmen einer 10-minütigen Sitzung, in welcher normales Internet Browsing simuliert wurde, ist die Regel nicht ausgelöst worden.

5.2.2 Detektion von C2-HTTP

Der Datenverkehr zwischen dem Sliver-Beacon und dem C2-Server waren weitestgehend dem Standard entsprechend. Die Havoc-Implementation des HTTP Beacons eher ungewöhnlich. Im Folgenden werden die praktische Umsetzung der in Unterabschnitt 5.1.2 identifizierten Ansätze diskutiert.

Passphrase Wörterbuch

Die sehr auffälligen Passphrase-artigen Wörter, welche in Verbindung mit `POST`-Requests festgestellt worden sind, waren in der technischen Umsetzung, ohne Kenntnis des gesamten Wörterbuchs nur behelfsweise umzusetzen.

So wird in der aktuellen Implementierung, siehe Listing 12 in Anhang G, auf den Hexadezimal-Werten der fünf exemplarischen Wörtern geprüft. Hierbei handelte es sich um `ICTERICAL`, `STARFISH`, `GUESTING`, `OVERORGANIZE` und `CONFERMENTS`. Diese Wörter sind bei einer Häufigkeitsanalyse einer Stichprobe von einigen wenigen Requests ermittelt worden. In einer zweiten Stufe könnte man entweder durch größere Stichproben bessere Ergebnisse erhalten. Da der Quellcode von Sliver Open Source ist, steht auch das verwendete Wörterbuch prinzipiell zur Verfügung[68]. Im Rahmen des Experiments ist innerhalb weniger Interaktionen mit dem Beacon bereits alarmiert worden, was für einen guten Ansatz spricht. Eine weitere Möglichkeit wäre, die Detektionslogik in ein Lua-Skript auszulagern, sodass die Hexadezimale Darstellung der Wörter auch einfacher zu pflegen ist, wie in dem aktuell benutzten PCRE.

Die False Positive Rate dürfte sehr gering ausfallen. Um den Vektor zusätzlich einzugrenzen und ein zufälliges Vorkommen der Wörter besser ausschließen zu können, wird neben der verwendeten HTTP-Methode zusätzlich auf den verwendeten URI geprüft.

Pfade und Cookies

Wie in Unterabschnitt 5.1.2 beschrieben, war eine Häufung gewisser HTTP-Pfade sowie Cookies festzustellen. Prinzipiell kann mit Suricata auf diese Parameter hin geprüft werden, sodass eine technische Implementation dieser Regel möglich wäre.

Hierbei wäre allerdings mit einer sehr hohen Anzahl an False Positive Detektionen zu rechnen. Sowohl die Pfade als auch die Cookies sind sehr generisch und können innerhalb eines legitimen Kontextes vorkommen. Aufgrund der fehlenden Sinnhaftigkeit ist von einer Implementation abgesehen worden.

POST-only

Die nachfolgende Regel soll nur matchen, wenn ausgehend von einem System ausschließen HTTP POST Requests versendet werden. Hierfür sollte zunächst die Flowbits Funktion verwendet werden, um sicherzustellen, dass nicht auch andere HTTP Methoden außer POST vorkommen. Die Flowbits stellt in Suricata die Möglichkeit bereit, auffällige Ereignisse innerhalb einer Session zu überprüfen und Merkmale oder Regeln kaskadieren. Aus nicht nachvollziehbaren Gründen endet der Flow bereits, sobald der Beacon die Antwort HTTP `200` durch den C2-Server erhält. Gleichzeitig ist in dem HTTP POST Request der Verbindungstyp `Keep-Alive` gesetzt, sodass die Verbindung nicht als beendet betrachtet werden sollte. Es ist unklar ob es sich hierbei um ein Fehlverhalten oder Fehlkonfiguration von Suricata handelt oder ob eine Eigenheit der Havoc-Implementation von HTTP zu diesem Verhalten führt.

Die Implementation kann Listing 13 in Anhang G entnommen werden. Die ersten zwei Regeln dienen der Flowbits Identifikation, die Dritte Regel korreliert die Flowbits.

Das ausschließliche Vorkommen von HTTP POST ist zwar ungewöhnlich, jedoch nicht zwangsläufig C2-Kommunikation zuzuordnen. Hierbei könnte es sich auch um Datenexfiltration handeln, aber auch im legitimen Kontext bei der Verwendung von APIs vorkommen, da hierbei typischerweise auch eine häufung von POST-Requests erzeugt werden.

Aufgrund der nicht festzustellenden Ursache, ist davon auszugehen, dass die Regel False Positive Detektionen verursacht. Auch könnte die Verwendung von APIs False Positive Detektionen verursachen. Diese Netzbereiche, sofern bestimmbar, könnten von der Regel ausgenommen werden, um eine höhere Konfidenz zu erlangen.

Auffällige Standardwerte in Havoc

Im Rahmen der Analyse konnten ungewöhnliche Standardparameter innerhalb des HTTP POST Requests festgestellt werden. Die dazugehörige Regel zur Detektion dieser Standardparameter kann Listing 14 in Anhang G entnommen werden.

Das die Regel False Positive Detektionen verursacht ist unwahrscheinlich. Sowohl die POST URI auf `/`, als auch der Content-Type `*/*` sind sehr unwahrscheinlich. Um etwaige ungewöhnliche Webserver Konfigurationen auszuschließen, sind beide Parameter in der Regel verbunden worden. Ungewöhnliche Standardwerte, welche durch das C2-Framework oder OST gesetzt werden, gelten in der Regel als guter Indikator. Es kann auch mit einer relativ hohen Konfidenz davon ausgegangen werden, dass diese Detektion auf das C2-Framework Havoc zurückzuführen ist.

5.2.3 Detektion von C2-HTTPS

Der Datenverkehr selber zeigte keine Merkmale, auf welcher eine Regel hin überprüfen könnte.

Wie in Unterabschnitt 5.1.3 beschrieben verwendet Sliver TLS 1.3, sodass die Zertifikate nicht mehr im Klartext übertragen werden. So basierten Detektionen vor TLS 1.3 unter anderem darauf, auf selbstsignierte Zertifikate oder nicht vertrauenswürdigen Ausstellern zu prüfen[31]. Dies ist nunmehr nicht möglich.

Ein möglicher Ansatz wäre es, den Datenverkehr zu entschlüsseln, sodass Suricata diesen untersuchen könnte. Aus zeitlichen Gründen und dem Kontextbezug der Studienarbeit wird dieser Ansatz nicht weiter verfolgt. Andere, alternative Ansätze wie z. B. das Fingerprinting mittels JA3/JA3S liegen außerhalb des betrachteten Kontextes einer NIDS und werden deshalb nur am Rande erwähnt.

Detektionen wären so eher im Kontext einer generischen Domänen-/IP-Reputationsprüfung, Anomalieerkennung oder mittels CTI möglich. Mangels Darstellung eines komplexen Netzwerks inklusive des Normalzustands-Baselinings kann dies im Rahmen der Studienarbeit nicht weiter verfolgt werden.

5.2.4 Detektion von C2-SMB

Wie in Unterabschnitt 5.1.4 beschrieben, werden in SMB implementierte Funktionen genutzt um den C2-Datenverkehr zu tunneln. Die in diesem Unterabschnitt dargestellten Regeln sind eher generisch zu betrachten, ein eindeutiger Bezug zu C2-Datenverkehr lässt sich nicht herstellen.

IPC$

Die Verwendung der IPC$-Freigabe lässt sich durch Suricata feststellen. Die dazugehörige Regel zur Detektion kann Listing 15 in Anhang H entnommen werden.

Hierbei müssten in einem realen Szenario Quelle und Ziel im Rahmen einer kontextuellen Betrachtung festgelegt bzw. als möglicherweise schadhaft klassifiziert werden, da die Freigabe auch für legitime Zwecke wie z. B. für Remote Procedure Call (RPC) verwendet wird.

Als weitere Möglichkeit könnte sich die Detektion auf eine vermehrte Verwendung bzw. einem Schwellwert für die Verwendung dieser Freigabe beziehen. Dies wäre entweder ein Hinweis darauf, dass das System als Brückenkopf verwendet wird oder von diesem System SMB-Scanning zu Discovery-Zwecken ausgeht. Die dazugehörige Regel zur Detektion kann Listing 16 in Anhang H entnommen werden.

Bekannt-maliziöse Named Pipe Namen

Ein weiterer, eher generischer Ansatz, jedoch wahrscheinlich mit weniger False Positive Detektionen, ist eine Regel, die auf bekannt-maliziöse Named Pipes prüft. Eine exemplarische Regel, welche lediglich eine der in GitHub aufgelisteten Named Pipes enthält[53], ist Listing 17 in Anhang H zu entnehmen. Die Regel soll lediglich die Möglichkeit und Umsetzung aufzeigen.

Anzumerken ist an dieser Stelle, dass die Regel SID `1000053` möglicherweise auch ausreichen würde. Dieses Beispiel soll die praktische Anwendung von Flowbits, einer Möglichkeit in Suricata, auffällige Ereignisse innerhalb einer Session zu überprüfen und zu kaskadieren. Ohne Flowbits kann aber tatsächlich nicht festgestellt werden, was `psexec` in diesem Moment ist, so könnte es sich auch um einen Share-Namen oder ähnliches handeln. Durch die vorgelagerten Regeln wird dieser Umstand insofern geprüft, dass es sich auch wirklich um eine Pipe handelt. Zwar existiert kein näherer Bezug als die Session selber, also kein definierbarer Abstand zwischen den

Ereignissen bzw. den jeweiligen Paketen, jedoch passieren diese in direkter Abfolge, sodass False Positives unwahrscheinlich scheinen.

