HAW HAMBURG

Bachelorarbeit

Stephan Jänecke

Erweiterung einer Threat Intelligence Plattform um Abfrage und Auswertung von Daten von Passive DNS Sensoren am Beispiel von YETI

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Stephan Jänecke

Erweiterung einer Threat Intelligence Plattform um Abfrage und Auswertung von Daten von Passive DNS Sensoren am Beispiel von YETI

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Dr. Klaus-Peter Kossakowski
Zweitgutachter: Prof. Dr.-Ing. Andreas Meisel

Eingereicht am: 10. Juli 2019

Stephan Jänecke

Thema der Arbeit

Erweiterung einer Threat Intelligence Plattform um Abfrage und Auswertung von Daten von Passive DNS Sensoren am Beispiel von YETI

Stichworte

passive dns, yeti, security

Kurzzusammenfassung

In dieser Bachelorarbeit wird die Threat Intelligence Plattform YETI erweitert, um Hosts aus aufgezeichneten Netzwerkverkehrsdaten in Kategorien einzuordnen. Dazu werden Passive DNS Daten und Reputationswerte eingebunden. Der Nutzer soll in einer Visualisierung eines Computernetzwerkes sehen können, ob Hosts aus diesem Netzwerk Aktionen ausgeführt haben, die auf eine mögliche Kompromittierung durch Schadsoftware schließen lassen.

Stephan Jänecke

Title of Thesis

Extension of a threat intelligence plattform to allow querying and analysing passive dns sensors using YETI

Keywords

passive dns, yeti, security

Abstract

In this thesis the threat intelligence platform YETI is extended to allow categorising hosts in a computer network. Passive DNS data, reputation values and network traffic data are joined to help the user find hosts that might possibly be compromised as they perform certain harmful actions.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einleitung

Das *Domain Name System* ist nicht nur Basis für Internet-Dienste, wie Web oder E-Mail, sondern bietet auch Erleichterung für Kriminelle. Hinter Domainnamen können sich Server verstecken, die Phishing Seiten bereitstellen, Malware verteilen oder Botnetze steuern. Nebenbei erleichtert die Verwendung des DNS auch die Lastverteilung und Sicherstellung der Verfügbarkeit. Gerade Server, die betrügerische Dienste anbieten, laufen Gefahr entdeckt und abgeschaltet zu werden. Es liegt im Interesse des Kriminellen, dass daher viele Server, wenn auch nur für kurze Zeit, erreichbar sind.

Häufig lässt sich ein Malwarebefall nicht so offensichtlich erkennen, wie bei Ransomware. Deren Ziel ist die Erpressung der Opfer, indem Daten verschlüsselt oder Systeme unbrauchbar gemacht werden. Bei Malware soll der Befall hingegen möglichst lange unentdeckt bleiben, um den Einbrechern weitere Aktionen zu ermöglichen.

Die Entdeckung solcher Malware kann sich auf die Analyse des Netzwerkverkehrs zwischen infizierten Systemen und der Außenwelt stützen. Welche Domainnamen werden beim DNS abgefragt? Werden darüber hinaus Verbindungen zu Hosts, die zu der Domain gehören, aufgebaut? Werden aus einem Mitschnitt des Netzwerkverkehrs die DNS-Anfragen extrahiert, können die enthaltenen Domainnamen mit Listen bekannter Malware Domains abgeglichen werden oder direkt bei einem Reputation Service abgefragt werden.

Passive DNS Dienstleister bieten die Möglichkeit DNS-Daten zu IP-Adressen oder Domains abfragen zu können, die nicht mehr im DNS eingetragen sind. So lassen sich auch im Nachhinein bekannte bösartige Domains und Adressen zuordnen. Die Analyse eines bereits erfolgten Angriffs wird so vereinfacht.

Dies kann durch eigene Passive DNS Sensoren ergänzt werden. So kann schneller oder automatisiert festgestellt werden, ob Systeme in eigenen Netzwerken betroffen sind oder waren.

1.1 Ziel der Arbeit

Die Threat-Intelligence-Plattform YETI soll um eine Möglichkeit zur Kategorisierung von Hosts in einem Netzwerk erweitert werden. Dazu wird der importierte Netzwerkverkehr auf DNS-Anfragen untersucht und Domains, die aufgelöst wurden, mit Daten von Passive DNS Dienstleistern und Reputation Services verbunden. Anhand der Auswertung der Daten werden Hosts kategorisiert. Soll bei Verdacht auf Malwarebefall festgestellt werden, ob zudem Informationen abflossen sind, wird untersucht, ob Kommunikation zwischen verdächtiger Domain und Host stattfand.

1.2 Zielgruppe der Arbeit

Leser dieser Arbeit sollten über grundlegende Kenntnisse im Gebiet der Rechnernetze verfügen und ein Interesse an IT-Sicherheit haben. Dem Nutzer der Erweiterung soll die Beurteilung erleichtert werden, ob Hosts in einem Netzwerk zu einem bestimmten Zeitpunkt gefährdet waren. Als Grundlage dienen dazu eine Aufzeichnung des Netzwerkverkehrs und die in YETI vorliegenden Daten.

1.3 Struktur

Die vorliegende Arbeit ist in sieben Kapitel und Anhang unterteilt. Kapitel 1 führt in die Arbeit ein und erläutert kurz die zugrunde liegende Problematik und deren Lösung. Kapitel 2 stellt die Konzepte vor, auf der die Lösung basieren soll. Kapitel 3 definiert die Anforderungen an die Lösung, die aus dem eigentlichen Problem und Randbedingungen erarbeitet werden. Kapitel 4 und Kapitel 5 stellen ein Design und dessen Umsetzung vor, dass den Anforderungen aus Kapitel 3 genügen soll. Kapitel 6 stellt dazu Tests vor, um die Implementierung auf Erfüllung der Anforderungen zu prüfen. Schließlich zieht Kapitel 7 ein Fazit und stellt Ansätze für Weiterentwicklungen vor. Anhang A sowie Anhang B sollen bei der Einrichtung von YETI und der entwickelten Erweiterung helfen.

1.4 Abgrenzung

Um den Gewinn durch die Einbindung von Passive DNS Daten darzustellen, soll Passive DNS kurz eingeführt werden. Die Einbindung in die Threat-Intelligence-Plattform YETI beschränkt sich auf die für Plugins vorgeschriebenen Schnittstellen, eine Modifikation der Quellen von YETI soll vermieden werden. So soll eine einfachere Inbetriebnahme der Erweiterung gewährleistet werden und eine Einbindung in das Upstream-Projekt ermöglicht werden. Um Abhängigkeiten zu vermeiden, werden auch Aktionen vom Nutzer erwartet. Darauf wird an geeigneter Stelle oder in Anhang B hingewiesen.

1.5 Definitionen

Um Mehrdeutigkeiten zu vermeiden, sollen wichtige Begriffe im Kontext dieser Arbeit definiert werden.

DNS Das *Domain Name System* ist ein hierarchisches System, dass im Internet Computern, Diensten und Ressourcen Namen zuordnet[6]. Es ist eine Kernkomponente des Internet. Im Folgenden wird besonderes Augenmerk auf die Domainnamen gelegt, um deren missbräuchliche Nutzung bei Angriffen zu zeigen.

Passive DNS Als Passive DNS wird die Aufzeichnung von DNS Anfragen und Antworten bezeichnet[10].

Cyber Threat Intelligence Als Threat Intelligence wird die Kombination von Wissen über mögliche Gefährdungen und Wissen über die Struktur, die Betriebsabläufe und die Nutzung eines (Computer-) Netzwerks bezeichnet[2]. Sie dient der Abwehr sowie der Prävention von Angriffen.

Threat-Intelligence-Plattform Die Threat-Intelligence-Plattform dient der Sammlung, Aufbereitung und Verteilung von Threat Intelligence Daten[9].

2 Grundlagen

In diesem Kapitel sollen Begriffe und Konzepte vorgestellt werden, die den folgenden Kapiteln als Basis dienen.

2.1 Passive DNS

Passive DNS bezeichnet das Speichern von DNS-Paketen in einem Netzwerk und deren Auswertung[10]. Dazu filtert ein Sensor die DNS-Pakete aus dem Netzwerk. So kann auch DNS-Verkehr sichtbar gemacht werden, der nicht an eigene DNS-Server oder Resolver gerichtet ist. Passive DNS ermöglicht zudem die Überwachung des DNS-Verkehrs ohne einen eigenen DNS-Server betreiben zu müssen.

