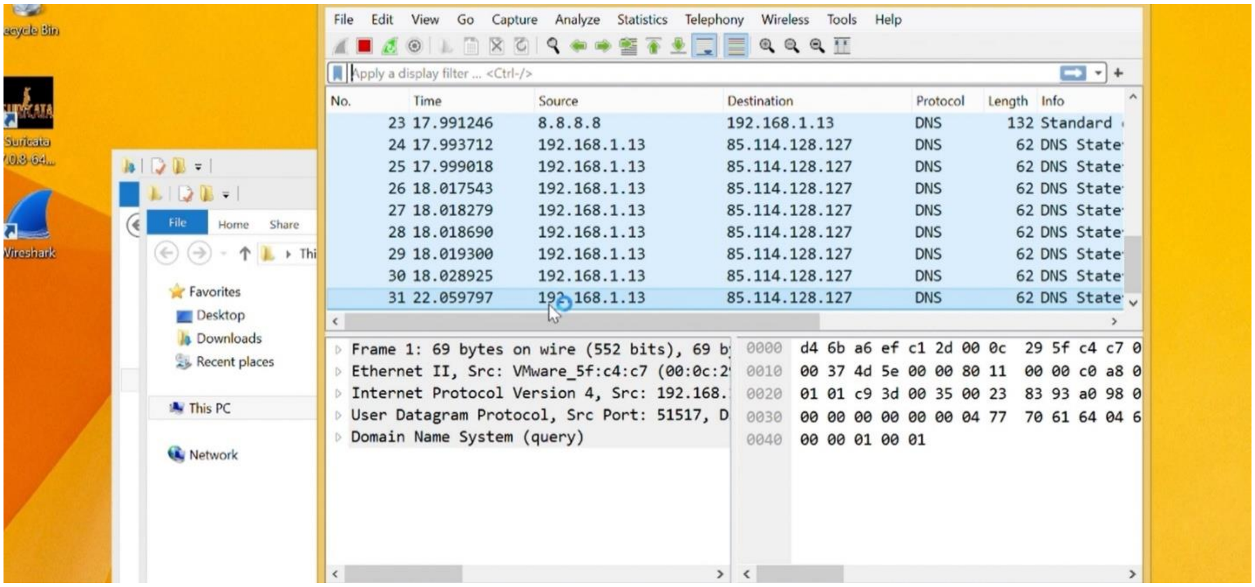
Suricata analysis and network capturing

1. I began by capturing network traffic using Wireshark while executing the Zeus malware for analysis.



ecycle Bin
Suricata
0.8-64...
Wireshark
File
Home
Share
Thi
Favorites
Desktop
Downloads
Recent places
This PC
Network
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
23	17.991246	8.8.8.8	192.168.1.13	DNS	132	Standard
24	17.993712	192.168.1.13	85.114.128.127	DNS	62	DNS State
25	17.999018	192.168.1.13	85.114.128.127	DNS	62	DNS State
26	18.017543	192.168.1.13	85.114.128.127	DNS	62	DNS State
27	18.018279	192.168.1.13	85.114.128.127	DNS	62	DNS State
28	18.018690	192.168.1.13	85.114.128.127	DNS	62	DNS State
29	18.019300	192.168.1.13	85.114.128.127	DNS	62	DNS State
30	18.028925	192.168.1.13	85.114.128.127	DNS	62	DNS State
31	22.059797	192.168.1.13	85.114.128.127	DNS	62	DNS State

```
Frame 1: 69 bytes on wire (552 bits), 69 b
Ethernet II, Src: VMware_5f:c4:c7 (00:0c:2
Internet Protocol Version 4, Src: 192.168.
User Datagram Protocol, Src Port: 51517, D
Domain Name System (query)
```

```
0000  d4 6b a6 ef c1 2d 00 0c  29 5f c4 c7 0
0010  00 37 4d 5e 00 00 80 11  00 00 c0 a8 0
0020  01 01 c9 3d 00 35 00 23  83 93 a0 98 0
0030  00 00 00 00 00 00 04 77  70 61 64 04 6
0040  00 00 01 00 01
```

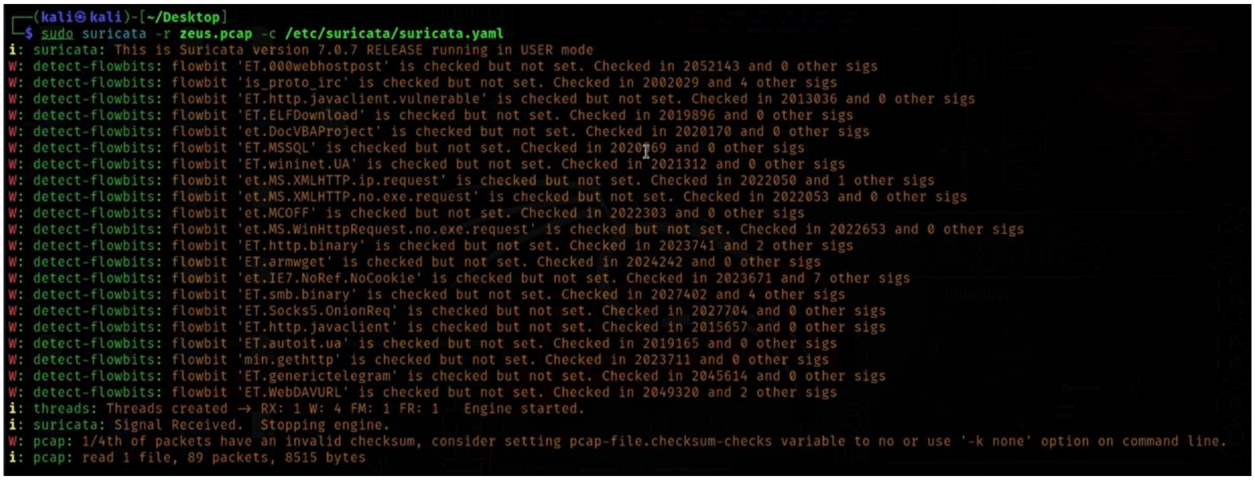
2. I incorporate the default rule sets for Suricata from the official Emerging Threats repository, specifically the *emerging-malware.rules* and *emerging-phishing.rules* files.

3. I have developed a set of detection rules specifically designed to identify Zeus malware. Below is a detailed explanation of these rules.