Ein anderer, eher konservativer Ansatz könnte es sein, legitime Named Pipes herauszufiltern und jede detektierte Pipe zu betrachten. Dieser Ansatz würde höchstwahrscheinlich zu vielen False Positive Detektionen führen. Diese Regel ist aufgrund des eingeschränkten Fokus des Experiments nicht praktisch umgesetzt worden.

5.2.5 Detektion von C2-mTLS

Der Datenverkehr selber zeigte keine Merkmale, auf welcher eine Regel hin überprüfen könnte.

Dadurch, dass der Datenverkehr im Falle von mTLS nicht entschlüsselt werden kann, würde auch so ein Ansatz scheitern.

Detektionen wären so eher im Kontext einer generischen Domänen-/IP-Reputationsprüfung, Anomalieerkennung oder mittels CTI möglich. Mangels Darstellung eines komplexen Netzwerks inklusive des Normalzustands-Baselinings kann dies im Rahmen der Studienarbeit nicht weiter verfolgt werden.

5.2.6 Detektion von C2-VPN (WireGuard)

Wie sich zeigte, ist der vermeintliche DNS-Datenverkehr derart malformed, dass Suricata diesen nicht als DNS erkennt. Die Detektion basieren somit auf den UDP-Keywords statt DNS-Keywords.

DNS Query ID

Da Suricata aus Gründen der Geschwindigkeit kein echtes Regex zulässt, mussten für die drei festgestellten Transaction IDs getrennte Regeln erstellt werden. Ein immer mindestens 2 Byte langer Content-Filter hätte nur auf jeweils einen der drei Werte zugetroffen, was wiederum das Regex überflüssig gemacht hätte. Speziell Transaction ID `0200` wurde ausgehend vom C2-Server zum Beacon verwendet, anders als `0100` und `0400`, welche vom Beacon zum C2-Server verwendet wurden. Deshalb musste eine getrennte Regel erstellt werden, da Source- und Destination-Port in einer gemeinsamen Regel ohnehin nicht zugetroffen hätte.

Die erstellten Regeln können Listing 18 in Anhang I entnommen werden.

Die erstellten Regeln sind nicht anfällig für False Positive Detektionen. Transaction IDs sind zufallsgeneriert um möglichst wenige Kollisionen zu erzeugen, dass zufällig genau einer der drei festgestellten Transaction IDs verwendet wird könnte durchaus geschehen. Die Regeln selber prüfen jedoch zusätzlich auf einen Schwellwert, welcher nach einer zufälligen Anfrage nicht zutreffen dürfte, da hier wie durch den Beacon wiederholt dieselben Transaction IDs verwendet werden.

Question- und Answer Count

Arithmetische Vergleiche sind in diesem Kontext nicht ohne weiteres durch eine klassische Regel zu erfassen. Hierfür könnte Lua-scripting verwendet werden, welches auch für Detektionen zur Verfügung steht.

Innerhalb des Lua-Skriptes könnte auf die hexadezimal Werte der Questions und Answers geprüft werden. Sollte ein Wert von z. B. 100 überstiegen sein, würde das Skript ein `True` zurückgeben, was wiederum zur Folge hätte, dass die Detektion gemeldet wird.

Questions und Answers hatten häufig Werte über 10.000, welcher sich deutlich von einer echten DNS-Query sowie die Response unterscheidet. Einige Dutzende wären wahrscheinlich schon als ungewöhnlich zu betrachten.

Eine weitere Implikation ist die benötigte Performance, durch die hier geplante Lua Detection. Eine Filterung ist aufgrund des fehlenden Arithmetischen Vergleichs nicht gegeben, so würde jedes DNS-Paket durch das Lua-Skript geprüft werden. Inwiefern dies einen spürbaren Einfluss auf die Auslastung des Systems hat, müsste unter realen Bedingungen geprüft werden. Möglicherweise würde dies alleine die Detektion unbrauchbar machen.

Leerer DNS Query Name

Der DNS Query Name steht immer an derselben Stelle, ab Byte 54, auch bei legitimen DNS-Queries.

Auf das leere Byte hin wird mit einem Vergleich auf das zutreffende Byte geprüft. Die erstellte Regel kann Listing 19 in Anhang I entnommen werden.

Im Rahmen des Experiments haben sich keine False Positives gezeigt, was auch als unwahrscheinlich gilt, da in der Hexadezimal-Darstellung des Query Names das erste Byte der Längenbezeichner ist[55, S. 9].

Leerer DNS Query Type

Auf einen weiteren leeren Wert, in diesem Fall `00 00` hin zu überprüfen, wäre möglich. Die dazugehörige Regel ist in Listing 20 in Anhang I dargestellt.

Diese Regel ist allerdings nicht sinnvoll. Sie trifft zwar auf alle DNS-Pakete zu, die durch den Sliver Beacon erzeugt werden, jedoch befindet sich dieses Feld nach dem DNS Query Name. Anders als die vorherigen Felder handelt es sich beim Query Name um ein variables Feld, sodass diese Regel bei legitimen DNS-Paketen in einem beliebigen anderen Teil prüfen würde. False Positives sind daher nicht auszuschließen und sogar sehr wahrscheinlich.

Der Hauptgrund hierfür ist, dass wie zu Beginn des Abschnitts erwähnt die Pakete so malformed sind, dass Suricata das Protokoll nicht erkennt. Ein Rückgriff auf die UDP Keywords hat somit zur Folge, dass nicht mehr dynamisch auf DNS-Felder zugegriffen werden kann.

Non-IN Query Class

Auf das nicht-zutreffen gewisser Werte zu überprüfen funktioniert etwas anders als bisher gesehen. So gilt alles als auffällig, was nicht der DNS Query Class 1 entspricht. Die dazugehörige Regel ist in Listing 21 in Anhang I dargestellt.

Auch hier gilt dasselbe Problem wie des leeren DNS Query Types. Die Query Class befindet sich nach dem DNS Query Name, weshalb eine Regel zur Detektion auch hier wenig sinnvoll wäre.

5.3 Fazit und Neubewertung

Auf Grundlage des in Kapitel 4 durchgeführten Experiments konnten in Abschnitt 5.1 auffällige Merkmale des Datenverkehrs betrachtet und für die meisten Protokolle in Abschnitt 5.2 eine darauf zutreffende Regeln implementiert werden:

- DNS: 3 Merkmale, 3 Regeln
- HTTP: 4 Merkmale, 3 Regeln
- HTTPS: 0 Merkmale
- SMB: 2 Merkmale, 2 Regeln
- mTLS: 0 Merkmale
- VPN (WireGuard): 5 Merkmale, 2 Regeln

Alle implementierten Regeln haben innerhalb des erneut durchgeführten Experiments den C2-Datenverkehr erkannt. Die Detektionsrate hat sich somit zu dem ursprünglichen Fazit aus Abschnitt 4.4 verbessert. Bezugnehmend auf die Verteilung unterschiedlicher C2-Protokolle von Sophos[30], kann weiterhin von schätzungsweise 33 % Erkennungsrate gesprochen werden. Allerdings sind neue Protokolle hinzugekommen, die von der Statistik nicht erfasst bzw. nicht erwähnt werden wie SMB und DNS. Ähnliche Statistiken, welche aber aktueller oder anderer Quelle entstammen haben sich nicht ermitteln lassen. Daher lässt sich neben der wie oben aufgeführten Detektionsregeln leider keine evidenzbasierte Verbesserung der Erkennungsrate festhalten.

Teilweise wäre eine feinere Detektionslogik über das Lua-Scripting möglich gewesen, dies konnte allerdings im Rahmen der zeitlichen Vorgabe nicht erfüllt werden. Zum Teil waren identifizierte Merkmale auch nicht dazu geeignet, für Detektionsregeln verwendet zu werden, da eine Vielzahl an False Positive Detektionen zu erwarten wäre. Eine Umsetzung dieser weniger sinnvollen Regeln hat deshalb nur im Rahmen einer Darstellung der Regel stattgefunden.

Nicht implementiert werden konnten Regeln für die verschlüsselte C2-Kommunikation über HTTPS und mTLS. Im Falle von HTTPS wäre dies wahrscheinlich möglich gewesen, wenn der verschlüsselte Datenverkehr entschlüsselt worden wäre. Eine Entschlüsselung von mTLS hingegen wäre höchstwahrscheinlich nicht möglich gewesen.

Da für die Protokolle, für die Regeln implementiert werden konnten, 2 oder 3 Regeln zur Verfügung stehen, ist davon auszugehen, dass unter Realbedingungen der C2-Datenverkehr wahrscheinlich auch erkannt werden würde. Nur wenige der Regeln

beziehen sich eindeutig auf C2, da der durch den Angreifer verfolgte Zweck variieren kann. So würde eine Datenexfiltration über DNS vermutlich auch auf Regeln zutreffen, welche sich auf die C2-Kommunikation beziehen. Wiederum ungewöhnliche Standardparameter wie z. B. der HTTP Content-Type `*/*` bei Havoc ist ein relativ gutes Indiz dafür, dass dieses Framework zum Einsatz kommt.