Passive DNS Daten können als Quelle dienen um die Reputation von Domains zu bewerten[1]. So hinterlässt Phishing charakteristische Spuren in DNS Einträgen[4]. Gefährliche Domains können lokal früher erkannt werden als es die Verwendung von Blacklists oder Reputation Services zulassen würde.

In Malware wird der Command and Control Server oft in Form eines Namens fest vorgegeben. Die Überwachung der DNS Anfragen der eigenen Hosts ermöglicht so festzustellen, ob Malwarebefall auf den Hosts vorliegt. Werden die Daten über einen längeren Zeitraum gespeichert, können laufende Malware Kampagnen oder Angriffe identifiziert werden.

Um Passive DNS einzusetzen müssen, nicht unbedingt eigene Sensoren installiert werden. Dienstleister stellen ihre eigene Datenbasis zur Verfügung und machen sie über Webfrontends oder APIs zugänglich.

2.2 Reputation

Reputation ist die Abschätzung des Risikos, dass von Domains, IP-Adressen und URLs ausgeht und dient als Threat Indicator oder Indicator of Compromise (IoC)[5, S. 18]. Ein hohes Risiko geht meist von Systemen aus, die in der Verbreitung von Malware oder SPAM oder in Phishing-Kampagnen involviert sind.

2.2.1 Erzeugung

Reputationswerte für Domains oder IP-Adressen entstehen durch die Auswertung von Honeypots[5, S. 20]. URLs und E-Mails werden extrahiert und die Web- oder Mailserver, die die betroffenen Services zur Verfügung stellen, auf Malwarebefall überprüft. In die Bewertung von Mailservern kann auch einfließen, ob die empfangenen Mails Merkmale von Phishing oder SPAM aufweisen.

2.2.2 ThreatCrowd

ThreatCrowd sieht sich als System zum Finden und Erforschen von Artefakten, die mit Cyber Threats zusammenhängen[1]. Der Dienst bietet den Nutzern die Möglichkeit abzustimmen, ob ein Artefakt schädlich ist oder nicht. Die Abstimmungsergebnisse können über das API[8] bezogen werden. Über selbiges kann auch die Abstimmung erfolgen. Nach der API-Dokumentation[8] haben die Werte, die `Votes` annehmen kann, die folgende Bedeutung:

-1 Eine Mehrheit stuft das Objekt als bösartig ein.

0 Es konnte keine Mehrheit ausgemacht werden.

1 Eine Mehrheit stuft das Objekt als nicht bösartig ein.

Somit entsteht der Reputationswert bei ThreatCrowd nicht durch die wissenschaftliche Analyse durch Sicherheitsforscher, sondern wird durch die Nutzer abgestimmt. Es ist anzunehmen, dass die Zeit bis zur Entdeckung eines bösartigen Artefakts sinkt. Die anonyme Stimmabgabe lässt Platz für Missbrauch und sollte bei der Abwägung der Zuverlässigkeit der Reputationswerte berücksichtigt werden.

[1] „ThreatCrowd is a system for finding and researching artefacts relating to cyber threats.” https://threatcrowd.blogspot.com/2015/03/tutorial.html

2.3 YETI

Die Entwickler von YETI sehen ihre Plattform als zentralen Punkt um Observables, IoCs, Tactics, Techniques and Procedures (TTP) und Wissen über Threats zu sammeln und zu verknüpfen[2].

2.3.1 Observables

Ein Observable hat einen Typ und kann ein entsprechendes Datum aufnehmen. Im *Context* des Observables steht ein beliebiger Text. Des Weiteren können Observables mit Tags versehen werden. Zusammenhängende Datensätze können in Observables gespeichert werden, indem die einzelnen Daten in Observables des entsprechenden Typs überführt und miteinander durch Links verknüpft werden.

2.3.2 Erweiterung

Erweiterungen für YETI werden in Module verpackt. Es wird unterschieden zwischen:

analytics Automatische oder manuelle Verarbeitung von Observables.

exports Dienen zum Exportieren der Daten [3].

feeds Fragen automatisch regelmäßig Datenquellen ab. Deren Einträge werden entweder durch den Feed selbst oder weitere Inline Analytics Module (siehe unten) in Observables überführt. Diese stehen dann zur Weiterverarbeitung zur Verfügung.

import methods Dienen zum Extrahieren von Daten aus Dokumenten. Kann manuell durch Userinteraktion im Webfrontend erfolgen.

Die Analytics Module lassen sich noch in Inline Analytics, Scheduled Analytics und Oneshot Analytics unterteilen. *Inline Analytics* werden automatisch ausgeführt, sobald neue Observables hinzugefügt werden. *Oneshot Analytics* werden manuell durch den User auf ausgewählte Observables angewandt. Die *Scheduled Analytics* werden zu bestimmten Zeitpunkten ausgeführt.

[2] Siehe ReadMe: https://github.com/yeti-platform/yeti/blob/master/README.md
[3] Zum Zeitpunkt des Verfassens nicht implementiert.

2.3.3 Investigation

Die Investigation ist der Kontext, in dem Observables vom Nutzer gruppiert und bearbeitet werden können. Die Verbindungen der Observables werden in einem Graphen visualisiert. Wählt die Nutzerin ein oder mehrere Observables aus, kann sie die Oneshot Analytics Module auf die Auswahl anwenden.

Verbindet der Nutzer innerhalb einer Investigation Observables mit Links, werden diese nicht in die Datenbank gespeichert. Sie sind nur innerhalb der Investigation nutzbar, in der sie erstellt wurden. Bestehende Observables können von Analytics Modulen hinzugefügt werden. Die Nutzerin kann Observables, die bei der Erstellung der Investigation generiert wurden, dem Graph hinzufügen und wieder daraus entfernen.

2.3.4 PCAP Investigation

Die Erweiterung für den Import von PCAP Aufzeichnungen als Investigations von Tim Stammerjohann soll genutzt werden, um schneller größere Netzwerke zu visualisieren. Sie bildet zudem TCP- und UDP-Verbindungen zwischen Hosts ab und kann IP-Adressen aus DNS-replies markieren[7, S. 15f]. Im Folgenden ist diese Erweiterung gemeint, wenn auf PCAP Investigations verwiesen wird.

Anhand des Quellcodes kann nachvollzogen werden, dass die Erweiterung Verbindungen und DNS Auflösungen auf Links abbildet. Die Dateinamen in den Beschreibungen der Listings verweisen auf die jeweiligen Codedateien der PCAP Investigations. Nach Zeile 14 in Listing 2.1 werden aufgelöste IP-Adressen mit ihren DNS-Namen durch Links mit der Bezeichnung `DNS response includes` verbunden. Für TCP- und UDP-Verbindungen werden die Links mit `Connection to` bezeichnet, wie in Zeile 21 in Listing 2.2 zu sehen.

2.3.5 ThreatCrowd Analytics Module

YETI bietet die Möglichkeit, mit dem OneShotAnalytics Module `ThreatCrowd`, die Threat Suchmaschine ThreatCrowd abzufragen. Anfragen werden über das öffentliche API abgewickelt. Die Antwort auf eine API-Abfrage legt das Modul als JSON-String im Context des verarbeiteten Observables ab. Da Hostnamen und IP-Adressen bereits vom

Listing 2.1: Ausschnitt aus `extract_observables_from_dns_response()` aus `extraction.py`

```
h = Hostname.get_or_create(value=dns_query.name)
h.add_source('PCAP')
ip_observables = []
for answer in dns.an:  # multiple possible ip addresses
    if answer.type == dpkt.dns.DNS_A:
        ip = Ip.get_or_create(value=ipv4_to_str(answer.rdata))
    elif answer.type == dpkt.dns.DNS_AAAA:
        ip = Ip.get_or_create(value=ipv6_to_str(answer.rdata))
    else:
        continue
    ip.add_source('PCAP')
    ip.add_source('DNS Response')
    ip_observables.append(ip)
h.active_link_to(ip_observables, description='DNS response includes',
    source='PCAP')
```