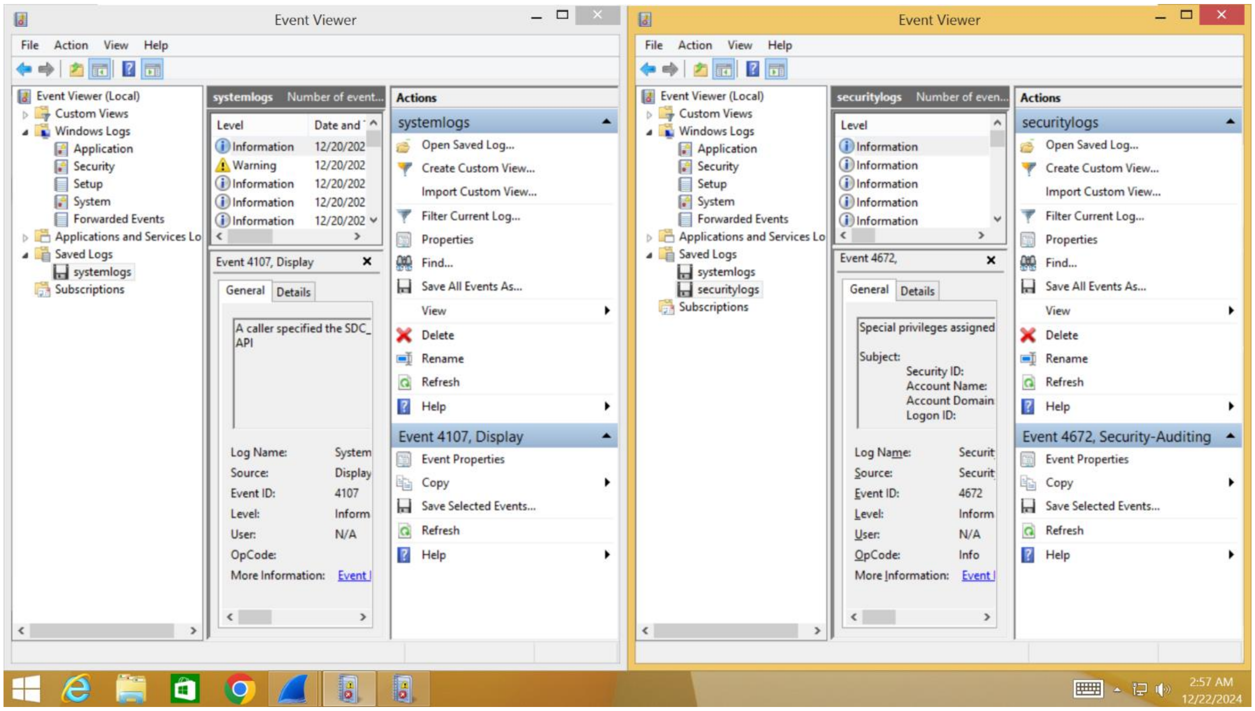
- **HTTP C2 Traffic Detection**
  Detects Zeus C2 communication using HTTP POST requests to URIs containing /gate.php within the first 10 bytes of the URI. Applicable for traffic directed to the server in established sessions (SID:100001).
- **Config File Download Detection**
  Identifies Zeus downloading configuration files by matching the User-Agent header (MSIE 6.0) and requests for /config.bin, in established connections to the server (SID:100003).
- **Specific Data Pattern in HTTP Traffic**
  Flags Zeus traffic containing specific byte patterns (DE AD BE EF and FE ED FA CE within 50 bytes) in HTTP data to the server (SID:100007).

- **DNS Query for Known Domain**
  Detects DNS queries for fpdownload.macromedia.com, a domain linked with Zeus activity (SID:100009).

4. Execute Suricata with the specified ruleset, then extract and export the Suricata logs along with the system and security logs.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo suricata -r zeus.pcap -c /etc/suricata/suricata.yaml
i: suricata: This is Suricata version 7.0.7 RELEASE running in USER mode
W: detect-flowbits: flowbit 'ET.000webhostpost' is checked but not set. Checked in 2052143 and 0 other sigs
W: detect-flowbits: flowbit 'is_proto_irc' is checked but not set. Checked in 2002029 and 4 other sigs
W: detect-flowbits: flowbit 'ET.http.javaclient.vulnerable' is checked but not set. Checked in 2013036 and 0 other sigs
W: detect-flowbits: flowbit 'ET.ELFDownload' is checked but not set. Checked in 2019896 and 0 other sigs
W: detect-flowbits: flowbit 'et.DocVBAProject' is checked but not set. Checked in 2020170 and 0 other sigs
W: detect-flowbits: flowbit 'ET.MSSQL' is checked but not set. Checked in 2020769 and 0 other sigs
W: detect-flowbits: flowbit 'ET.wininet.UA' is checked but not set. Checked in 2021312 and 0 other sigs
W: detect-flowbits: flowbit 'et.MS.XMLHTTP.ip.request' is checked but not set. Checked in 2022050 and 1 other sigs
W: detect-flowbits: flowbit 'et.MS.XMLHTTP.no.exe.request' is checked but not set. Checked in 2022053 and 0 other sigs
W: detect-flowbits: flowbit 'et.MCOFF' is checked but not set. Checked in 2022303 and 0 other sigs
W: detect-flowbits: flowbit 'et.MS.WinHttpRequest.no.exe.request' is checked but not set. Checked in 2022653 and 0 other sigs
W: detect-flowbits: flowbit 'ET.http.binary' is checked but not set. Checked in 2023741 and 2 other sigs
W: detect-flowbits: flowbit 'ET.armwget' is checked but not set. Checked in 2024242 and 0 other sigs
W: detect-flowbits: flowbit 'et.IE7.NoRef.NoCookie' is checked but not set. Checked in 2023671 and 7 other sigs
W: detect-flowbits: flowbit 'ET.smb.binary' is checked but not set. Checked in 2027402 and 4 other sigs
W: detect-flowbits: flowbit 'ET.Socks5.OnionReq' is checked but not set. Checked in 2027704 and 0 other sigs
W: detect-flowbits: flowbit 'ET.http.javaclient' is checked but not set. Checked in 2015657 and 0 other sigs
W: detect-flowbits: flowbit 'ET.autoit.ua' is checked but not set. Checked in 2019165 and 0 other sigs
W: detect-flowbits: flowbit 'min.gethttp' is checked but not set. Checked in 2023711 and 0 other sigs
W: detect-flowbits: flowbit 'ET.generictelegram' is checked but not set. Checked in 2045614 and 0 other sigs
W: detect-flowbits: flowbit 'ET.WebDAVURL' is checked but not set. Checked in 2049320 and 2 other sigs
i: threads: Threads created → RX: 1 W: 4 FM: 1 FR: 1   Engine started.
i: suricata: Signal Received.  Stopping engine.
W: pcap: 1/4th of packets have an invalid checksum, consider setting pcap-file.checksum-checks variable to no or use '-k none' option on command line.
i: pcap: read 1 file, 89 packets, 8515 bytes
```



Level	Date and
Information	12/20/202
Warning	12/20/202
Information	12/20/202
Information	12/20/202
Information	12/20/202

Event 4107, Display

A caller specified the SDC_ API

Log Name:	System
Source:	Display
Event ID:	4107
Level:	Inform
User:	N/A
OpCode:	
More Information:	Event l

Level
Information
Information
Information
Information
Information

Event 4672,

Special privileges assigned

Subject:
Security ID:
Account Name:
Account Domain
Logon ID:

Log Name:	Securit
Source:	Securit
Event ID:	4672
Level:	Inform
User:	N/A
OpCode:	Info
More Information:	Event l

Splunk Analysis

Name: Abdelrahman Farid Elsaid 2106145

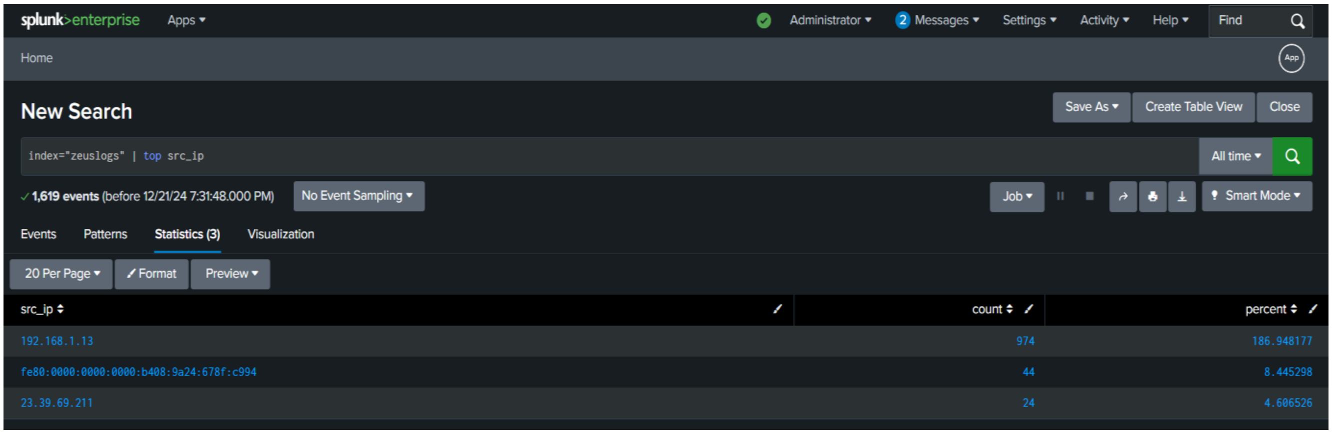
Basic Investigation

The first step was to ingest 4 Important Files into Splunk

1) Security Events Logs (csv)
2) System Events Logs (csv)
3) Suricata Alert Logs (json)
4) Another Suricata Alert Logs (txt)

The Next Step was to Perform Basic Search Across all files to gather Information

1) Perform this SPL Query to get the top Source IP
   (index="zeuslogs" | top src_ip)



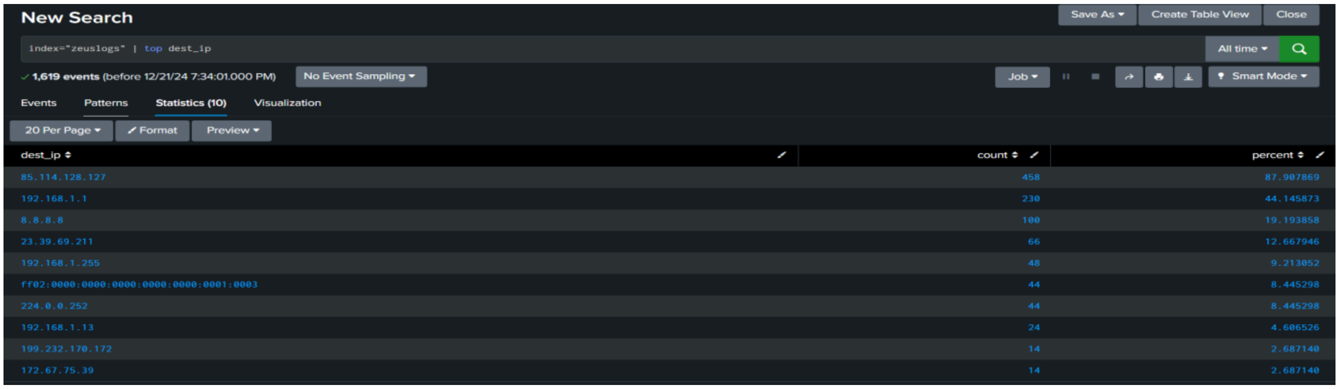
New Search