Die Analyse des C2-Datenverkehrs hat gleichzeitig auch gezeigt, wie wichtig es ist, dass unterschiedliche Detektionsmechanismen ineinander greifen, da nicht ein Mechanismus, also z. B. eine NIDS, alles detektieren kann. Abschnitt 5.2 hat vor allem für die Anomaliedetektion gezeigt, dass hierfür an Suricata angebundene Systeme oder eigene Regeln notwendig sind, da Suricata keine Alarme für etwas auslöst, für das keine Regel existiert. Außerhalb bestimmter Ereignisse in der Anwendungsschicht wie z. B. unterschiedliche erkannte Protokolle innerhalb einer Session, müssen die Anomalien durch Regeln erkannt werden.

Ein weiterer, eher negativ aufgefallener Faktor betrifft die Erstellung der Regeln. Suricata deckt zwar viele unterschiedliche und auch gängiger Protokolle ab, jedoch gibt es eine starke Varianz der verfügbaren Keywords. Keywords sind für das jeweilige Protokoll verfügbar und entsprechen im weiteren Sinne den Datenfeldern im Payload des Pakets, so kann z. B. mit `http.cookie` direkt auf die Informationen innerhalb des HTTP Cookie-Header zugegriffen werden. Der Vorteil liegt darin, dass kein Offset innerhalb des Pakets bekannt sein muss, was zum Teil auch nicht möglich ist, wenn Datenfelder variable Längen besitzen. Besonders negativ ist dabei DNS aufgefallen, wo nur die beiden Keywords für `opcode` und die Query existieren. Mehr Keywords, welche auch das DNS Response Paket betreffen, sind erst mit Suricata in Version 8 verfügbar, welche zum Zeitpunkt der Erstellung dieser Arbeit lediglich als Beta existiert. Die Verfügbarkeit von Keywords eines bestimmten Protokolls beeinflusst deshalb die Qualität der Regel. In einigen Fällen musste deshalb auf einen reinen Content-Match innerhalb des Payloads zurückgegriffen werden, welcher je nach Datenfeld nicht so robust gegenüber Varianzen innerhalb des Payloads ist.

Ungeachtet etwaiger negativer Effekte auf die Performance der Suricata NIDS wäre es vorteilhaft, optional verschlüsselten Datenverkehr direkt in Suricata entschlüsseln zu können.

Ein Fazit über die gesamte Arbeit und noch nicht erwähnter Teilaspekte wird im nachfolgenden Kapitel 6 gezogen werden.

6 Ergebnisse, Evaluation und Ausblick

In diesem letzten Kapitel werden die zentralen Ergebnisse des durchgeführten Experiments dargestellt. In Abschnitt 6.1 werden bestehende Grenzen aufgeführt werden. In Abschnitt 6.2 wird ein Fazit über das durchgeführte Experiment sowie die selbst verfassten Detektionsregeln gezogen werden. In Abschnitt 6.3 werden die bestehenden Implikationen näher beschrieben. Abschnitt 6.4 wird zuletzt einen Ausblick über mögliche nachfolgende Arbeiten geben.

6.1 Herausforderungen und Grenzen

Die nachfolgend beschriebenen Herausforderungen und Grenzen haben die vorliegende Arbeit beeinflusst, würden wahrscheinlich aber auch in Teilbereichen nachfolgende Arbeiten, siehe Abschnitt 6.4, betreffen.

Eine Herausforderung betraf vor allem die zeitliche Komponente. Hierbei spielte neben dem nötigen operativen Aufwand auch eine Abgrenzung des Experiments eine Rolle. Neben der Einrichtung des virtuellen Netzwerks war auch die Installation und Konfiguration eines IDS-/IPS-Systems sowie mehrerer C2-Frameworks oder Tools durchzuführen. Aufgrund der hohen Individualität der missbräuchlich verwendeten Protokolle war in der Regel mehr als eine Regel pro Protokoll vonnöten. So zeigte sich, dass zwar ungewöhnliche Merkmale innerhalb des Protokolls identifiziert werden konnten, diese aber nicht unbedingt für eine Detektionsregel zu gebrauchen sind, weil eine relativ hohe *false positive* Rate zu erwarten wäre. Auch war die Regelerstellung in Teilen limitiert, entweder weil das Protokoll selber keine Untersuchung zuließ, wie z. B. mTLS oder VPN, oder weil nicht die nötigen Keywords für die Regel zur Verfügung standen, sodass eine weniger spezifische Regel erstellt werden musste.

Eine Grenze war in dieser Arbeit in monetärer und lizenzrechtlicher Hinsicht. Cobalt Strike, eines der mit am häufigst vorkommenden OST, stand nicht zur Verfügung. Sliver und Havoc verfügen zwar über ähnliche Implementationen der C2-Kommunikation, ein einzelnes OST genauer zu untersuchen bzw. die C2-Kommunikation

alle drei Systeme im direkten Vergleich zu analysieren hätte eine weitere, bisher nicht untersuchte Perspektive gegeben.

6.2 Fazit

Die Auswahl von Open Source NIDS beschränkt sich auf Snort und Suricata, wobei sich Suricata als zukunftsweisender gezeigt hat. Neben dem relativ umfänglichen ET Open Regelwerk existieren auch weitere Regelwerke, welche sich theoretisch einbinden lassen würden, jedoch wäre mit einer immer größeren Anzahl an False Positiv Detektionen zu rechnen. Nach der initialen Implementierung der NIDS müsste deshalb in einer Übergangsphase diese False Positives ausgeklammert oder die zutreffende Regel angepasst werden.

Das in Kapitel 4 durchgeführte Experiment hat gezeigt, dass ET Open zwar über 6.000 Regeln verfügt, welche dem Oberbegriff C2 zuzuordnen sind, sich aber zumindest im Kontext der C2-Kommunikation nur bedingt darauf verlassen werden kann. Über die Snort Community Rules lies sich aufgrund der geringen Anzahl importierter Regeln keine genauere Aussage treffen.

Zusammenfassend lässt sich festhalten, dass mit Suricata eine sehr solide Grundlage für den weiteren Aufbau der IT-Sicherheit im Netzwerk darstellt. Suricata selbst sollte durch unterschiedliche Third-Party Tools erweitert werden, um vollumfänglich von den Daten zu profitieren, die das System bereitstellen kann. Neben einer Oberfläche für das Alarm-Management, wäre auch das Threathunting in den NMS-Daten zu erwähnen, sowie einem Baselining und Analyse der NMS-Daten zur Anomaliedetektion.

6.3 Implikationen

Aus dem Fazit lässt sich ableiten, dass IT-Security Produkte oder Software in der Regel keine Lösung für ein Problem sind, lediglich Werkzeuge, um ein Problem zu lösen.

Etwaige Regeln müssen immer zusätzlich auf die eigene Infrastruktur angepasst, sowie eigene Regeln hinzugefügt werden. Ein weiterer Aspekt in Bezug auf Detektionsregeln, der Suricata oder andere Open Source Lösungen betrifft, ist die Tatsache, dass ein erhöhter Wartungs- und Pflegebedarf besteht. Open Source ist zwar *kostenlos*, bindet gleichzeitig aber auch mehr Ressourcen, verglichen zu kommerziellen

Systemen. Viele Aspekte werden an den Hersteller der NGFW oder NIDS ausgelagert, so muss z. B. keine regelmäßige Evaluation über aktuelle tactics, techniques, and procedures (TTP) erfolgen und diese in Form von Regeln implementieren. Dies passiert automatisch über Pattern-Updates des jeweiligen Herstellers. Das Gleiche gilt für Betriebssystem-Updates, im Falle von Suricata muss z. B. geprüft werden, ob das Update von Paketen eventuell zu Problemen in der Anwendung selber führen könnte.

Eine zunehmende Nutzung von C2-Protokollen wie VPN oder mTLS erschwert eine signaturbasierte Detektion. Hier wird einerseits der zuvor ausgeführte Punkt zur Anomaliedetektion relevant, auf der anderen Seite ist auch eine genaue Kenntnis über das eigene Netzwerk mindestens genauso wichtig. Die Freigabe von Firewall-Regeln nach dem Least-Privilege Prinzip sollte so weit möglich umgesetzt sein, sodass nicht überspitzt dargestellt ein Server in der DMZ eine VPN-Verbindung zu einem unbekannten Server hält. Genauso wichtig ist hierbei die Identifikation der verwendeten Protokolle, weil einfache Portfreigaben nicht mehr dem nötigen Sicherheitsniveau entsprechen. Hierbei kann Suricata als NIDS unterstützen, sollte die eingesetzte Firewall über keine Protokoll- oder Applikationserkennung verfügen.