Listing 2.2: Ausschnitt aus `_create_investigation()` aus `investigation.py`

```
obs_src_ip = Ip.get_or_create(value=src_ip)
obs_src_ip.add_source('PCAP')
src_ip_stats = self._statistics[src_ip]

tcp_flags_set = src_ip_stats.get('tcp_flags_sent')
if tcp_flags_set is not None and len(tcp_flags_set) == 1 and 'SYN' in
    tcp_flags_set:
    obs_src_ip.tag('syn_without_ack')

dst_ips = src_ip_stats.pop('dst_ips')
if len(dst_ips) > 1:
    observables_to_add.append(obs_src_ip)

for dst_ip in dst_ips:
    try:
        ip_count[dst_ip] += 1
    except KeyError:
        ip_count[dst_ip] = 1

    obs_dst_ip = Ip.get_or_create(value=dst_ip)
    obs_dst_ip.add_source('PCAP')
    obs_src_ip.active_link_to(obs_dst_ip, 'Connection to', 'PCAP')
```

Listing 2.3: Rückgabe für die Anfrage `microsoftwindowsupdate.net`


```
{"response_code":"1","resolutions":[{"last_resolved":"2017-11-07","
ip_address":"-"},{"last_resolved":"2016-12-14","ip_address":"
64.4.6.233"},{"last_resolved":"2016-09-25","ip_address":"65.55.39.12"
},{"last_resolved":"2013-08-28","ip_address":"87.98.167.141"}],"
hashes":[],"emails":["domains@microsoft.com","msnhst@microsoft.com"],
"subdomains":["srv01.microsoftwindowsupdate.net","srv05.
microsoftwindowsupdate.net"],"references":["httpwww.cylance.
comassetsCleaverCylance_Operation_Cleaver_Report.pdf"],"votes":-1,"
permalink":"https:\/\/www.threatcrowd.org\/domain.php?domain=
microsoftwindowsupdate.net"}
```


Modul in Observables überführt werden und mit dem Ausgangsobservable verbunden werden, müssen noch die Votes extrahiert werden.

Beispielsweise kann der Antwort für die Abfrage der Domain `microsoftwindowsupdate.net` aus Listing 2.3 entnommen werden, dass diese als bösartig eingestuft wurde[4]. Der Wert von `votes` beträgt `-1`.

[4]Die Werte und deren Bedeutung wurden in Unterabschnitt 2.2.2 erläutert.

3 Anforderungen

3.1 Nichtfunktionelle Anforderungen

3.1.1 Systemanforderungen

Die Erweiterung wurde in einer virtuellen Maschine unter Debian 9 Stretch getestet. Für die PCAP Investigations musste die Python Installation in der Version 2.7.13 um das Paket `dpkt` erweitert werden. Die Installation ist in Anhang A dokumentiert.

Der Arbeitsspeicher ist mit 2 GB knapp bemessen und sollte größer ausfallen, um Abstürze von `redis` bei größeren Datenmengen zu vermeiden.

3.2 Funktionale Anforderungen

3.2.1 Benutzerinteraktion

Für bereits vorhandene Observables vom Typ Hostname oder IP-Adresse soll der Nutzer die Kategorisierung des Observables in der Investigation anstoßen können.

3.2.2 Abfrage von Passive DNS Daten

Der Hostname oder die IP-Adresse aus dem Observable werden genutzt, um über das API des Passive DNS Dienstleisters, hier CirclDNS, die zugehörigen Passive DNS Daten abzufragen. Sie werden dem Observable hinzugefügt. Dazu wird das vorhandene Oneshot Analytics Module `circl_pdns` genutzt.

3.2.3 Abfrage des Reputation Service

Die Rückgabe des in YETI integrierten ThreatCrowd Analytics Module soll berücksichtigt werden. Dazu soll der Wert von `votes` berücksichtigt werden.

3.2.4 Kategorisierung

Zur Kategorisierung werden mit dem Observable verbundene Hostnamen und IP-Adressen betrachtet. Bewertungen durch einen Reputation Service werden berücksichtigt. Es wird zwischen internen und externen Hosts unterschieden. Erstere können durch Hostnamen oder IP-Adressen repräsentiert werden und müssen explizit markiert werden. Bei externen Hosts wird unterschieden, ob sie als `evil` markiert wurden oder nicht.

good Das Observable ist nicht als `evil` markiert[1].

evil Das Observable wird als verdächtig eingestuft. Dies kann an der Bewertung des Reputation Service liegen oder an Tags der betrachteten Nachbarn.

Über die Kommunikationbeziehungen zwischen internen und externen Hosts können Hosts aus dem internen Bereich eingestuft werden.

normal Der Host ist weder mit `infected` oder `data-leak` markiert[2].

infected Der Host hat versucht einen Hostname oder eine IP-Adresse aufzulösen, der beziehungsweise die als `evil` markiert wurde.

data-leak Der Host ist bereits als `infected` markiert, hat allerdings auch Daten mit einem Host ausgetauscht, der als `evil` markiert wurde.

[1] Eine `good` Markierung wird nicht explizit gesetzt.
[2] Eine `normal` Markierung wird nicht explizit gesetzt.

4 Konzept

Die Kategorisierung der Hosts und Domains unterscheidet zwischen internen, aus einem eigenen Netzwerk, und externen, aus dem Internet oder Netzwerken, die nicht unter eigener Kontrolle stehen. Im Folgenden werden mit *Hosts* auch Domain Namen und IP-Adressen bezeichnet. Wie aus Abbildung 4.1 ersichtlich, sollen die Gruppen anhand des `internal` Tags unterschieden werden. Entsprechend wird ein Observable wie in Abschnitt 4.1 oder Abschnitt 4.2 weiterverarbeitet.

4.1 Bewertung externer Hosts

Als Basis für die Bewertung externer Hosts dienen die Reputation des Observables und die Nachbarn des Observables.

Nach Abbildung 4.2 werden zu Beginn die Daten aus dem Context des Reputationservice ausgewertet. Bewertet dieser den Hosts negativ, wird er als `evil` markiert. Liegt keine Bewertung vor, werden die Tags der verknüpften IP-Adressen und Hostnames ausgewertet. Sind dort Tags vergeben, die nicht in einer Whitelist freigegeben wurden, wird der Host als `evil` markiert. Prinzipbedingt führt dies zu Fehlalarmen, wenn die Whitelist schlecht gepflegt wird.

4.2 Bewertung interner Hosts

Für die Bewertung der internen Hosts werden die Verbindungsdaten des ausgewählten Hosts betrachtet. Dazu werden die benachbarten Observables auf mit ihnen verbundene Hostnames überprüft, die Inhalt einer DNS Anfrage waren. Existieren hier mit `evil` markierte IP-Adressen oder Hostnames, wird das ursprüngliche Observable mit dem Tag