```
index="zeuslogs" | top src_ip
```

✓ 1,619 events (before 12/21/24 7:31:48.000 PM)

src_ip	count	percent
192.168.1.13	974	186.948177
fe80:0000:0000:0000:b408:9a24:678f:c994	44	8.445298
23.39.69.211	24	4.606526

2) Perform this SPL Query to get the top Destination IP
   (index="zeuslogs" | top dest_ip)



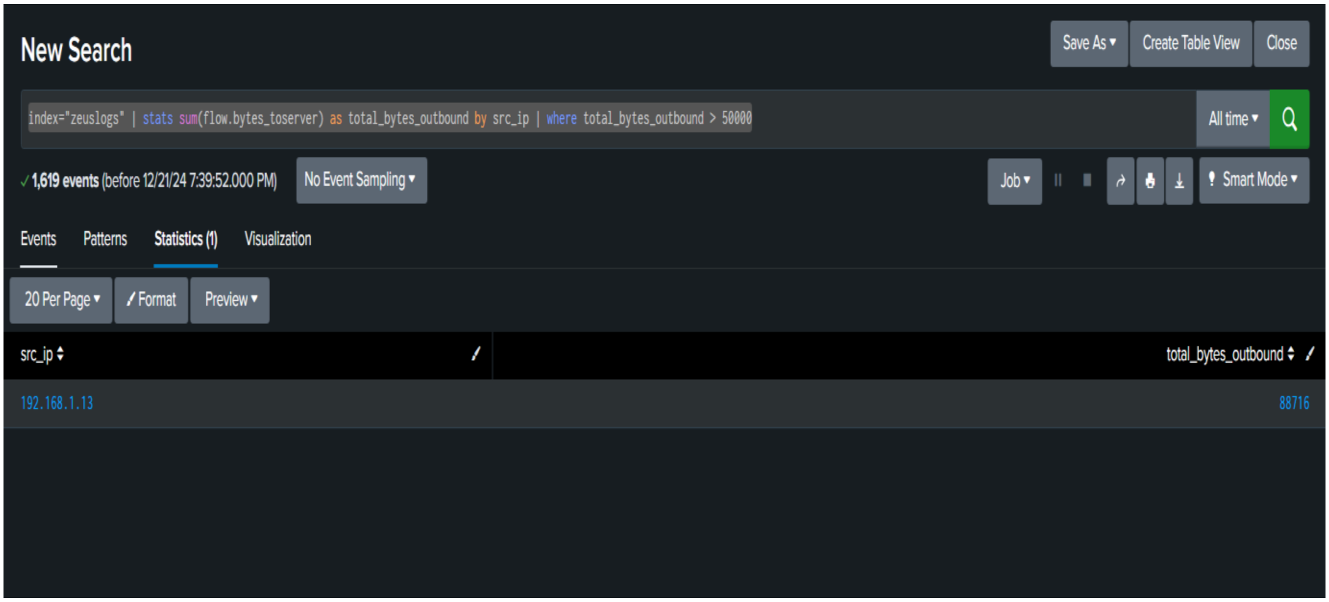
New Search

```
index="zeuslogs" | top dest_ip
```

✓ 1,619 events (before 12/21/24 7:34:01.000 PM)

dest_ip	count	percent
85.114.128.127	458	87.907869
192.168.1.1	230	44.145873
8.8.8.8	100	19.193858
23.39.69.211	66	12.667946
192.168.1.255	48	9.213052
ff02:0000:0000:0000:0000:0000:0001:0003	44	8.445298
224.0.0.252	44	8.445298
192.168.1.13	24	4.606526
199.232.170.172	14	2.687140
172.67.75.39	14	2.687140

3) Perform this SPL Query to identify IPs generating Significant Outbound Traffic based on bytes (index="zeuslogs" | stats sum(flow.bytes_toserver) as total_bytes_outbound by src_ip | where total_bytes_outbound > 50000)

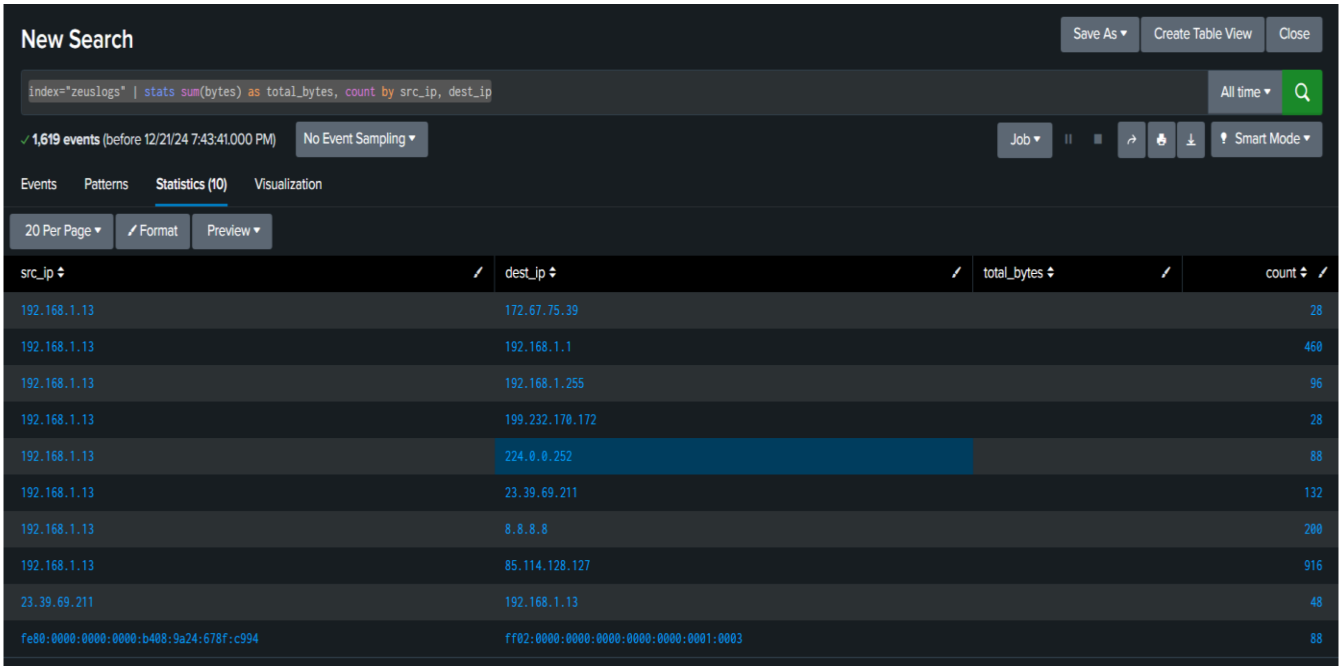