6.4 Ausblick

Vor allem die Anomaliedetektion ist ein für die Zukunft interessantes Thema, da sich im Verlauf der Arbeit gezeigt hat, dass zumindest in der Vergangenheit die Benutzung verschlüsselter Datenkanäle zugenommen hat. Auch lassen die meisten C2-Frameworks eine gewisse Konfiguration des Beacons zu, was sich wiederum auch zum Teil in der C2-Kommunikation selber auswirkt. Eine Umgehung bestehender Signaturen sind so denkbar. Die Detektion auf Basis von Usecases wie: *Server X kommuniziert plötzlich exzessiv zu einem Server im Internet, den zuvor noch niemand kontaktiert hatte* würden eine Erkennung auch ohne Entschlüsselung der Daten oder Kenntnis der speziellen Implementation des Beacons ermöglichen. Hierfür würde der Rückgriff auf passiv bezogene NetFlow- oder NMS-Daten genügen. Speziell deshalb ist auch das Thema der Künstliche Intelligenz (KI) relevant, da der technologische Fortschritt in diesem Feld auch für eine verbesserte Erkennung führen wird. Derartige Lösungen existieren bereits heute über z. B. Zeek[9] bzw. mit dem Zeek Analysis Tools[39].

Auf der operativen Ebene könnte Suricata weiter betrachtet werden. Wie festgestellt worden ist, fehlen einigen Protokollen Keywords, um sinnvoll Regeln zur Detektion

verfassen zu können. Eine programmatische Erweiterung der Protokolle und der verfügbaren Keywords wäre von Vorteil.

Zuletzt ist zu erwähnen, dass Detektionsregeln in der Regel nicht dazu gedacht sind, für einen Zeitraum von mehreren Jahren zu schützen. Neben sich ändernden TTPs sollten auch regelmäßig C2-Frameworks oder andere OST betrachtet werden, um frühzeitig Regeln verfassen zu können. Eine weitere oder nachfolgende Arbeit könnte sich in einer identischen Ausprägung zu dieser Arbeit, jedoch speziell mit der Anomaliedetektion beschäftigen. Dies vor allem im Fokus auf KI zur Detektion, mangels der Möglichkeit dies innerhalb einer virtuellen Umgebung umzusetzen, müsste dies wahrscheinlich in einem echten Firmennetzwerk implementiert werden. Eine weitere Möglichkeit wäre die C2-Kommunikation von Malware unterschiedlicher Familien aus pcaps zu replayen und die Detektionsraten dahingehend zu überprüfen. Auch wäre ein analoges Experiment zur Detektion von C2-Kommunikation speziell bei kommerziellen Produkten relevant.

A ET Open Regelwerk

```
2351 http
2277 dns
 746 tls
 608 tcp
 408 http1
 110 tcp-pkt
  29 udp
  15 smtp
   5 icmp
   3 ftp
   2 smb
   1 tcp-stream
   1 ip
```

Listing 4: Häufigkeitsverteilung der Protokolle, siehe Unterabschnitt 3.3.1

B dnscat2

DNS-Query an C2 dnscat2 Server:

```
; <<>> DiG 9.10.6 <<>> @8.8.8.8 see-two.xyz
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 50199
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
; OPT=15: 00 16 41 74 20 64 65 6c 65 67 61 74 69 6f 6e 20 73 65 65 2d 74 77
↪  6f 2e 78 79 7a 20 66 6f 72 20 73 65 65 2d 74 77 6f 2e 78 79 7a 2f 61
↪  ("..At delegation see-two.xyz for see-two.xyz/a")
;; QUESTION SECTION:
;see-two.xyz.            IN    A

;; Query time: 2165 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sat Aug 24 17:21:52 CEST 2024
;; MSG SIZE  rcvd: 89
```

Listing 5: DNS-Query an dnscat2

Die DNS-Query erzeugt eine Fehlermeldung im Anwendungsprotokoll:

```
Protocol exception caught in dnscat DNS module (for more information, check window 'dns1'):
#<DnscatException: Received a packet with no questions>
Caught an error: undefined method `serialize' for nil:NilClass
/home/dnscat/dnscat2/server/libs/dnser.rb:678:in `block in serialize'
/home/dnscat/dnscat2/server/libs/dnser.rb:677:in `each'
/home/dnscat/dnscat2/server/libs/dnser.rb:677:in `serialize'
/home/dnscat/dnscat2/server/libs/dnser.rb:819:in `reply!'
/home/dnscat/dnscat2/server/libs/dnser.rb:775:in `error!'
/home/dnscat/dnscat2/server/tunnel_drivers/driver_dns.rb:357:in `rescue in block in initialize'
/home/dnscat/dnscat2/server/tunnel_drivers/driver_dns.rb:293:in `block in initialize'
/home/dnscat/dnscat2/server/libs/dnser.rb:872:in `block (2 levels) in on_request'
/home/dnscat/dnscat2/server/libs/dnser.rb:843:in `loop'
/home/dnscat/dnscat2/server/libs/dnser.rb:843:in `block in on_request'
Caught an error: undefined method `response_template' for #<DNSer::Transaction:0x000075b6fbb825e0 @s=#<UDPSocket:fd 5, AF_INET, 0.0.0.0, 53>, @request=#<DNSer::Packet:0x000075b6fbb828b0 @trn_id=0, @qr=0, @opcode=2, @flags=0, @rcode=0, @questions=[], @answers=[]>, @host="152.32.227.252", @port=38760, @sent=false, @cache={0=>{:request=>#<DNSer::Packet:0x000075b6fbb828b0 @trn_id=0, @qr=0, @opcode=2, @flags=0, @rcode=0, @questions=[], @answers=[]>, :response=>#<DNSer::Packet:0x000075b6fbb82590 @trn_id=0, @qr=1, @opcode=2, @flags=3, @rcode=3, @questions=[nil], @answers=[]>}}, @response=#<DNSer::Packet:0x000075b6fbb82590 @trn_id=0, @qr=1, @opcode=2, @flags=3, @rcode=3, @questions=[nil], @answers=[]>>
/home/dnscat/dnscat2/server/libs/dnser.rb:876:in `rescue in block (2 levels) in on_request'
/home/dnscat/dnscat2/server/libs/dnser.rb:871:in `block (2 levels) in on_request'
/home/dnscat/dnscat2/server/libs/dnser.rb:843:in `loop'
/home/dnscat/dnscat2/server/libs/dnser.rb:843:in `block in on_request'
```

Listing 6: Fehlermeldung in dnscat2 auf valide DNS-Query

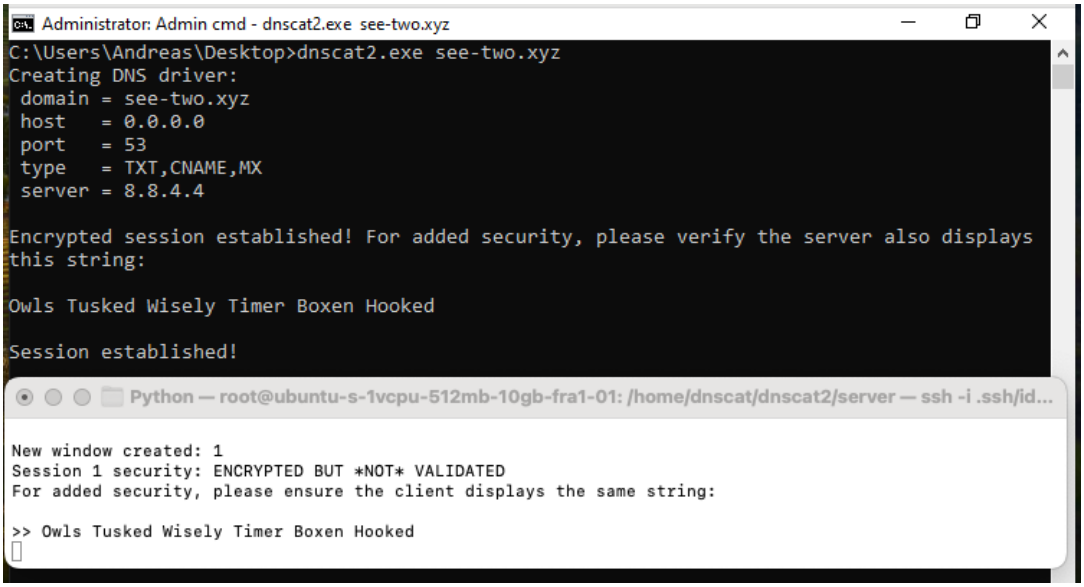

Abbildung 10: Exemplarische Ausführung des DNS-Beacons in dnscat2

C Sliver

Die Erstellung der verwendeten Beacons erfolgte wie folgt:

```
generate beacon --http 172.16.0.100:80 --save /home/andreas/http.exe
generate beacon --http 172.16.0.100:443 --save /home/andreas/https.exe
generate beacon --mtls 172.16.0.100:8888 --save /home/andreas/mtls.exe
generate beacon --wg 172.16.0.100:53 --save /home/andreas/wg.exe
```