Abbildung 4.1: ThreatTagger

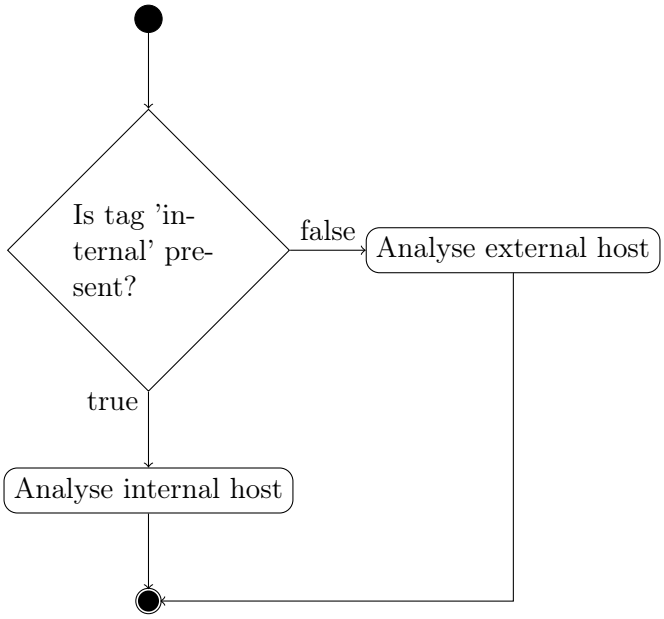


Abbildung 4.2: Analyse external hosts

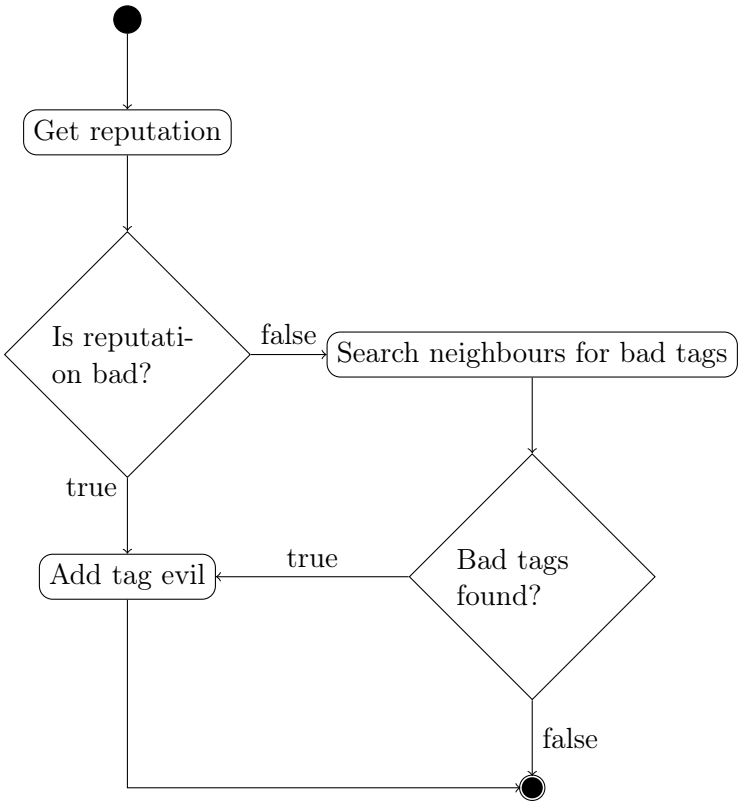
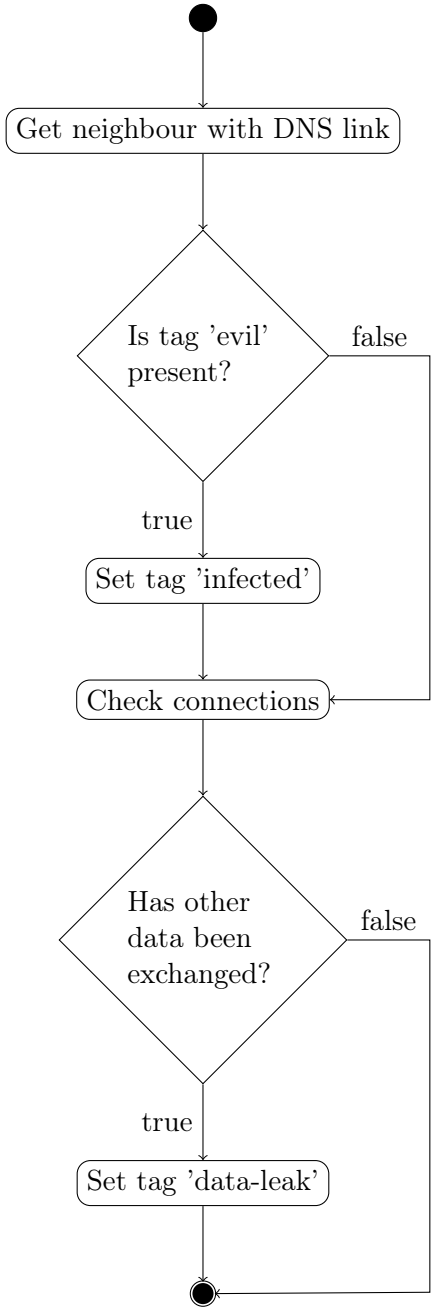


Abbildung 4.3: Analyse internal hosts



`infected` versehen. Der Ablauf ist in Abbildung 4.3 angedeutet. Hat sich der untersuchte Host darüber hinaus mit einer Adresse verbunden, die bereits als `evil` getaggt wurde, bekommt das Observable des Ursprungshosts das Tag `data-leak`.

5 Implementierung

Bei der Implementierung wurde Wert darauf gelegt, die Erweiterung in YETI einzubinden, ohne dazu die Plattform anpassen zu müssen. Der Code wird in ein einzelnes Modul `tagger.py` verpackt und `ThreatTagger` genannt. Auf Auszüge aus dem Code wird in den jeweiligen Abschnitten verwiesen. Zeilennummern werden nicht explizit angegeben, um die Lesbarkeit zu unterstützen. Die Zeilennummerierung in den Listings entspricht der in der Quelldatei.

5.1 Einbindung in YETI

Der Hauptzweck des Designs ist die Verarbeitung von Daten. Innerhalb von YETI bieten sich dazu die Analytics Module an, die auf Observables operieren.

Zur Entscheidung zwischen Oneshot Analytics und Inline Analytics muss betrachtet werden, ob die Kategorisierung automatisch erfolgen soll oder durch den User angestoßen werden muss.

Eine automatische Kategorisierung hat den Vorteil, dass auch bei neu importierten Observables direkt eine Beurteilung vorliegt und bei der weiteren Verarbeitung innerhalb einer Investigation zur Verfügung steht. Darin besteht auch ein Nachteil. Durch Feeds werden laufend viele Observables mit IP-Adressen und Hostnamen generiert. Der zusätzliche Aufwand durch die Kategorisierung würde somit unnötig Ressourcen für Observables verbrauchen, die in der Investigation möglicherweise nicht relevant werden. In der Implementierung wird daher ein Oneshot Analytics Module umgesetzt. Der User kann die Kategorisierung für Observables anstoßen, die für ihn auch relevant sind.

Des Weiteren gibt es keine Abhängigkeiten von anderen Modulen. Der User ist so selbst verantwortlich, die Datengrundlage für die Analyse bereitzustellen. Bei der Implementierung wurden PCAP-Investigations vorausgesetzt, die mit Daten von Circl PDNS und ThreatCrowd ergänzt wurden.

5.2 Einstufung von Hosts

Die Kategorisierung von Hosts unterscheidet zwischen zwei Gruppen: den internen und den externen Hosts. Unterschiede und Merkmale der beiden Gruppen wurden in der Einleitung zu Kapitel 4 und Abschnitt 4.1 erläutert. Entscheidend ist, ob das Tag `internal` gesetzt ist.

Bei internen Hosts wird angenommen, dass aus der PCAP Investigation erzeugte Verbindungen zu anderen Observables existieren. TCP- und UDP Verbindungen werden durch die Beschreibung `Connection to` markiert.

Externe Hosts können auch IP-Adressen oder Hostnames sein, zu denen keine TCP- oder UDP-Verbindung bestand.

5.2.1 Interne Hosts

Anhand von Listing 5.1 soll erläutert werden, wie interne Hosts verarbeitet werden. Liegt ein interner Host vor, wird im Graphen nach Informationen aus DNS requests und replies gesucht. Eine genaue Beschreibung erfolgt in Abschnitt 5.3. Wurden solche Informationen – in Form von Observables – gefunden, wird geprüft, ob diese mit dem Tag `evil` versehen wurden. Stellt sich heraus, dass solche Observables vorliegen, wird das Observable für den internen Host mit dem Tag `infected` markiert. Des Weiteren wird jetzt überprüft, ob zwischen den Observables Links mit der Beschreibung `Connection to` existieren. Ist auch das der Fall, wird dem internen Host noch das Tag `data-leak` zugewiesen.

5.2.2 Externe Hosts

Der Code für externe Hosts ist kompakter, wie in Listing 5.2 zu sehen. Ob externe Hosts als `evil` eingestuft werden, hängt davon ab, ob sie einen schlechten Reputationswert haben. Die Entscheidung wird in Abschnitt 5.5 erläutert.