```
index="zeuslogs" | stats sum(flow.bytes_toserver) as total_bytes_outbound by src_ip | where total_bytes_outbound > 50000
```

✓ 1,619 events (before 12/21/24 7:39:52.000 PM)

src_ip	total_bytes_outbound
192.168.1.13	88716

4) Perform this SPL Query to count the number of events between source and destination IPs (index="zeuslogs" | stats sum(bytes) as total_bytes, count by src_ip, dest_ip)

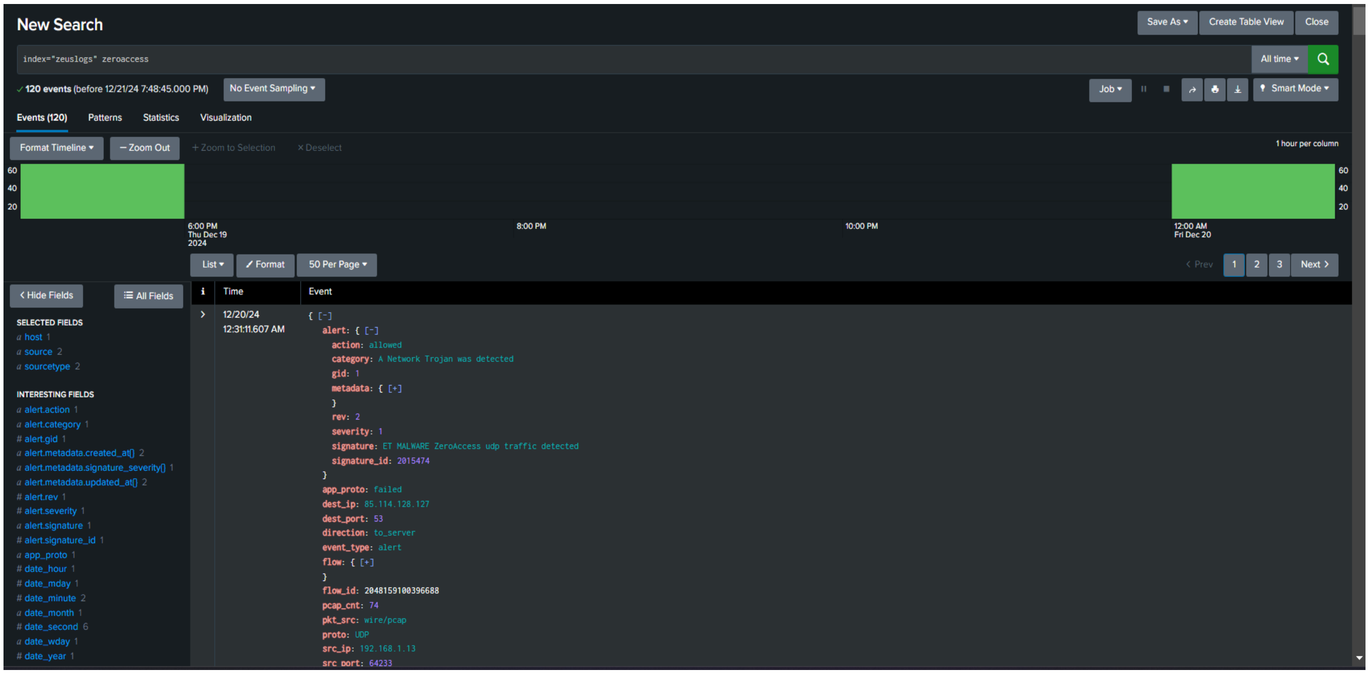


```
index="zeuslogs" | stats sum(bytes) as total_bytes, count by src_ip, dest_ip
```

✓ 1,619 events (before 12/21/24 7:43:41.000 PM)

src_ip	dest_ip	total_bytes	count
192.168.1.13	172.67.75.39		28
192.168.1.13	192.168.1.1		460
192.168.1.13	192.168.1.255		96
192.168.1.13	199.232.170.172		28
192.168.1.13	224.0.0.252		88
192.168.1.13	23.39.69.211		132
192.168.1.13	8.8.8.8		200
192.168.1.13	85.114.128.127		916
23.39.69.211	192.168.1.13		48
fe80:0000:0000:0000:b408:9a24:678f:c994	ff02:0000:0000:0000:0000:0000:0001:0003		88

5) Perform This SPL Query to Retrieve all Logs related to ZeroAcess Malware
(index="zeuslogs" zeroaccess)



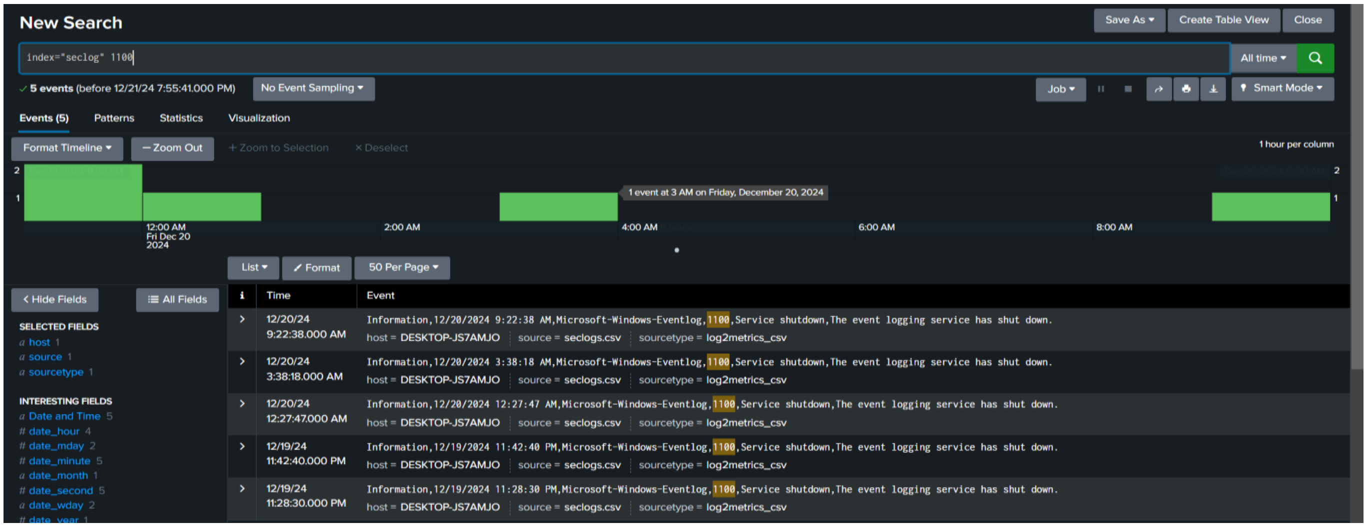
```
index="zeuslogs" zeroaccess
```

✓ 120 events (before 12/21/24 7:48:45.000 PM)