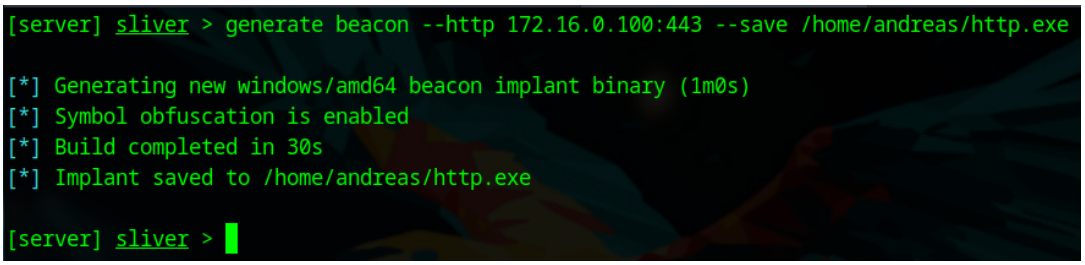

Abbildung 11: Exemplarische Ausgabe von Sliver zur Erstellung eines http-Beacons

D Havoc

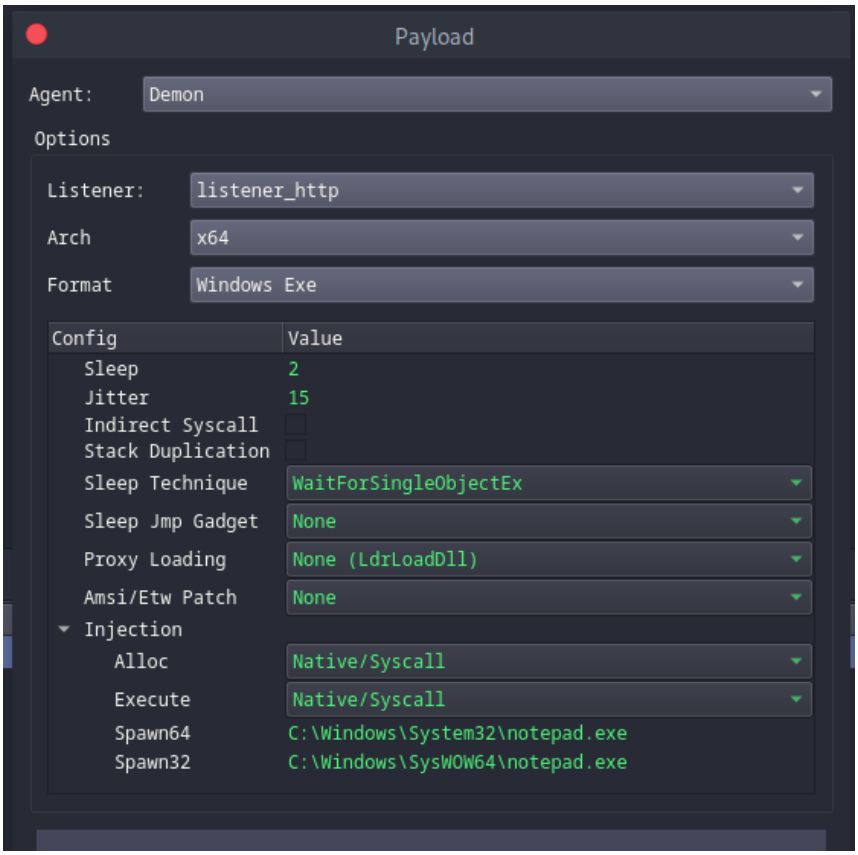

Abbildung 12: Exemplarische Konfiguration eines http-Beacons in Havoc

E Detektionen

Die nachfolgend aufgelisteten Regel zur Detektion beziehen sich auf Abschnitt 4.3.

```
alert ip any any -> any any (msg:"SURICATA Applayer Mismatch protocol both
  ↪ directions"; \
  flow:established; \
  app-layer-event:applayer_mismatch_protocol_both_directions;
  ↪ flowint:applayer.anomaly.count,+,1; classtype:protocol-command-decode;
  ↪ \
sid:2260000; rev:1;)
```

Listing 7: Inhalt der Suricata Regel SID 2260000

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET MALWARE Havoc
  ↪ Framework Header in HTTP Response"; \
  flow:established,to_client; \
  http.header_names; to_lowercase; content:"x-havoc|0d 0a|"; fast_pattern; \
reference:url,github.com/HavocFramework/Havoc/blob/main/Teamserver/profiles/havoc.yaotl;
  ↪ \
reference:url,twitter.com/MichalKoczwara/status/1652986620658761729; \
classtype:trojan-activity; sid:2045270; rev:2; \
metadata:attack_target Client_Endpoint, created_at 2023_05_01, deployment
  ↪ Perimeter, deployment SSLDecrypt, confidence High, signature_severity
  ↪ Major, updated_at 2024_04_26;)
```

Listing 8: Inhalt der Suricata Regel SID 2045270

F Regeln zur Detektion von C2-DNS

Nachfolgend aufgelistete Regeln beziehen sich auf die Detektion von C2 via DNS aus Unterabschnitt 5.2.1.

```
1 alert dns any any -> any 53 (msg:"Multiple Huge amount of DNS-Queries
  ↪ possibel C2-Channel or exfiltration detected"; \
2     dns.opcode:0; \
3     threshold:type both, track by_src, count 30, seconds 60; \
4 sid:1000020; rev:1;)
```

Listing 9: Suricata Regel zur Detektion von erhöhten DNS-Queries durch ein einzelnes System

```
1 alert dns any any -> any 53 (msg:"Multiple DNS Query Length exceeds 25
  ↪ characters, possible C2-Channel or exfiltration detected"; \
2     dns.opcode:0; dns.query; bsize:>25; \
3 threshold:type both, track by_src, count 10, seconds 60; \
4 sid:1000021; rev:1;)
```

Listing 10: Suricata Regel zur Detektion von DNS-Queries mit auffällig langem FQDN

```
1 alert dns any 53 -> any any (msg:"Multiple DNS Answer to single Host exceeds
  ↪ 100 Byte, possible C2-Channel detected"; \
2     dsize:>=100; \
3 threshold:type both, track by_dst, count 10, seconds 60; \
4 sid:1000022; rev:1;)
```

Listing 11: Suricata Regel zur Detektion von großen DNS Responses

G Regeln zur Detektion von C2-HTTP

Nachfolgend aufgelistete Regeln beziehen sich auf die Detektion von C2 via HTTP aus Unterabschnitt 5.2.2.

```
alert http any any -> any 80 (msg:"C2-Sliver: Suspicous Keywords detected in
↪ HTTP POST"; \
    http.method; content:"POST"; \
    http.uri; content:"php"; nocase; \
    file.data; pcre:"/\x49\x43\x54\x45\x52\x49\x43\x41\x4c| \
    \x53\x54\x41\x52\x46\x49\x53\x48| \
    \x47\x55\x45\x53\x54\x49\x4e\x47| \
    \x4f\x56\x45\x52\x4f\x52\x47\x41\x4e\x49\x5a\x45| \
    \x43\x4f\x4e\x46\x45\x52\x4d\x45\x4e\x54\x53/"; \
classtype:trojan-activity; sid:1000400; rev:1;)
```

Listing 12: Exemplarische Suricata Regel zur Detektion von auffälligen durch Sliver verwendete Wörter

```
alert http any any -> any 80 (msg:"HTTP POST flow detected"; \
    flow:to_server,established; \
    http.method; content:"POST"; \
flowbits:set,http_post; flowbits:noalert; \
sid:1000401; rev:1;)

alert http any any -> any 80 (msg:"HTTP GET request in flow detected"; \
    flow:to_server,established; \
    http.method; content:"GET"; \
flowbits:set,non_http_post; flowbits:unset,http_post; flowbits:noalert; \
sid:1000402; rev:1;)

alert ip any any -> any any (msg:"Flow with only HTTP POST requests"; \
    flow:to_server,established; \
flowbits:isset,http_post; flowbits:isnotset,non_http_post; \
threshold:type both, track by_src, count 10, seconds 60; \
sid:1000403; rev:2;)
```

Listing 13: Suricata Regel zur Detektion von ausschließlichen HTTP-POST Verbindungen

```
alert http any any -> any 80 (msg:"C2-Havoc: Suspicious HTTP Default
↪ Parameters"; \
    http.method; content:"POST"; \
    http.uri; content:"/"; urilen:1; \
    http.content_type; content:"*/*"; \
threshold:type both, track by_src, count 10, seconds 60; \
sid:1000404; rev:1;)
```

Listing 14: Suricata Regel zur Detektion von Havoc Standardparametern

H Regeln zur Detektion von C2-SMB

Nachfolgend aufgelistete Regeln beziehen sich auf die Detektion von C2 via SMB aus Unterabschnitt 5.2.4.