5.3 Extrahierung von DNS Daten

Die Informationen aus DNS replies werden in einer PCAP Investigation mit Links mit der Beschreibung `DNS response includes` verbunden. IP-Adressen werden so mit

Listing 5.1: Ein interner Host wird getaggt

```
229         if observable.has_tag("internal"):
230
231             resolved_observables = set()
232             evil_observables = set()
233
234             # Check for DNS trafic resolving evil hostname or reverse
235             # resolving evil ip address (infected)
236             ips, hostnames = Tagger.get_resolved_ips_and_hostnames(
                    observable)
237             resolved_observables = resolved_observables.union(ips,
                    hostnames)
238             logging.debug("Resolved observables are: {}".format(
                    resolved_observables))
239
240             for item in resolved_observables:
241                 if item.has_tag("evil"):
242                     logging.debug("{} is evil".format(item))
243                     evil_observables.add(item)
244
245             if evil_observables:
246                 try:
247                     logging.debug("Observable is infected")
248                     observable.tag("infected")
249                 except ObservableValidationError as error:
250                     logging.error(
251                         "Error while adding tag: {}"
252                         .format(error.msg))
253
254                 # check for connections to resolved evil ip addresses
255                 # (data-leak)
256                 if Tagger.has_connected_to(observable,
                        evil_observables):
257                     try:
258                         logging.debug("Observable leaked data")
259                         observable.tag("data-leak")
260                     except ObservableValidationError as error:
261                         logging.error(
262                             "Error while adding tag: {}"
263                             .format(error.msg))
```

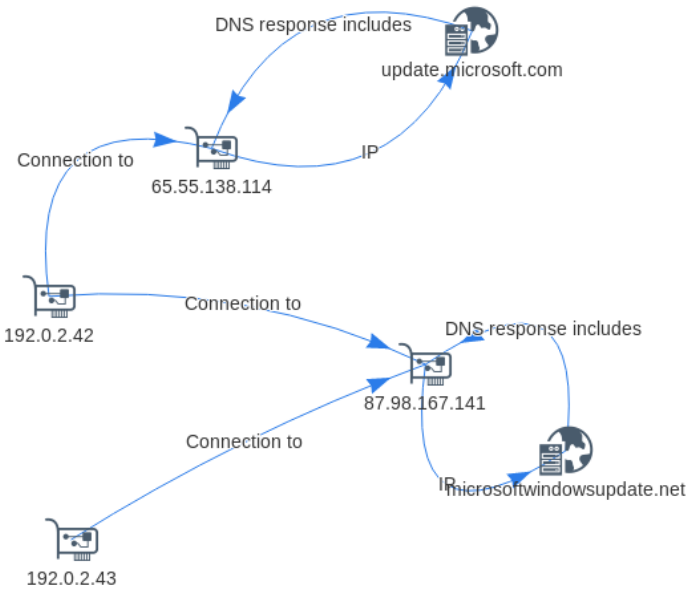
Listing 5.2: Ein externer Host wird getaggt

```
264         else:
265             if Tagger.has_bad_reputation(observable, results.settings
                ):
266                 try:
267                     logging.debug("Observable is evil")
268                     observable.tag("evil")
269                 except ObservableValidationError as error:
270                     logging.error(
271                         "Error while adding tag: {}"
272                         .format(error.msg))
```

Listing 5.3: Extrahierung von Hostnames und IP-Adressen

```
153         if neighbors:
154             for entry in neighbors[0]:
155                 links = entry[1].neighbors().values()
156                 logging.debug("{} has links: {}".format(
157                     entry,
158                     links))
159                 for link_entries in links:
160                     for connection in link_entries:
161                         description = connection[0].info()["
                            description"]
162                         if description == 'DNS response includes':
163                             logging.debug("{} reverse resolves to {}"
                                .format(
164                                 entry[1],
165                                 connection[1]))
166                             rr_hostnames.add(connection[1])
167                             r_ips.add(entry[1])
168 
169         # Get all ip addresses hostnames resolve to
170         if rr_hostnames:
171             for hostname in rr_hostnames:
172                 links = hostname.neighbors().values()
173                 logging.debug("{} has links: {}".format(
174                     hostname,
175                     links))
176                 for link_groups in links:
177                     for link in link_groups:
178                         description = link[0].info()["description"]
179                         if description == 'DNS response includes':
180                             logging.debug("{} resolves to {}".format(
181                                 hostname,
182                                 link[1]))
183                             r_ips.add(link[1])
```

Abbildung 5.1: Wer hat welche Domain aufgelöst?



Hostnamen verbunden, aus denen sie aufgelöst wurden. In die andere Richtung können auch Hostnamen in IP-Adressen aufgelöst werden.

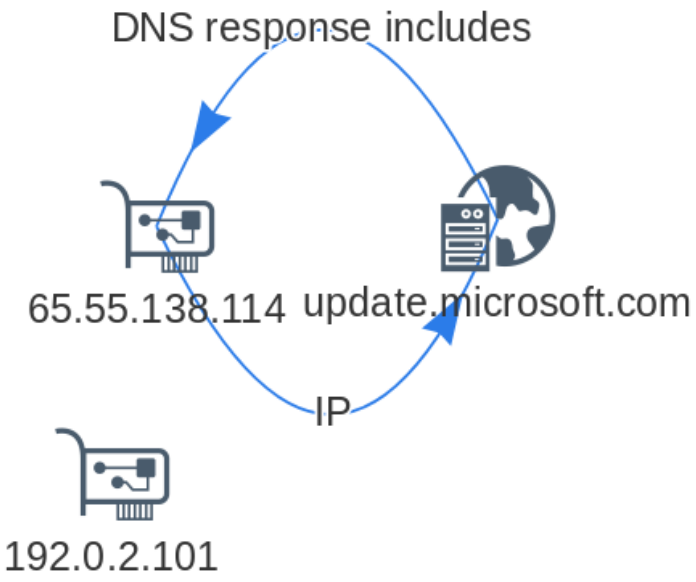
Listing 5.3 den zeigt wichtigsten Teil des Vorgangs im Code. Die Nachbarn des Ausgangsobservable werden nach Links mit der Beschreibung `DNS response includes` untersucht und die so verknüpften Hostnames gespeichert.

Anschließend werden auch die Hostnames auf Links mit derselben Beschreibung überprüft und die resultierenden IP-Adressen gespeichert. So sollten alle IP-Adressen gefunden worden sein, die aus dem Hostname aufgelöst wurden. IP-Adressen und Hostnames aus ThreatCrowd oder Passive DNS Datensätzen werden in diesem Schritt nicht berücksichtigt.

5.3.1 Probleme

Leider wird in der PCAP Investigation an keiner Stelle festgehalten, von welchem Host der DNS request kam. So können sich zwei Hosts `192.0.2.42` und `192.0.2.43` in Abbildung 5.1 mit einem dritten Host `87.98.167.141` verbunden haben. Letzterer wurde aus der Domain `microsoftwindowsupdate.net` aufgelöst. Aus den Kontexten

Abbildung 5.2: Kein DNS request ohne connection



der Hosts `192.0.2.42` und `192.0.2.43` geht hervor, dass sie DNS Verbindungen aufgebaut haben. Nun kann nicht festgelegt werden, ob `192.0.2.42` oder `192.0.2.43` den DNS Request gesendet hat. Hat `192.0.2.42` stattdessen `65.55.138.114` per DNS aufgelöst und besteht eine weitere Verbindung zwischen beiden, kann nun nicht mehr eindeutig gesagt werden, dass `192.0.2.42` auch `87.98.167.141` aufgelöst hat. So können Hosts fälschlicherweise mit `infected` und `data-leak` markiert werden.

Ein weiterer daraus resultierender Mangel wiegt umso schwerer. Ohne TCP- oder UDP-Verbindung wird kein Link zwischen den Observables erzeugt. Somit kann nicht festgestellt werden, ob der Host DNS requests für Hostnames erzeugt hat ohne sich anschließend mit den dazugehörigen IP-Adressen zu verbinden. Daraus folgt, dass ein Host, der als `infected` gewertet wurde, auch immer mit `data-leak` markiert wird. Auf Abbildung 5.2 ist klar sichtbar: Ohne eine Verbindung mit `65.55.138.114` wird nicht ersichtlich, dass `192.0.2.101` die Domain `update.microsoft.com` aufgelöst hat.

5.4 Einbindung von Passive DNS Daten

Für die Arbeit wurde das Oneshot Analytics Modul für den Circl Passive DNS Dienst berücksichtigt. Andere Anbieter sollten aber vergleichbar integriert sein.

Es wird vorausgesetzt, dass der Nutzer zuvor das Passive DNS Modul auf einen externen Host seiner Wahl angewendet hat. Das Resultat wird in den Context des Observables geschrieben. Durch Inline Analytics Module werden für die in der Antwort enthaltenen IP-Adressen, Domainnames und Hostnames zusätzlich Observables erzeugt und mit dem Ausgangsobservable verbunden. So können die Daten leicht in die Bewertung in Abschnitt 5.5 einfließen.

5.5 Einbindung von ThreatCrowd

Die explizite Erwähnung von ThreatCrowd in der Abschnittsbezeichnung deutet bereits an, dass zugunsten geringerer Komplexität eine stärkere Abhängigkeit zum ThreatCrowd Modul gewählt wurde. Die Reputationswerte werden nicht automatisch aufbereitet wie Hostnames oder IP-Adressen. So muss im Falle von ThreatCrowd der Context des Ausgangsobservables nach dem Reputationswert durchsucht werden. Auch ThreatCrowd liefert zum Host weitere IP-Adressen und Hostnames, die wieder mit dem Ausgangsobservable und Daten aus YETI verbunden werden.