```
12/20/24      { [-]
12:31:11.607 AM  alert: { [-]
                   action: allowed
                   category: A Network Trojan was detected
                   gid: 1
                   metadata: { [+]
                   }
                   rev: 2
                   severity: 1
                   signature: ET MALWARE ZeroAccess udp traffic detected
                   signature_id: 2015474
                 }
                 app_proto: failed
                 dest_ip: 85.114.128.127
                 dest_port: 53
                 direction: to_server
                 event_type: alert
                 flow: { [+]
                 }
                 flow_id: 2048159100396688
                 pcap_cnt: 74
                 pkt_src: wire/pcap
                 proto: UDP
                 src_ip: 192.168.1.13
                 src_port: 64233
```

The Next Step is to Co-Relate Security Events with System Events to Detect Suspicious Behavior

1) Perform this SPL Query to retrieve all events related to security with event ID 1100 related to Shutdown of Logging Service and notice the event timed at 9:22:38 AM
(index=”seclog” 1100)



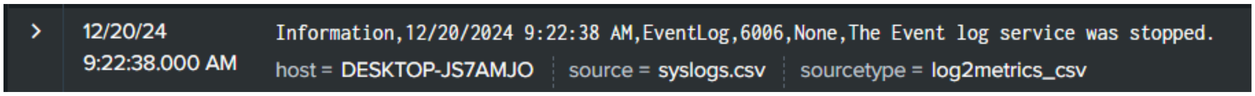
```
index="seclog" 1100
```

✓ 5 events (before 12/21/24 7:55:41.000 PM)

```
12/20/24         Information,12/20/2024 9:22:38 AM,Microsoft-Windows-Eventlog,1100,Service shutdown,The event logging service has shut down.
9:22:38.000 AM   host = DESKTOP-JS7AMJO | source = seclogs.csv | sourcetype = log2metrics_csv

12/20/24         Information,12/20/2024 3:38:18 AM,Microsoft-Windows-Eventlog,1100,Service shutdown,The event logging service has shut down.
3:38:18.000 AM   host = DESKTOP-JS7AMJO | source = seclogs.csv | sourcetype = log2metrics_csv

12/20/24         Information,12/20/2024 12:27:47 AM,Microsoft-Windows-Eventlog,1100,Service shutdown,The event logging service has shut down.
12:27:47.000 AM  host = DESKTOP-JS7AMJO | source = seclogs.csv | sourcetype = log2metrics_csv

12/19/24         Information,12/19/2024 11:42:40 PM,Microsoft-Windows-Eventlog,1100,Service shutdown,The event logging service has shut down.
11:42:40.000 PM  host = DESKTOP-JS7AMJO | source = seclogs.csv | sourcetype = log2metrics_csv

12/19/24         Information,12/19/2024 11:28:30 PM,Microsoft-Windows-Eventlog,1100,Service shutdown,The event logging service has shut down.
11:28:30.000 PM  host = DESKTOP-JS7AMJO | source = seclogs.csv | sourcetype = log2metrics_csv
```

The next step is to analyze the system logs in the same timing that we got from the security log that we identified earlier using this SPL Query, and notice this specific Log

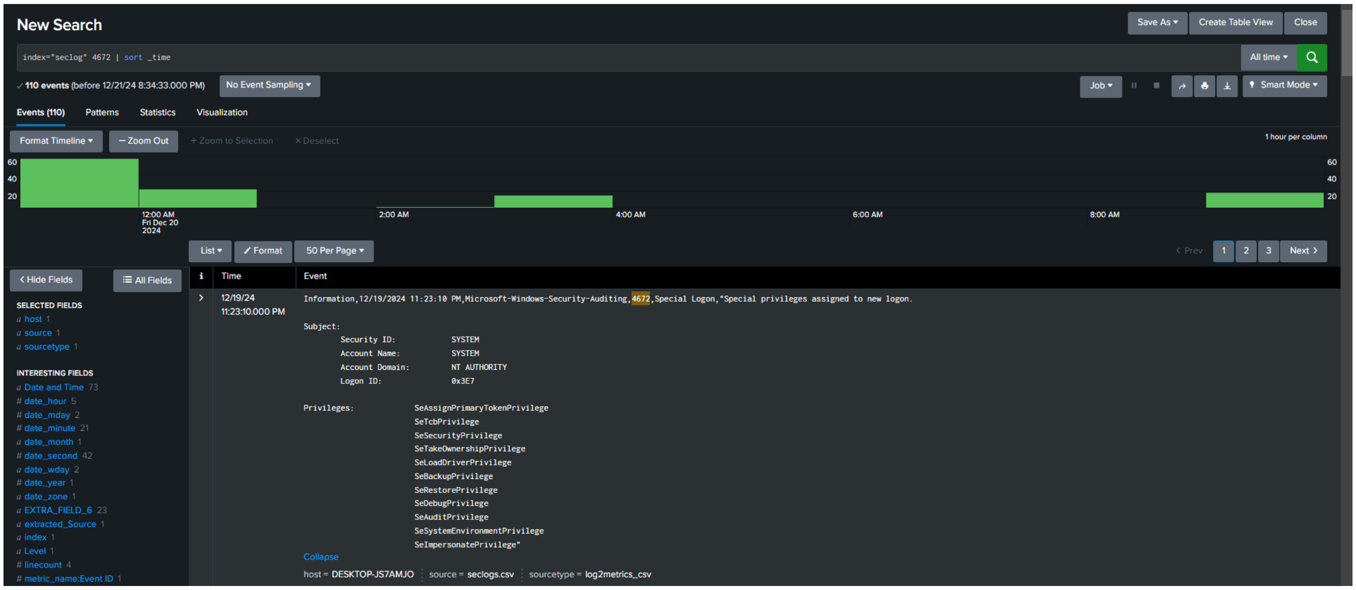
(index="syslogs" | sort -_time)



```
12/20/24
9:22:38.000 AM    Information,12/20/2024 9:22:38 AM,EventLog,6006,None,The Event log service was stopped.
                  host = DESKTOP-JS7AMJO | source = syslogs.csv | sourcetype = log2metrics_csv
```

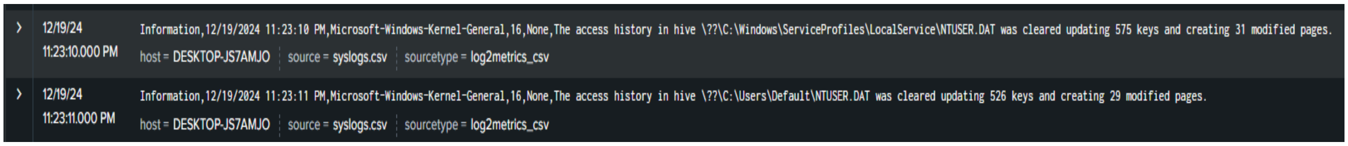
So, we can conclude that the Malware Stopped the logging service.

2) Perform this SPL Query to retrieve all events related to security with event ID 4672 related to Special privileges assigned to new logon and notice this event timed at 11:23:10 PM (index="seclog" 4672 | sort _time)