```
alert smb any any -> any 445 (msg:"SMB Tree Connect Request to IPC$
↪ detected"; \
    smb.named_pipe; content:"IPC"; nocase; \
sid:10000040; rev:1;)
```

Listing 15: Suricata Regel zur Detektion zur Nutzung der IPC-Freigabe durch SMB

```
alert smb any any -> any 445 (msg:"SMB-Scanning for IPC$ detected"; \
    smb.named_pipe; content:"IPC"; nocase; \
    threshold:type both, track by_src, count 10, seconds 60; \
sid:10000041; rev:1;)
```

Listing 16: Suricata Regel zur Detektion von möglichem SMB-Scanning der IPC-Freigabe

```
alert smb any any -> any 445 (msg:"SMB Tree Connect to IPC$ detected"; \
    smb.named_pipe; content:"IPC"; nocase; \
    flow:to_server,established; \
    flowbits:set,tree_connect_ipc; \
sid:1000050; rev:1;)

alert smb any 445 -> any any (msg:"SMB Share Type 02 (named pipe) detected
    ↪ after IPC$ Tree Connect"; \
    content:"|10 00 02 00|"; \
    flow:to_client,established; \
    flowbits:isset,tree_connect_ipc; \
    flowbits:set,share_type_02; \
sid:1000051; rev:1;)

alert smb any any -> any 445 (msg:"SMB: Create Request File detected"; \
    content:"|00 00 05 00|"; \
    flow:to_server,established; \
    flowbits:isset,share_type_02; \
    flowbits:set,create_file_request_detected; \
sid:1000052; rev:1;)

alert smb any any -> any 445 (msg:"SMB known malicious named pipe 'psexec'
    ↪ detected"; \
    content:"|70 00 73 00 65 00 78 00 65 00 63 00|"; \
    flow:to_server,established; \
    flowbits:isset,create_file_request_detected; \
sid:1000054; rev:1;)
```

Listing 17: Suricata Regeln mit Nutzung von flowbits zur Detektion bei der Nutzung bekannt-maliziöser Named Pipe durch SMB

I Regeln zur Detektion von C2-VPN (WireGuard)

Nachfolgend aufgelistete Regeln beziehen sich auf die Detektion von C2 via VPN (WireGuard) aus Unterabschnitt 5.2.6.

```
alert udp any any -> any 53 (msg:"Sliver: Possible DNS Tunnel detected,
↪ re-use of suspicious dns.id 0100"; \
    content:"|01 00|"; offset:0; depth:2; \
    threshold:type both, track by_src, count 5, seconds 120; \
classtype:trojan-activity; sid:10000025; rev:1;)

alert udp any 53 -> any any (msg:"Sliver: Possible DNS Tunnel detected,
↪ re-use of suspicious dns.id 0200"; \
    content:"|02 00|"; offset:0; depth:2; \
    threshold:type both, track by_src, count 5, seconds 120; \
classtype:trojan-activity; sid:10000026; rev:1;)

alert udp any any -> any 53 (msg:"Sliver: Possible DNS Tunnel detected,
↪ re-use of suspicious dns.id 0400"; \
    content:"|04 00|"; offset:0; depth:2; \
    threshold:type both, track by_src, count 5, seconds 60; \
classtype:trojan-activity; sid:10000027; rev:1;)
```

Listing 18: Suricata Regel zur Detektion ungewöhnlicher DNS Transaction IDs

```
alert udp any any -> any 53 (msg:"Sliver: Possible DNS Tunnel detected,
↪ dns.qry.name empty"; \
    content:"|00|"; offset:12; depth:1; \
    threshold:type both, track by_src, count 5, seconds 60; \
classtype:trojan-activity; sid:10000029; rev:1;)
```

Listing 19: Suricata Regel zur Detektion eines leeren DNS Query Names

```
alert udp any any -> any 53 (msg:"Sliver: Possible DNS Tunnel detected,
↪ dns.qry.type empty"; \
    content:"|00 00|"; offset:13; depth:2; \
    threshold:type both, track by_src, count 5, seconds 60; \
classtype:trojan-activity; sid:10000030; rev:1;)
```

Listing 20: Suricata Regel zur Detektion eines leeren DNS Query Types