Auch hier ist die Nutzerin in der Verantwortung das ThreatCrowd Modul auf dem Observable anzuwenden, dessen Reputation sie anschließend einschätzen lassen möchte.

Listing 5.4 zeigt den Code, der entscheidet, ob das Observable auffällig ist. Ist der von ThreatCrowd gelieferte `score` nicht `0`, wird die Bewertung von ThreatCrowd übernommen. So können Fehleinschätzungen von ThreatCrowd nicht festgestellt werden, die in YETI vorliegenden Daten werden nicht berücksichtigt.

Kann ThreatCrowd zum angefragten Host keine Aussage treffen oder sind keine Antworten von ThreatCrowd im Observables hinterlegt, werden die Nachbarn des Hosts im Graphen ausgewertet.

Die *Auswertung* beschränkt sich darauf, Nachbarn auf Tags zu überprüfen. Sind Tags vorhanden, die nicht auf die Ignore list gesetzt wurden[1], wird der Nachbar vermerkt.

[1] Zu Details und Einrichtung siehe Abschnitt B.2.

Listing 5.4: Ist das Observable *böse*?

```
102     def has_bad_reputation(observable, settings):
103         """
104         Parse ThreatCrowd output for reputation value.
105 
106         If 'votes' is '-1' observable is tagged as 'evil'.
107         If 'votes' is '0' links are searched for evil neighbours.
108         """
109         logging.debug("{} has context: {}".format(
110             observable,
111             observable.context))
112         score = 0
113         for entry in observable.context:
114             if entry['source'] == 'threatcrowd_query':
115                 if 'votes' in entry['raw']:
116                     try:
117                         raw_json = json.loads(entry['raw'])
118                     except Exception as e:
119                         logging.error(
120                         "Exception while reading raw values:{}".
                                format(e.message))
121                     logging.debug(
122                     "Raw json values are {}".format(raw_json))
123                     score = raw_json["votes"]
124                     logging.debug("{} has score {}".format(observable
                            , score))
125         logging.debug("Score is {}".format(score))
126         if score == -1:
127             logging.debug("Observable shall be tagged as 'evil'")
128             return True
129         if score == 0:
130             if Tagger.has_tagged_neighbors(observable, settings):
131                 logging.debug("Observable shall be tagged as 'evil'")
132                 return True
133         logging.debug("Observable shall *not* be tagged as 'evil'")
134         return False
```

Sind Nachbarn mit Tags vorhanden, hat das betrachtete Observable eine schlechte Reputation.

Die Verwendung von Tags beruht auf der Annahme, dass Observables mit Tags markiert werden, wenn sie auffällig sind. Der Nutzer muss darauf achten, dass durch fehlende Tags auf der Ignore list keine False Positives erzeugt werden. Als Nebenprodukt propagieren einmal gesetzte `evil` Tags durch den Graphen, bis sie bei Nachbarn von internen Hosts angelangen. Sie dienen dann als Grundlage für die Einstufung des internen Hosts. Voraussetzung dafür ist, dass die Nutzerin hinreichend viele Observables mit dem Tagger überprüft.

6 Tests

Die Szenarien in diesem Kapitel sollen zeigen, dass die Anforderungen aus Abschnitt 3.2 umgesetzt wurden. Zudem soll nachvollzogen werden können, ob die Implementation der Dokumentation in Kapitel 5 folgt.

6.1 Einstufung externer Hosts

Es soll verifiziert werden, dass eine von ThreatCrowd als *schädlich* eingestufte Domain zu einer entsprechenden Markierung im Graphen führt.

Um eine passende Domain zu finden, genügt ein Blick auf die Liste schädlicher Domains[1] bei ThreatCrowd[2]. Für schädliche IP-Adressen existiert eine weitere Liste[3].

Dazu wird eine beliebige Domain – im Beispiel `microsoftwindowsupdate.net` – von der Liste gewählt und einer Investigation als `Hostname` Observable hinzugefügt. Nachdem ThreatCrowd auf dem Observable ausgeführt wurde, folgt ein Aufruf des Threat Taggers. Das Observable sollte jetzt mit `evil` markiert sein, wie in Abbildung 6.1 rot hervorgehoben wurde.

6.2 Einstufung interner Hosts

Das Szenario aus Abschnitt 6.1 wird um einen internen Host erweitert. Dieser hat die Domain `microsoftwindowsupdate.net` aufgelöst und sich mit einer der zurückgegebenen IP-Adressen verbunden. Nach Anwenden des Threat Taggers auf die externe IP-Adresse `87.98.167.141` und die interne IP-Adresse `192.0.2.42` sollte Letztere

[1] https://www.threatcrowd.org/feeds/domains.txt
[2] Weitere Details zu den Listen: https://threatcrowd.blogspot.com/2016/02/crowdsourced-feeds-from-threatcrowd.html
[3] https://www.threatcrowd.org/feeds/ips.txt

Abbildung 6.1: Eine schädliche Domain

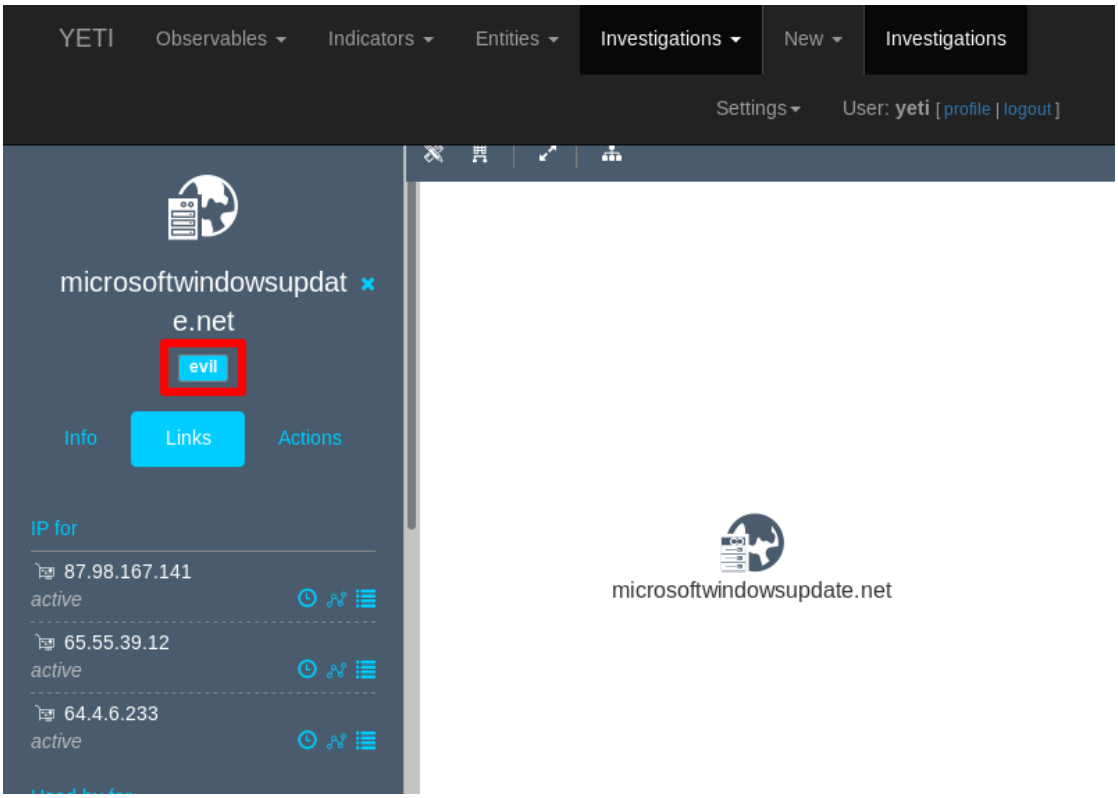
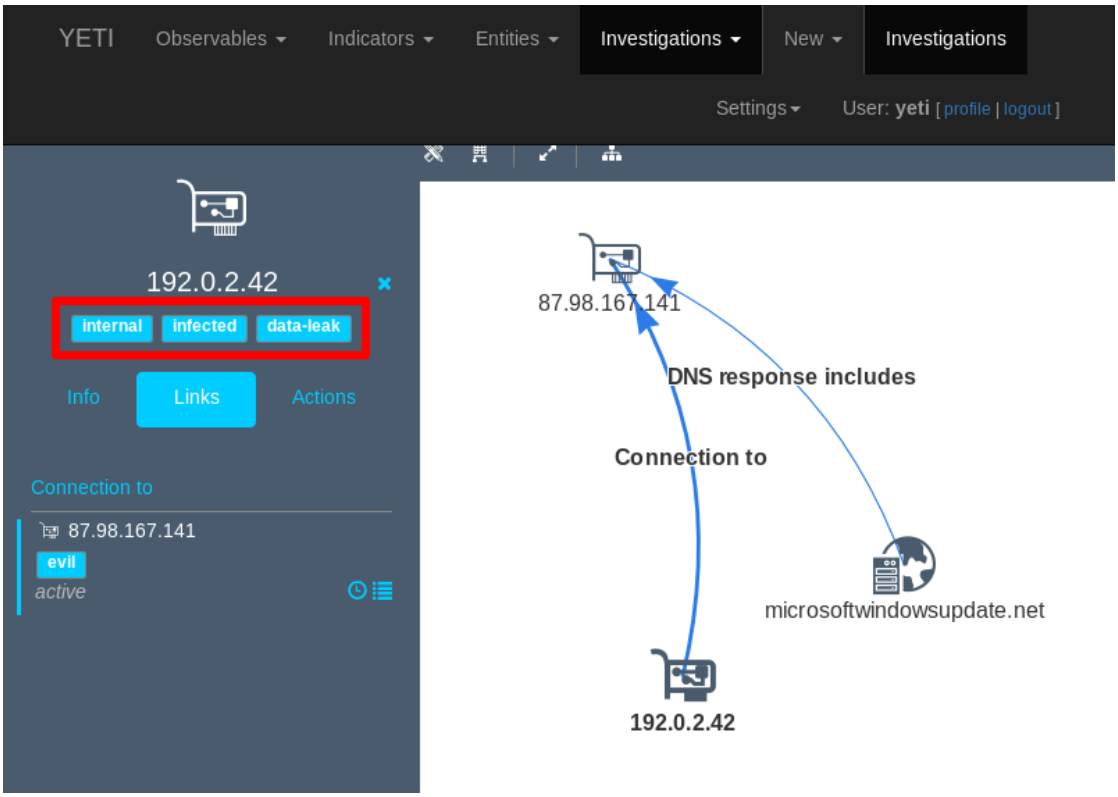