```
index="seclog" 4672 | sort _time
✓ 110 events (before 12/21/24 8:34:33.000 PM)

Time              Event
12/19/24          Information,12/19/2024 11:23:10 PM,Microsoft-Windows-Security-Auditing,4672,Special Logon,"Special privileges assigned to new logon.
11:23:10.000 PM
                  Subject:
                          Security ID:            SYSTEM
                          Account Name:           SYSTEM
                          Account Domain:         NT AUTHORITY
                          Logon ID:               0x3E7

                  Privileges:             SeAssignPrimaryTokenPrivilege
                                          SeTcbPrivilege
                                          SeSecurityPrivilege
                                          SeTakeOwnershipPrivilege
                                          SeLoadDriverPrivilege
                                          SeBackupPrivilege
                                          SeRestorePrivilege
                                          SeDebugPrivilege
                                          SeAuditPrivilege
                                          SeSystemEnvironmentPrivilege
                                          SeImpersonatePrivilege"
                  Collapse
                  host = DESKTOP-JS7AMJO | source = seclogs.csv | sourcetype = log2metrics_csv
```

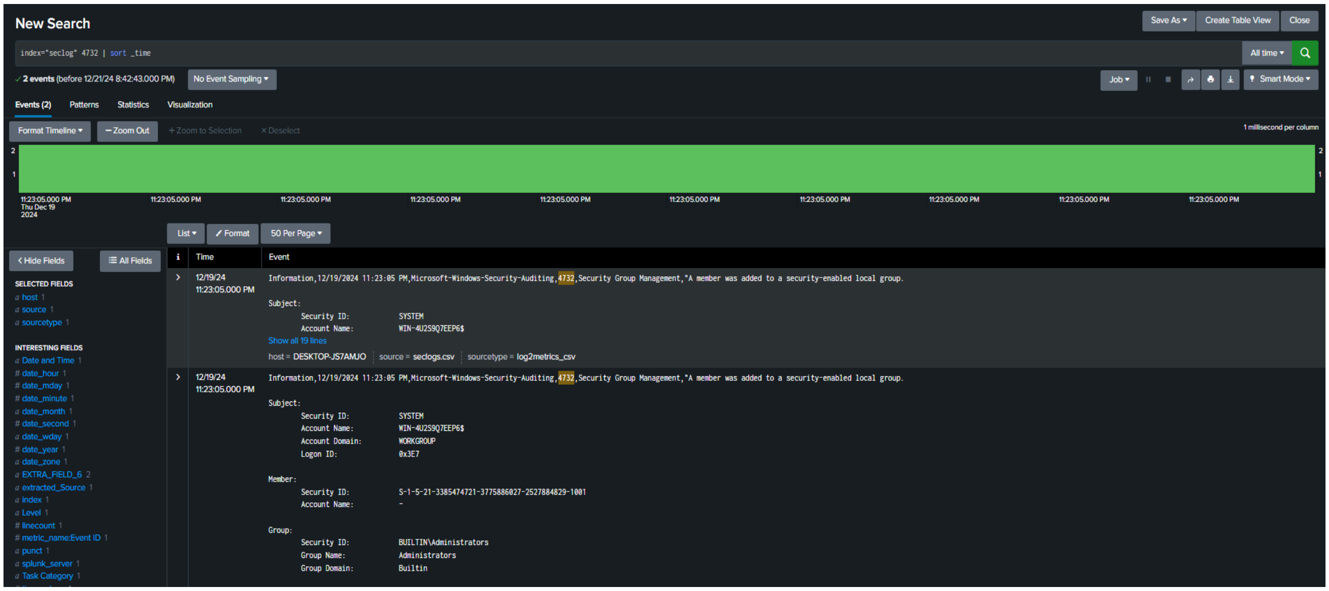
The next step is to analyze the system logs in the same timing that we got from the security log that we identified earlier using this SPL Query, and notice this specific Log



```
12/19/24          Information,12/19/2024 11:23:10 PM,Microsoft-Windows-Kernel-General,16,None,The access history in hive \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT was cleared updating 575 keys and creating 31 modified pages.
11:23:10.000 PM   host = DESKTOP-JS7AMJO | source = syslogs.csv | sourcetype = log2metrics_csv

12/19/24          Information,12/19/2024 11:23:11 PM,Microsoft-Windows-Kernel-General,16,None,The access history in hive \??\C:\Users\Default\NTUSER.DAT was cleared updating 526 keys and creating 29 modified pages.
11:23:11.000 PM   host = DESKTOP-JS7AMJO | source = syslogs.csv | sourcetype = log2metrics_csv
```

So, we can conclude that the Malware elevated privileges using `SYSTEM` and cleaned traces.

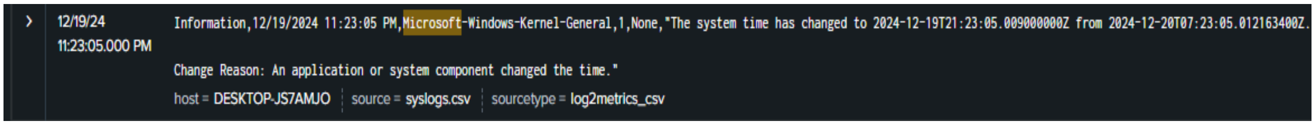
3) Perform this SPL Query to retrieve all events related to security with event ID 4732 related to adding account to security enabled group (administrators) and notice this event timed at 11:23:05 PM
(index="seclog" 4732 | sort _time)



```
index="seclog" 4732 | sort _time
```

```
12/19/24
11:23:05.000 PM
Information,12/19/2024 11:23:05 PM,Microsoft-Windows-Security-Auditing,4732,Security Group Management,"A member was added to a security-enabled local group.

Subject:
        Security ID:            SYSTEM
        Account Name:           WIN-4U2S9Q7EEP6$
Show all 19 lines
host = DESKTOP-JS7AMJO | source = seclogs.csv | sourcetype = log2metrics_csv

12/19/24
11:23:05.000 PM
Information,12/19/2024 11:23:05 PM,Microsoft-Windows-Security-Auditing,4732,Security Group Management,"A member was added to a security-enabled local group.

Subject:
        Security ID:            SYSTEM
        Account Name:           WIN-4U2S9Q7EEP6$
        Account Domain:         WORKGROUP
        Logon ID:               0x3E7

Member:
        Security ID:            S-1-5-21-3385474721-3775886027-2527884829-1001
        Account Name:           -

Group:
        Security ID:            BUILTIN\Administrators
        Group Name:             Administrators
        Group Domain:           Builtin
```

The next step is to analyze the system logs in the same timing that we got from the security log that we identified earlier using this SPL Query, and notice this specific Log



```
12/19/24
11:23:05.000 PM
Information,12/19/2024 11:23:05 PM,Microsoft-Windows-Kernel-General,1,None,"The system time has changed to 2024-12-19T21:23:05.009000000Z from 2024-12-20T07:23:05.012163400Z.