```
alert udp any any -> any 53 (msg:"Sliver: Possible DNS Tunnel detected,
↪ suspicious dns.qry.class"; \
    byte_test:2,!=,1,12; byte_test:2,!=,255,12; \
threshold:type both, track by_src, count 5, seconds 60; \
classtype:trojan-activity; sid:10000022; rev:2;)
```

Listing 21: Suricata Regel zur Detektion eines abweichenden DNS Query Class

Literaturverzeichnis

[1] *0. Welcome.md | Havoc Documentation*. 2024. URL: https://havocframework.com/docs/welcome (besucht am 18. 08. 2024).

[2] *11. Pivoting.md | Havoc Documentation*. 2024. URL: https://havocframework.com/docs/pivoting (besucht am 18. 08. 2024).

[3] LockBit 3.0. *LockBit BLOG*. 2024. URL: http://lbb2llze7ab4rnq4jumsy4ihsqzpuysaofpz2e43foocwmrzsokumqid.onion/rules (besucht am 19. 08. 2024).

[4] Aleks, Kostas und Alessandro Di Carlo. *2022 Year in Review - The DFIR Report*. 2023. URL: https://thedfirreport.com/2023/03/06/2022-year-in-review/#command-and-control (besucht am 18. 08. 2024).

[5] BishopFox. *GitHub - BishopFox/sliver: Adversary Emulation Framework*. 2024. URL: https://github.com/BishopFox/sliver (besucht am 18. 08. 2024).

[6] BishopFox. *Sliver Docs:* 2024. URL: https://sliver.sh/docs (besucht am 18. 08. 2024).

[7] BishopFox. *Sliver Docs: DNS C2*. 2024. URL: https://sliver.sh/docs?name=DNS+C2 (besucht am 22. 08. 2024).

[8] Akram Abd Eldjalil Boukebous u. a. „A Comparative Analysis of Snort 3 and Suricata“. In: *Proceedings of the 2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)* (2023).

[9] Erdem Bozdag. *Anomaly-based Intrusion Detection System using unsupervised ML approach*. 2022. URL: https://medium.com/hootsuite-engineering/anomaly-based-intrusion-detection-system-using-machine-learning-a18e88694ce0 (besucht am 01. 09. 2024).

[10] Kevin Breen. *Detecting and decrypting Sliver C2 – a threat hunter's guide*. 2023. URL: https://www.immersivelabs.com/blog/detecting-and-decrypting-sliver-c2-a-threat-hunters-guide/ (besucht am 22. 08. 2024).

[11] Bundeskriminalamt. *BKA - Cybercrime*. 2024. URL: https://www.bka.de/DE/UnsereAufgaben/Deliktsbereiche/Cybercrime/cybercrime_node.html (besucht am 19. 08. 2024).

[12] Bundeskriminalamt. *BKA - Im Fokus: Bundeslagebild Cybercrime 2023*. 2024. URL: https://www.bka.de/DE/AktuelleInformationen/Statistiken Lagebilder/Lagebilder/Cybercrime/2023/CC_2023.html (besucht am 19. 08. 2024).

[13] Tom Clavel. *What Is NetFlow? - Gigamon Blog*. 2023. URL: https://blog.gigamon.com/2018/01/08/what-is-netflow/ (besucht am 15. 08. 2024).

[14] Gerald Combs und Guy Harris. *Capture Setup / Ethernet - Wireshark Wiki*. 2020. URL: https://wiki.wireshark.org/CaptureSetup/Ethernet (besucht am 01. 04. 2024).

[15] The MITRE Corporation. *Application Layer Protocol: Web Protocols, Sub-technique T1071.001 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/techniques/T1071/001/ (besucht am 15. 08. 2024).

[16] The MITRE Corporation. *Credential Access, Tactic TA0006 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/tactics/TA0006/ (besucht am 15. 08. 2024).

[17] The MITRE Corporation. *Defense Evasion, Tactic TA0005 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/tactics/TA0005/ (besucht am 15. 08. 2024).

[18] The MITRE Corporation. *Discovery, Tactic TA0007 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/tactics/TA0007/ (besucht am 15. 08. 2024).

[19] The MITRE Corporation. *Execution, Tactic TA0002 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/tactics/TA0002/ (besucht am 15. 08. 2024).

[20] The MITRE Corporation. *Impact, Tactic TA0040 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/tactics/TA0040/ (besucht am 15. 08. 2024).

[21] The MITRE Corporation. *Initial Access, Tactic TA0001 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/tactics/TA0001/ (besucht am 15. 08. 2024).

[22] The MITRE Corporation. *Lateral Movement, Tactic TA0008 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/tactics/TA0008/ (besucht am 15. 08. 2024).

[23] The MITRE Corporation. *Matrix - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/matrices/enterprise/ (besucht am 15. 08. 2024).

[24] The MITRE Corporation. *Persistence, Tactic TA0003 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/tactics/TA0003/ (besucht am 15. 08. 2024).

[25] The MITRE Corporation. *Privilege Escalation, Tactic TA0004 - Enterprise | MITRE ATT&CK*. 2024. URL: https://attack.mitre.org/tactics/TA0004/ (besucht am 15. 08. 2024).

[26] *Cybersecurity in Deutschland | Statista*. 2024. URL: https://de.statista.com/statistik/studie/id/7320/dokument/internetkriminalitaet-statista-dossier/ (besucht am 01. 04. 2024).

[27] Hervé Debar und Jouni Viinikka. *Foundations of Security and Design III - FOSAD 2004/2005 Tutorial Lectures*. 1. Auflage. Springer-Verlag Berlin Heidelberg, 2005. ISBN: 978-3-540-28955-5.

[28] *ELK Stack*. 2024. URL: https://www.elastic.co/de/elastic-stack (besucht am 30. 09. 2024).

[29] R. Fielding u. a. *RFC 2616 - Hypertext Transfer Protocol – HTTP1.1*. 1999. URL: https://datatracker.ietf.org/doc/html/rfc2616 (besucht am 15. 08. 2024).

[30] Sean Gallagher. *Nearly half of malware now use TLS to conceal communications - Sophos News*. 2021. URL: https://news.sophos.com/en-us/2021/04/21/nearly-half-of-malware-now-use-tls-to-conceal-communications/ (besucht am 01. 04. 2024).

[31] Bala Ganesh. *Finding the Evil in TLS 1.2 Traffic – Detecting Malware on Encrypted Traffic - Security Investigation*. 2021. URL: https://www.socinvestigation.com/finding-the-evil-in-tls-1-2-traffic-detecting-malware-on-encrypted-traffic/ (besucht am 20. 08. 2024).

[32] *GitHub - bammv/sguil: Sguil client for NSM*. 2017. URL: https://github.com/bammv/sguil (besucht am 18. 08. 2024).

[33] *GitHub - BC-SECURITY/Empire*. 2024. URL: https://github.com/BC-SECURITY/Empire (besucht am 18. 08. 2024).

[34] *GitHub - crawl3r/DaaC2*. 2021. URL: https://github.com/crawl3r/DaaC2 (besucht am 18. 08. 2024).

[35] *GitHub - HavocFramework/Havoc: The Havoc Framework*. 2024. URL: https://github.com/HavocFramework/Havoc (besucht am 18. 08. 2024).

[36] *GitHub - iagox86/dnscat2*. 2022. URL: https://github.com/iagox86/dnscat2 (besucht am 20. 08. 2024).

[37] *GitHub - maldevel/gdog*. 2017. URL: https://github.com/maldevel/gdog (besucht am 18. 08. 2024).

[38] *GitHub - rapit7/metasploit-framework*. 2021. URL: https://github.com/rapid7/metasploit-framework (besucht am 18. 08. 2024).

[39] *GitHub - SuperCowPowers/zat: Zeek Analysis Tools (ZAT)*. 2024. URL: https://github.com/SuperCowPowers/zat (besucht am 01. 09. 2024).

[40] Juan Andres Guerrero-Saade. *AcidRain | A Modem Wiper Rains Down on Europe - SentinelOne*. 2022. URL: https://www.sentinelone.com/labs/acidrain-a-modem-wiper-rains-down-on-europe/ (besucht am 18. 08. 2024).

[41] *Home - Suricata*. 2024. URL: https://suricata.io.

[42] Ltd. Huawei Technologies Co. *Data Communications and Network Technologies*. 1. Auflage. Springer, 2023. ISBN: 978-981-19-3029-4.

[43] Malwarebytes Inc. *Cryptojacking - Was ist das, und wie funktioniert es? | Malwarebytes*. 2024. URL: https://www.malwarebytes.com/de/cryptojacking (besucht am 17. 08. 2024).

[44] Fortra LLC. *Appropiate Covert Channels | Cobalt Strike*. 2015. URL: https://www.cobaltstrike.com/blog/appropriate-covert-channels (besucht am 18. 08. 2024).

[45] Fortra LLC. *Cobalt Strike Beacon | Cobalt Strike Features*. 2015. URL: https://www.cobaltstrike.com/product/features/beacon (besucht am 18. 08. 2024).

[46] Fortra LLC. *Cobalt Strike | Adversary Simulation and Red Team Operations*. 2024. URL: https://www.cobaltstrike.com (besucht am 18. 08. 2024).

[47] Fortra LLC. *HTTP Beacon and HTTPS Beacon*. 2024. URL: https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/listener-infrastructure_beacon-http-https.htm (besucht am 18. 08. 2024).

[48] Fortra LLC. *SMB Beacon*. 2024. URL: https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/listener-infrastructure_beacon-smb.htm (besucht am 18. 08. 2024).

[49] Fortra LLC. *Welcome to Cobalt Strike*. 2024. URL: https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/welcome_main.htm (besucht am 18. 08. 2024).

[50] Shiva Mandalam. *Real-world Examples Of Emerging DNS Attacks and How We Must Adapt*. 2021. URL: https://www.paloaltonetworks.com/blog/2021/05/netsec-dns-attacks/ (besucht am 01. 04. 2024).

[51] Sebastiano Mariani. *It's Raining Implants: How to Generate C2 Framework Implants At Scale - VMware Security Blog - VMware*. 2023. URL: https://blogs.vmware.com/security/2023/04/its-raining-implants-how-to-generate-c2-framework-implants-at-scale.html (besucht am 30. 09. 2024).

[52] Avigayil Mechtinger, Ryan Robinson und Joakim Kennedy. *Vermilion Strike: Linux and Windows Re-implementation of Cobalt Strike*. 2021. URL: https://intezer.com/blog/malware-analysis/vermilionstrike-reimplementation-cobaltstrike/ (besucht am 18. 08. 2024).

[53] *Microsoft-365-Defender-Hunting-Queries/Command and Control/C2-NamedPipe.md at master*. 2021. URL: https://github.com/microsoft/Microsoft-365-Defender-Hunting-Queries/blob/master/Command%20and%20Control/C2-NamedPipe.md (besucht am 25. 08. 2024).

[54] Aleksandar Milenkosk und Jim Walter. *Crimeware Trens | Ransomware Developers Turn to Intermittent Encryption to Evade Detection - SentinelOne*. 2022. URL: https://www.sentinelone.com/labs/crimeware-trends-ransomware-developers-turn-to-intermittent-encryption-to-evade-detection/ (besucht am 18. 08. 2024).

[55] Peter Mockapetris. *DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION*. 1987. URL: https://www.ietf.org/rfc/rfc1035.txt (besucht am 15. 08. 2024).

[56] Lindsey O'Donnell-Welch. *Ransomware Payments Hit 1.1B Record in 2023 | Decipher*. 2024. URL: https://duo.com/decipher/ransomware-payments-hit-usd1-1b-record-in-2023 (besucht am 30. 09. 2024).

[57] Bhavin Patel. *DNS Query Length With High Standard Deviation - Splunk Security Content*. 2024. URL: https://research.splunk.com/network/1a67f15a-f4ff-4170-84e9-08cf6f75d6f5/ (besucht am 25. 08. 2024).