Abbildung 6.2: Ein interner Host hat eine Verbindung mit einer schädlichen IP-Adresse aufgebaut

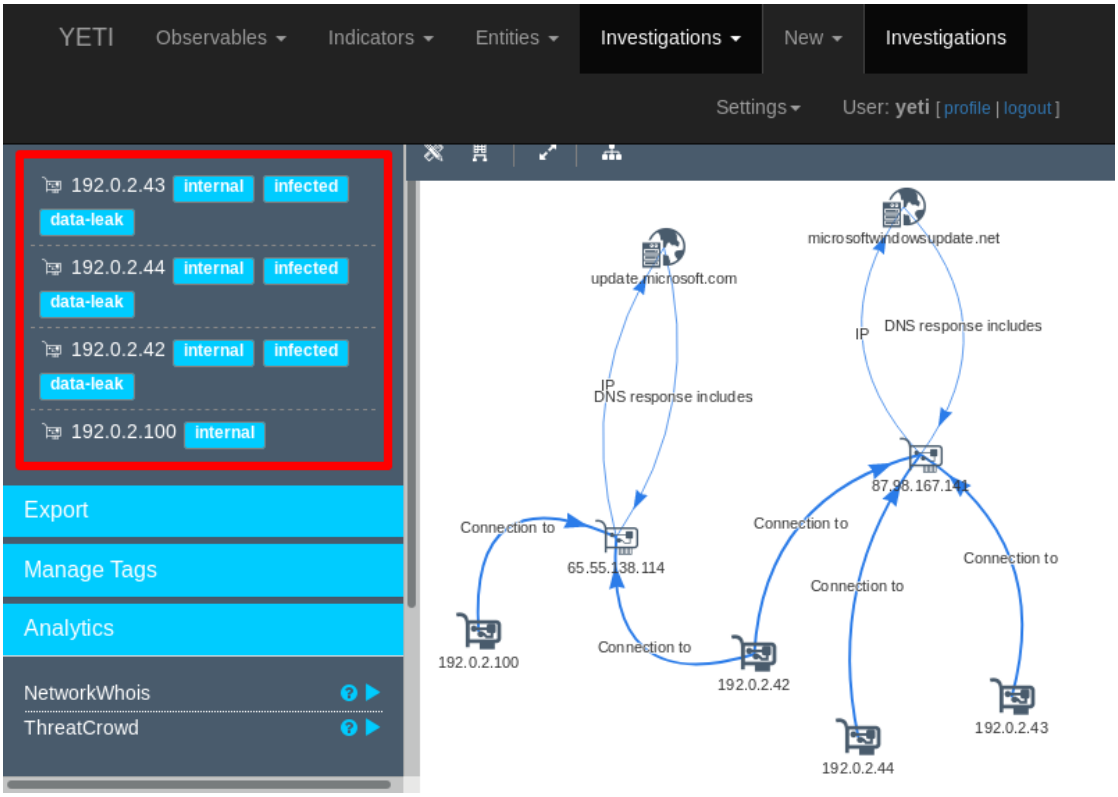


als `infected` und `data-leak` markiert worden sein. Das Ergebnis ist in Abbildung 6.2 rot markiert. Die Erzeugung des Szenarios ist nicht innerhalb einer Investigation möglich, sondern muss durch ein Skript vorbereitet werden. Abschnitt 6.4 zeigt die Erstellung der Testdaten.

6.3 Komplexes Szenario

Das in Abbildung 6.3 dargestellte Szenario soll eine alltäglichere Situation repräsentieren. Drei interne Hosts lösen die *böse* Domain `microsoftwindowsupdate.net` auf, während zwei Hosts auch die *gute* Domain `update.microsoft.com` auflösen. Nach Abfragen der Domains bei ThreatCrowd und Einstufung derselbigen und deren IP-Adressen sollte das in Abbildung 6.3 markierte Ergebnis vorliegen. Bis auf den internen Host `192.0.2.100` sollten alle anderen internen Hosts als `infected` und `data-leak`

Abbildung 6.3: *Gute* und *Böse* Hosts in einem Netzwerk



markiert worden sein.

6.4 Erstellung der Testszenarien

Im Kontext von Investigations können über das Webfrontend keine komplexen Szenarien erstellt werden. Obervables aus der Datenbank können nur bei der Erstellung einer Investigation genutzt werden, danach können Observables nur durch Analytics Module aus der Datenbank bezogen werden. Links, die in der Investigation erstellt werden, können nicht in die Datenbank gespeichert werden[4]. Wenn für das Testszenario kein PCAP-Mitschnitt zur Verfügung steht, können stattdessen mit einem Skript Observables und Links in die Datenbank geschrieben werden. Für die Reproduktion der Testdaten kann das Skript `createtestobjects.py` aus den Quellen genutzt werden.

[4]Siehe dazu die Notiz am Ende von Use cases – Ingesting and enriching a third-party report – Starting an Investigation in der Dokumentation[3].

7 Fazit

Ziel der Arbeit war es innerhalb von YETI verschiedene Datenquellen zu verknüpfen und auszuwerten. Die Umsetzung innerhalb der vorgesehenen Erweiterungsschnittstellen ist durch die Verwendung der Analytics Module erfolgt. Über den üblichen Einsatzzweck hinaus wurden die Relationen eines einzelnen Observables und die Rückgaben anderer Module analysiert. Dabei wurden Eigenheiten der Plattform und anderer Module festgestellt und dokumentiert. Die Stärke von YETI, unzählige Informationen miteinander zu verknüpfen, sollte an der einen oder anderen Stelle sichtbar geworden sein. Am Ende steht ein Modul, dass eine Hilfestellung für die Einschätzung von Hosts in Netzwerken bietet. Die Präsentation der Ergebnisse und der Automatisierungsgrad sind sicherlich noch ausbaufähig.

7.1 Ausblick

Schon die bestehende Arbeit kann durch Detailverbesserungen profitieren. So sollten die Schritte in Anhang B entfallen.

Vom Nutzer wird erwartet, dass er seine Hosts manuell mit Tags versieht. Die internen Hosts könnten beim Import des PCAP Dumps bereits passend getaggt werden. Dazu müsste auch die PCAP Investigation Erweiterung angepasst werden, um beispielsweise eine Liste der internen IP-Adressen oder des gesamten Adressbereichs entgegenzunehmen.