[58] Norbert Pohlmann. *Cyber-Sicherheit*. 2. Auflage. Springer Nature, 2022. ISBN: 978-3-658-36243-0.

[59] Umesh Hodeghatta Rao und Umesha Nayak. *The InfoSec Handbook: An Introduction to Information Security*. 1. Auflage. Apress, 2014. ISBN: 978-1-4302-6383-8.

[60] *Referenz: Azure Policy-Gastkonfigurationsbaseline für Windows - Azure Policy | Microsoft Learn*. 2024. URL: https://learn.microsoft.com/de-de/azure/governance/policy/samples/guest-configuration-baseline-windows#security-options---accounts (besucht am 22. 08. 2024).

[61] E. Rescorla. *RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3*. 2018. URL: https://datatracker.ietf.org/doc/html/rfc8446 (besucht am 15. 08. 2024).

[62] Durgesh Sangvikar u. a. *Attackers Exploiting Public Cobalt Strike Profiles*. 2024. URL: https://unit42.paloaltonetworks.com/attackers-exploit-public-cobalt-strike-profiles/ (besucht am 22. 08. 2024).

[63] *Set-MpPreference (Defender) | Microsoft Learn*. 2024. URL: https://learn.microsoft.com/en-us/powershell/module/defender/set-mppreference?view=windowsserver2019-ps (besucht am 22. 08. 2024).

[64] Niraj Shivtarkar und Shatak Jain. *Havoc Across the Cyberspace | Blog | Zscaler*. 2023. URL: https://www.zscaler.com/blogs/security-research/havoc-across-cyberspace (besucht am 25. 08. 2024).

[65] Bundesamt für Sicherheit in der Informationstechnik. *BSI - Bundesamt für Sicherheit in der Informationstechnik - BSI-Leitfaden zur Einführung von Intrusion-Detection-Systemen*. 2020. URL: https://www.bsi.bund.de/dok/6624202 (besucht am 16. 08. 2024).

[66] Bundesamt für Sicherheit in der Informationstechnik. *Positionspapier Zero Trust 2023*. 2023. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeLeitlinien/Zero-Trust/Zero-Trust_04072023.pdf?__blob=publicationFile&v=4 (besucht am 06. 10. 2024).

[67] Michael Sikorski und Andrew Honig. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. 1. Auflage. No Starch Press, Inc., 2012. ISBN: 978-1-59327-290-6.

[68] *sliver/util/encoders/english_test.go at 3c151ed172076b654a5eff8b84686addb2ba3e73 - BishopFox/sliver - GitHub*. 2023. URL: https://github.com/BishopFox/sliver/blob/3c151ed172076b654a5eff8b84686addb2ba3e73/util/encoders/english_test.go#L44 (besucht am 01. 09. 2024).

[69] *Snort - Network Intrusion Detection & Prevention System*. 2024. URL: https://www.snort.org.

[70] *Snort 3 User Manual*. 2024. URL: https://github.com/snort3/snort3/releases/download/3.3.2.0/snort_user.pdf (besucht am 15. 08. 2024).

[71] *Suricata User Guide - Suricata 7.0.6 documentation*. 2024. URL: https://docs.suricata.io/_/downloads/en/suricata-7.0.6/pdf/ (besucht am 15. 08. 2024).

[72] Janos Szurdi, Rebekah Houser und Daiping Liu. *Domain Shadowing: A Stealthy Use of DNS Compromise for Cybercrime*. 2022. URL: https://unit42.paloaltonetworks.com/domain-shadowing/ (besucht am 01. 04. 2024).

[73] Cisco Talos. *Snort Rules and IDS Software Download*. 2024. URL: https://www.snort.org/downloads/community/snort3-community-rules.tar.gz (besucht am 19. 08. 2024).

[74] Inc. Team Cymru. *Team Cymru Threat Research auf X*. 2023. URL: https://x.com/teamcymru_S2/status/1626597384284438532 (besucht am 19. 08. 2024).

[75] *The Zeek Network Security Monitor*. 2024. URL: https://zeek.org.

[76] Emerging Threats. *emerging-all.rules*. 2024. URL: https://rules.emergingthreats.net/open/suricata-7.0.3/emerging-all.rules (besucht am 19. 08. 2024).

[77] *Timeline - Suricata*. 2024. URL: https://suricata.io/timeline/.

[78] Rico Valdez. *Detect Long DNS TXT Record Response - Splunk Security Content*. 2024. URL: https://research.splunk.com/deprecated/05437c07-62f5-452e-afdc-04dd44815bb9/ (besucht am 25. 08. 2024).

[79] Gaebler Ventures. *Domain Length Research - How Important Is URL Length? - Choosing a Domain Name*. 2009. URL: https://www.gaebler.com/Domain-Length-Research.htm (besucht am 25. 08. 2024).

[80] Dark Vortex. *Brute Ratel C4 | Badger doesn't care. It takes what it wants!* 2024. URL: https://bruteratel.com (besucht am 18. 08. 2024).

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

A	Address. 11, 64
API	Application Programming Interfaces. 25, 41, 72, *Glossar:* Application Programming Interfaces
ARP	Address Resolution Protocol. 10, 15
AV	Antivirus. 6, 18
AWS	Amazon Web Services. 66
C2	Command and Control. 6–9, 19, 24–33, 47–52, 54–56, 58–63, 66, 67, 69–75, 78–83, 85, 91, 92, 94, 96, 104, *Glossar:* Command and Control
CLI	Command Line Interface. 35, 41, 44
CNAME	Canonical Name. 11, 63–65
CTI	Cyber Threat Intelligence. 38, 44–46, 73, 75
DDoS	Distributed Denial of Service. 5, 24
DGA	Domain Generation Algorithm. 26
DNS	Domain Name System. 11, 12, 26, 47, 48, 50, 52, 54, 56–59, 63–65, 68–70, 75–79, 85, 86, 91, 104, 106
DPI	Deep Packet Inspection. 46
EDR	Endpoint Detection and Response. 6, 18
EVE JSON	Extensible Event Format JSON. 42, 44, 46
FCS	Frame Check Sequence. 11
FQDN	Fully Qualified Domain Name. 68
FTP	File Transfer Protocol. 41
HIDS	Host-based intrusion detection system. 18, 34, 39, 49
HTTP	Hypertext Transfer Protocol. 13, 27, 41, 47, 50, 51, 58, 60–62, 65, 66, 71–73, 78, 79, 92, 106
HTTPS	Hypertext Transfer Protocol Secure. 13, 14, 27, 28, 43, 50, 51, 62, 67, 78
ICMP	Internet Control Message Protocol. 27
IDS	Intrusion Detection System. 6–9, 14, 16, 18, 34, 41, 80, 111
IoC	Indicator of Compromise. 45, 112
IP	Internet Protocol. 10–12, 15, 24, 26, 52, 53, 57, 58, 63, 73, 75
IPC	Interprozesskommunikation. 67

IPS	Intrusion Prevention System. 6–9, 14, 16, 41, 80, 111
JSON	JavaScript Object Notation. 42, 44, 46
KI	Künstliche Intelligenz. 82, 83
LAN	Local Area Network. 10, 21, 24, 52–57
LOLBins	Living Off The Land Binaries. 23
MAC	Media Access Control. 10
MISP	Malware Information Sharing Platform & Threat Sharing. 44
MitM	Man-in-the-Middle. 15, 21
mTLS	Mutual TLS. 28, 50, 62, 75, 78, 80, 82
MX	Mail Exchange. 11, 63, 64
NAT	Network Address Translation. 15, 57
NGFW	Next-Generation Firewall. 6, 14, 39, 82
NIDS	Network Intrusion Detection System. 14, 16–18, 26, 27, 34–40, 49, 52–54, 73, 79, 81, 82, 111, 112, *Glossar:* Network Intrusion Detection System
NIPS	Network Intrusion Prevention System. 16, 17, 35, 39, 111
NMS	Network Monitoring Systems. 36, 37, 39, 44, 81, 82
NS	Name Server. 12, 26, 53, 56, 57, 59
OST	Offensive Security Tools. 32, 51, 73, 80, 83, 104
P2P	Peer-to-Peer. 25
pcap	Packet Capture. 37, 83
PCRE	Perl Compatible Regular Expressions. 47, 71
PoC	Proof of Concept. 7
PTR	Pointer. 12
RaaS	Ransomware as a Service. 20
RAT	Remote Access Trojan. 24, 31
RDP	Remote Desktop Protocol. 30
RPC	Remote Procedure Call. 74
RR	Resource Record. 11, 26, 63–65
SIEM	Security Information and Event Management. 14, 36, 37, 42, 46, 112, *Glossar:* Security Information and Event Management
SMB	Server Message Block. 27, 28, 41, 50, 51, 54, 55, 58, 61, 62, 67, 74, 78, 94, 104
SMTP	Simple Mail Transfer Protocol. 41
SNI	Server Name Indication. 48
TCP	Transport Control Protocol. 11, 13, 14, 41, 67

TLD	Top-Level-Domain. 12
TLS	Transport Layer Security. 13, 14, 28, 43, 47, 48, 50, 67, 73
TTL	Time-to-Live. 12
TTP	tactics, techniques, and procedures. 82, 83
UDP	User Datagram Protocol. 11, 68, 75, 77
URI	Uniform Resource Identifier. 66, 71, 73
URL	Uniform Resource Locator. 19, 31
VLAN	Virtual Local Area Network. 16
VM	Virtuelle Maschine. 22, 52–57
VPN	Virtual Private Network. 21, 28, 29, 50, 62, 68, 69, 78, 80, 82, 96

Glossar

Application Programming Interfaces	Schnittstelle, welche auf Funktionen zugreifen kann. Dies ermöglicht anderen Anwendungen oder Systemen, Daten über die API abzufragen oder auszutauschen. 25, 107
Command and Control	Kommunikation zwischen einem mit einer Malware infizierten System und steuernden Akteur, in der Regel mit schadhaften Absichten. 6, 107
Defense in Depth	Defense in Depth beschreibt eine Absicherungsstrategie von Netzwerken, welche über mehrere Ebenen hinweg aufgebaut wird. Hierbei greifen meist mehrschichtige Sicherheitseinrichtungen ineinander, um Zugangspunkte vor ungewollten Zugriffen zu schützen. Die Idee dahinter ist, dass wenn eine Schutzmaßnahme umgangen werden konnte eine der anderen Schutzmaßnahmen eingreift[66, S. 7 f., 29]. 6, 49

NetFlow	NetFlow ist ein Protokoll, welches Informationen über Ein-/Ausgehenden Datenverkehr einer Netzwerkschnittstelle protokolliert. Hierbei werden bestimmte Metadaten wie Quelle, Ziel, Quell-/Zielport, Paketgröße und ähnliche Informationen der Pakete gesammelt und korreliert. So entsteht z. B. ein Bild darüber, welche Systeme zu welchen Uhrzeiten miteinander kommunizieren und wie viele Daten ausgetauscht werden. Solche Informationen kann aus der Sicht eines Netzwerkadministrators relevant sein, da so festgestellt werden kann, ob das Netzwerk den real herrschenden Bedingungen noch entspricht oder nicht. Aus der Sicht der IT-Sicherheit können solche Informationen genutzt werden, um Anomalien im Netzwerk festzustellen, da z. B. Historische Daten darüber vorliegen, wie hoch das Datenvolumen ist oder welche Verbindungen zwischen welchen Systemen normal sind[13]. 14, 82
Network Intrusion Detection System	Konkretere funktionale Beschreibung des Begriffs IDS oder IPS. NIDS detektieren mögliche Angriffe oder andere Anomalien im Bereich des Netzwerkverkehrs. Analog zu IDS und IPS existiert auch der Begriff NIPS, im Rahmen dieser Arbeit werden NIDS und NIPS analog und synonym verwendet. 14, 108

Security Information and Event Management	Bei einem SIEM-System handelt es sich um eine speziellere Ausprägung eines zentralen Logmanagement-Systems. Analog werden auch hier Logdaten unterschiedlicher Systeme wie Netzwerkkomponenten sowie Server-, Clientsysteme oder Protokolle anderer Dienste zusammengeführt. Durch die Korrelation vieler Ereignisse unterschiedlicher Quellen können Angriffe, Anomalien oder Sicherheitsverletzungen frühzeitig detektiert werden. 14, 108
Threathunting	Unter Threathunting versteht man die Proaktive Suche nach Spuren von Cyberangriffen, ohne, dass es einen Hinweis auf mögliche Sicherheitsverletzungen gibt. Darunter fällt auch die Suche nach konkreten IoC, welche aus z. B. Alarmen einer NIDS hervorgehen. 37, 38, 81