Bevor das Modul nutzbar ist, müssen Tags auf die Ignore list gesetzt werden. Dies ist dem Umstand geschuldet, dass YETI keine leeren Optionen erlaubt. Es ist nicht vorgesehen, dass über Moduleinstellungen Parameter geändert werden, die über API-Keys oder Kombinationen aus Passwort und Benutzerkonto hinausgehen. Wünschenswert wären Standardwerte, die im Modul vorgegeben werden und vom Nutzer geändert werden

können. Dann müssten die intern verwendeten Tags auch nicht hart in das Modul geschrieben werden.

Der `ThreatTagger` sollte nach jedem Durchlauf einen Bericht erstellen, der in der Übersicht über die verfügbaren Analytics Module für ein Observable abrufbar ist. Darin soll aufgelistet werden, ob Tags gesetzt werden und weshalb dies geschieht.

Für eine bessere Visualisierung der Tags sollten die Knoten und Kanten des Graphen entsprechend eingefärbt werden.

Literaturverzeichnis

[1] Antonakakis, Manos ; Perdisci, Roberto ; Dagon, David ; Lee, Wenke ; Feamster, Nick: *Building a Dynamic Reputation System for DNS*. 2010. – URL http://static.usenix.org/event/sec10/tech/full_papers/Antonakakis.pdf

[2] CERT-UK: *An introduction to threat intelligence*. 2015. – URL https://web.archive.org/web/20160920192621/https://www.cert.gov.uk/wp-content/uploads/2015/03/An-introduction-to-threat-intelligence.pdf

[3] Chopitea, Thomas ; Muller, Gael: *YETI documentation*. 2016. – URL https://yeti-platform.readthedocs.io/en/latest/index.html

[4] Diedrich, Nick: *Erkennung von Angriffsmustern in Passiv-DNS-Daten*. 2017. – URL http://edoc.sub.uni-hamburg.de/haw/volltexte/2017/4012/pdf/Bachelorarbeit_Nick_Diedrich.pdf

[5] Friedman, Jon ; Bouchard, Mark: *Definitive Guide to Cyber Threat Intelligence*. 1997

[6] Mockapetris, P.: *Domain Names, Implementation and Specification*. 1987. – URL https://www.rfc-editor.org/rfc/rfc1035

[7] Stammerjohann, Tim: *Erweiterung einer Threat-Intelligence-Plattform zur Verarbeitung von Netzwerkverkehrsdaten im PCAP-Format*. 2017

[8] threatcrowd: *ThreatCrowd API v2*. 2018. – URL https://github.com/AlienVault-OTX/ApiV2

[9] Wagner, Cynthia ; Dulaunoy, Alexandre ; Wagener, Gérard ; Iklody, Andras: *MISP, The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform*. 2016

[10] Weimer, Florian: *Passive DNS Replication*. 2005. – URL http://static.enyo.de/fw/volatile/pdr-draft-11.pdf

A Installation

Das auf der CD beiliegende YETI-Release ist auf den Stand von Commit `554db1e`[1] aus dem *master* Zweig. Es wurde bereits um die PCAP-Investigation erweitert. Die Installation von YETI beschreibt die Dokumentation[3]. Der Aufruf von `pip install -r requirements.txt` installiert auch die Abhängigkeiten des PCAP Patches. Insbesondere setzt der Patch das Python Modul *dpkt* in Version *1.9.1* voraus.

Nach der Installation wird das Modul `tagger.py` in den Ordner `yeti/plugins/analytics/public/` kopiert. Nach einem Neustart des Systemd-Services `yeti_oneshot` oder von `yeti.py` steht das Modul unter *Oneshot analytics* bereit und muss noch konfiguriert werden, wie in Abschnitt B.2 beschrieben wird.

Um die Testdaten erzeugen zu können, sollte das Skript `createtestobjects.py` in den Ordner `yeti/tests/` kopiert und dort ausgeführt werden.

[1] https://github.com/yeti-platform/yeti/commit/554db1

B Einrichtung

B.1 Vorbereitung der Hosts

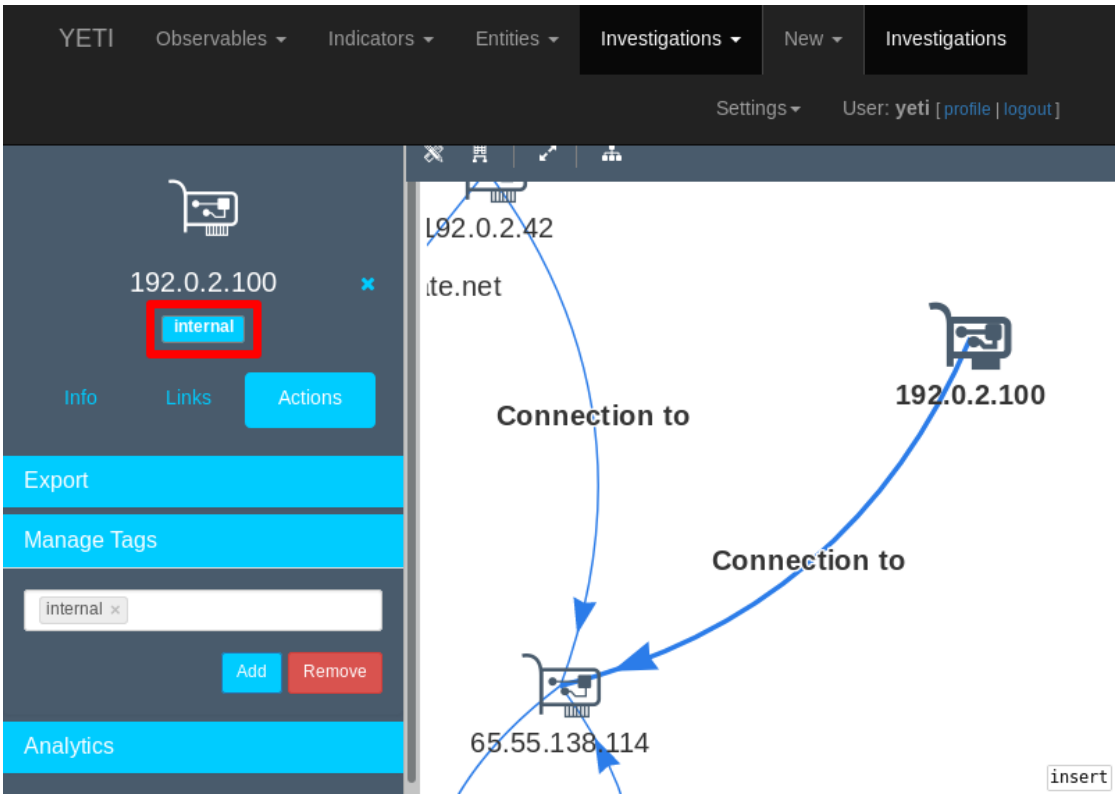
Nach der Erstellung einer PCAP-Investigation müssen noch die internen Hosts markiert werden. In der Graphansicht kann dazu dem Host, der meist durch eine IP-Adresse repräsentiert wird, das Tag `internal` hinzugefügt werden. Dazu können auch mehrere Knoten auf einmal markiert werden. Abbildung B.1 zeigt ein Observable, bei dem bereits `internal` hinzugefügt wurde.

B.2 Konfiguration des Moduls

Nach der Installation ist das Modul noch nicht im Kontextmenü eines Knoten verfügbar, es muss erst in den Benutzereinstellungen konfiguriert werden[1]. Die Einstellung `Ignore tags` innerhalb der *Miscellaneous settings* muss mindestens einen Tag enthalten. Die konfigurierten Tags und die internen Tags `internal`, `infected` und `data-leak` werden bei der Analyse nicht berücksichtigt. Sollen keine zusätzlichen Tags berücksichtigt werden, kann einfach ein bereits definiertes internes Tag verwendet werden. Mehrere Tags werden durch Leerzeichen getrennt. Nach Speichern der Einstellung ist das Modul aktiv.

[1] In der Menüleiste zu erreichen über `profile` neben dem Benutzernamen.

Abbildung B.1: Mit `internal` getaggt



Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Erweiterung einer Threat Intelligence Plattform um Abfrage und Auswertung von Daten von Passive DNS Sensoren am Beispiel von YETI

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original