Change Reason: An application or system component changed the time."
host = DESKTOP-JS7AMJO | source = syslogs.csv | sourcetype = log2metrics_csv
```

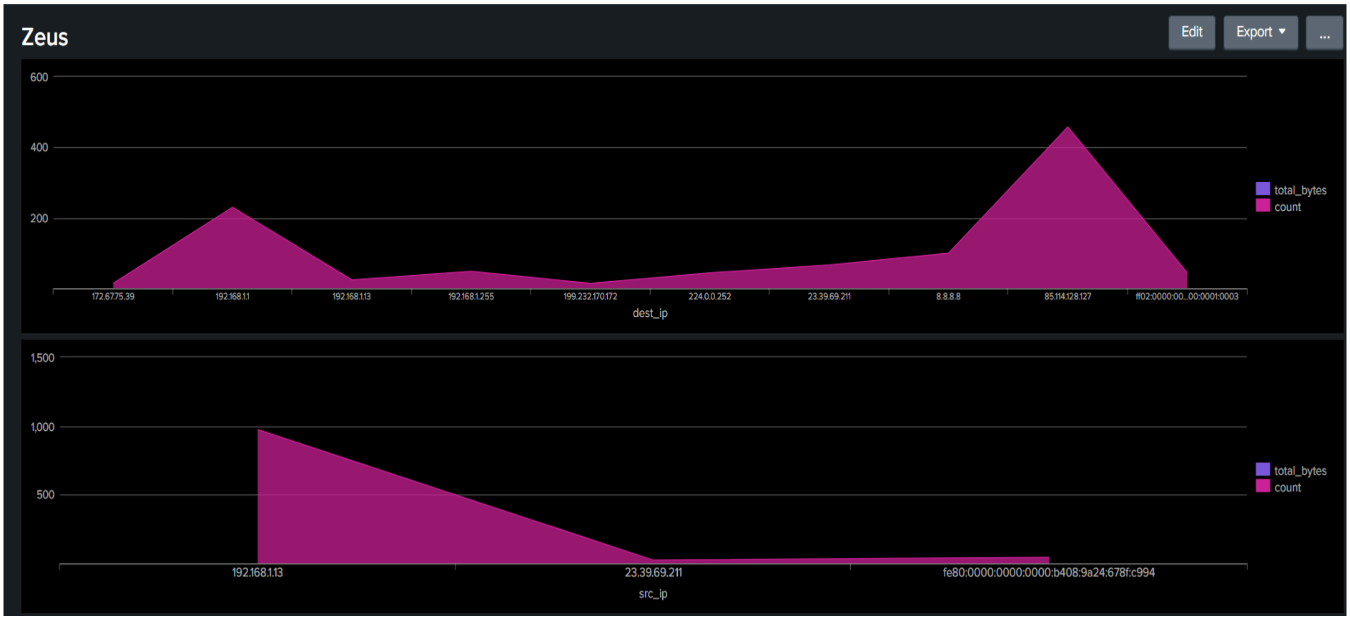
So, we can conclude that the Malware changed the system time to an earlier timestamp to Manipulate logs, evade detection, and alter security controls.

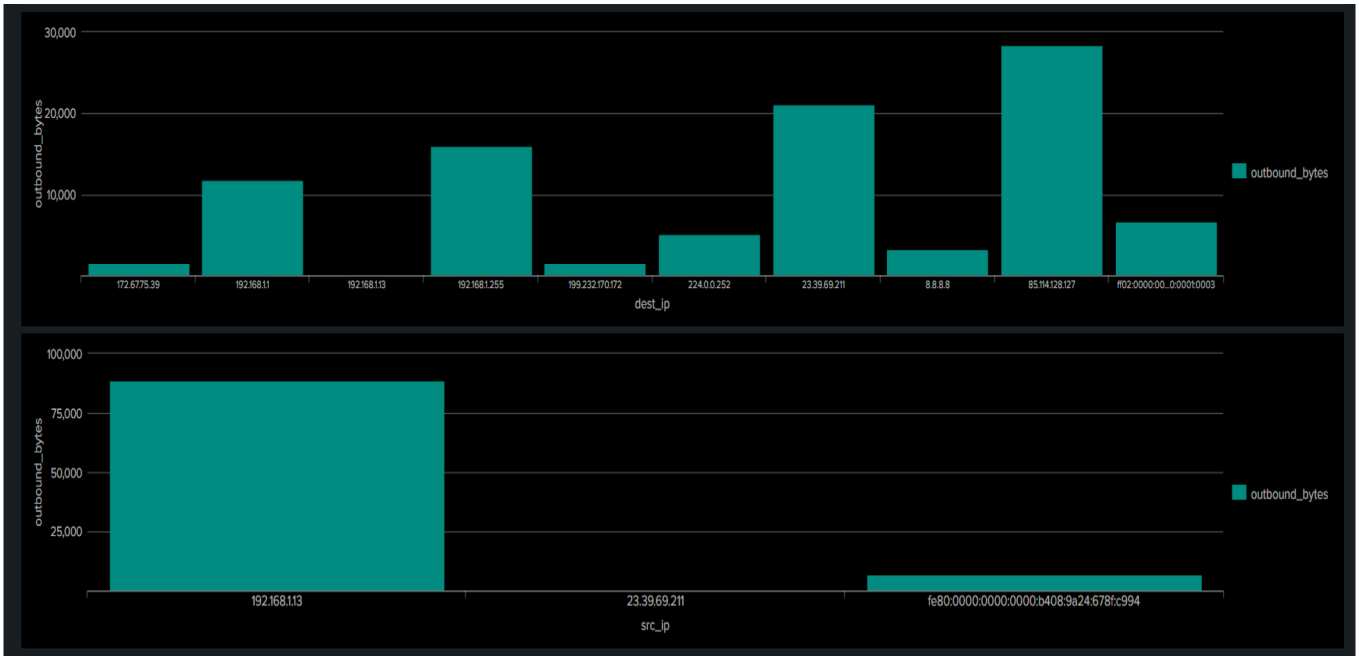
The Last Step is to create Visual Dashboard to track Malicious Activity

The dashboard was built over these 4 SPL Queries:

1) index="zeuslogs" | top dest_ip : Top Destination IPs based on events count.
2) index="zeuslogs" | top src_ip : Top Source IPs based on events count.
3) index="zeuslogs" | stats sum(flow.bytes_toserver) as outbound_bytes by src_ip : Calculates the total amount of outbound data sent to servers by each source IPs
4) index="zeuslogs" | stats sum(flow.bytes_toserver) as outbound_bytes by dest_ip : Calculates the total amount of outbound data sent to each destination IP.

The Dashboard



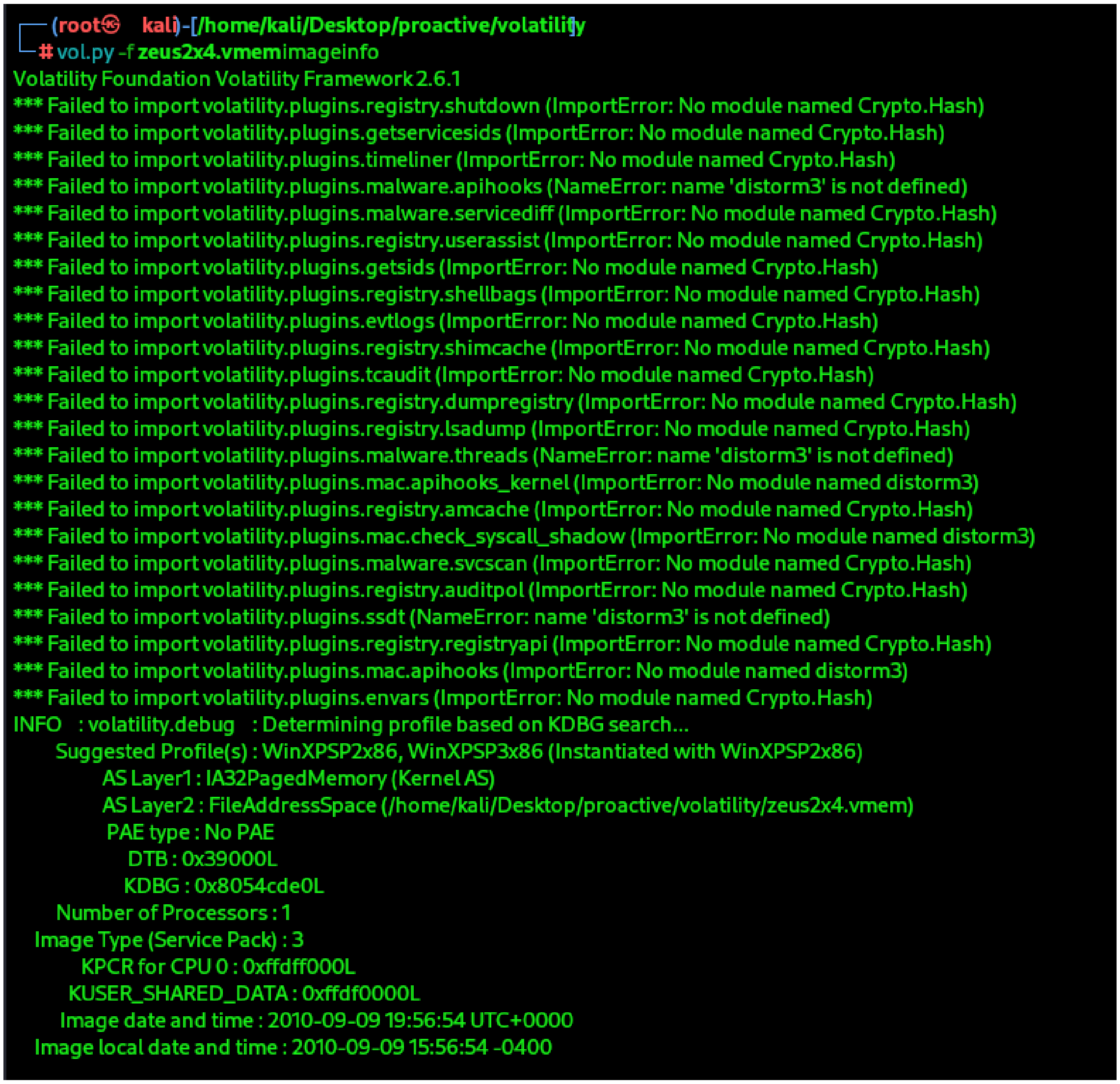


Analyzing The Zeus Banking Trojan with Volatility

This report aims to analyze a memory dump of a potentially compromised system using the Volatility 2 Framework to identify active and injected processes related to Zeus and investigate associated network connections. I will utilize the volatility plugins to study the processes, memory strings, memory code injection, and network connections.

Identifying the system

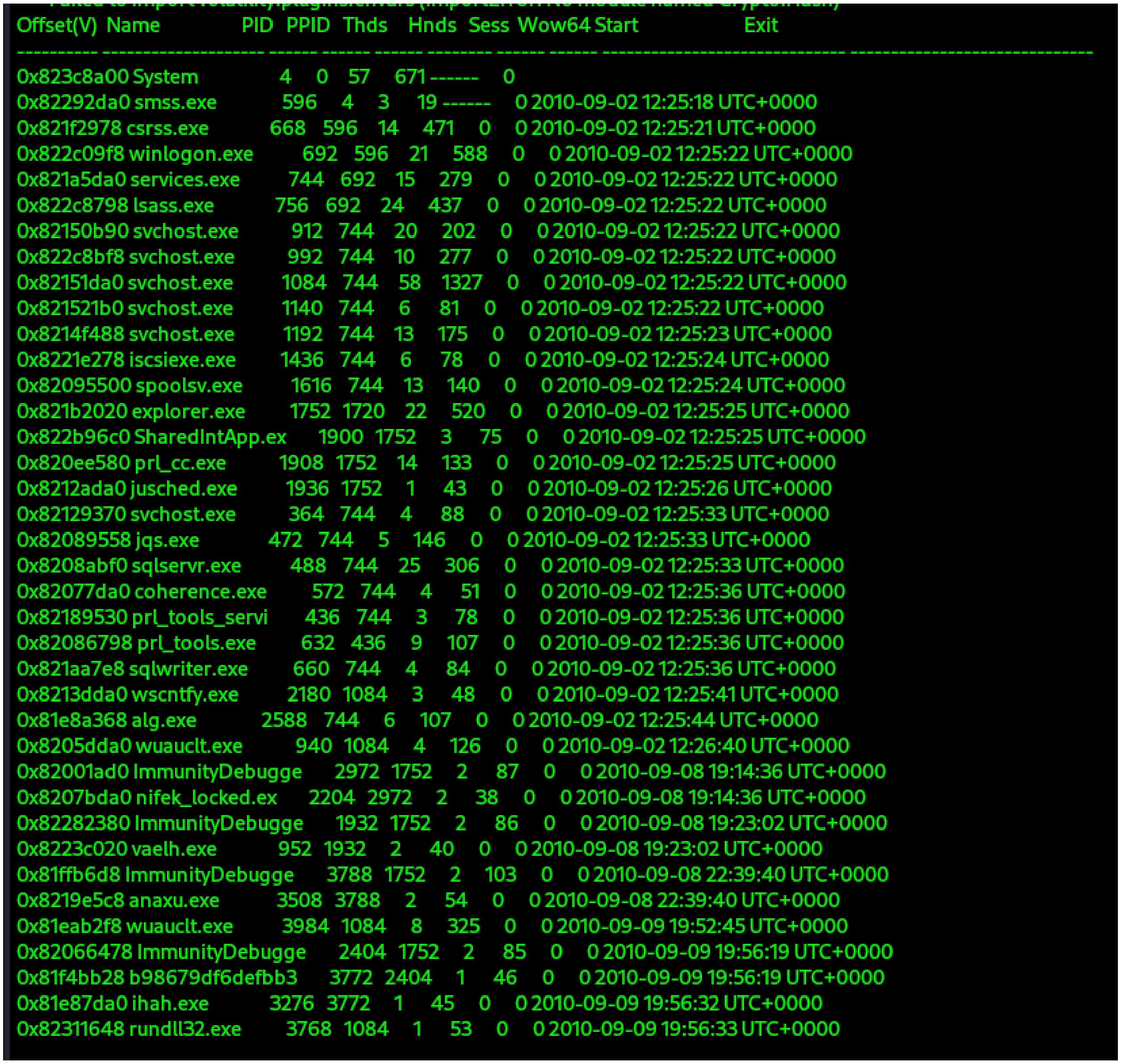
First, we analyze the image information to know what we're dealing with using the imageinfo module which reveals it's a windows XP memory dump



```
┌──(root㉿kali)-[/home/kali/Desktop/proactive/volatility]
└─# vol.py -f zeus2x4.vmem imageinfo
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.registry.shutdown (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getservicesids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.timeliner (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.servicediff (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.userassist (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getsids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shellbags (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.evtlogs (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shimcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.tcaudit (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.dumpregistry (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.registry.amcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.malware.svcscan (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.auditpol (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.ssdt (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.registry.registryapi (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.envars (ImportError: No module named Crypto.Hash)
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/kali/Desktop/proactive/volatility/zeus2x4.vmem)
                      PAE type : No PAE
                           DTB : 0x39000L
                          KDBG : 0x8054cde0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2010-09-09 19:56:54 UTC+0000
     Image local date and time : 2010-09-09 15:56:54 -0400
```

Enumerating Processes

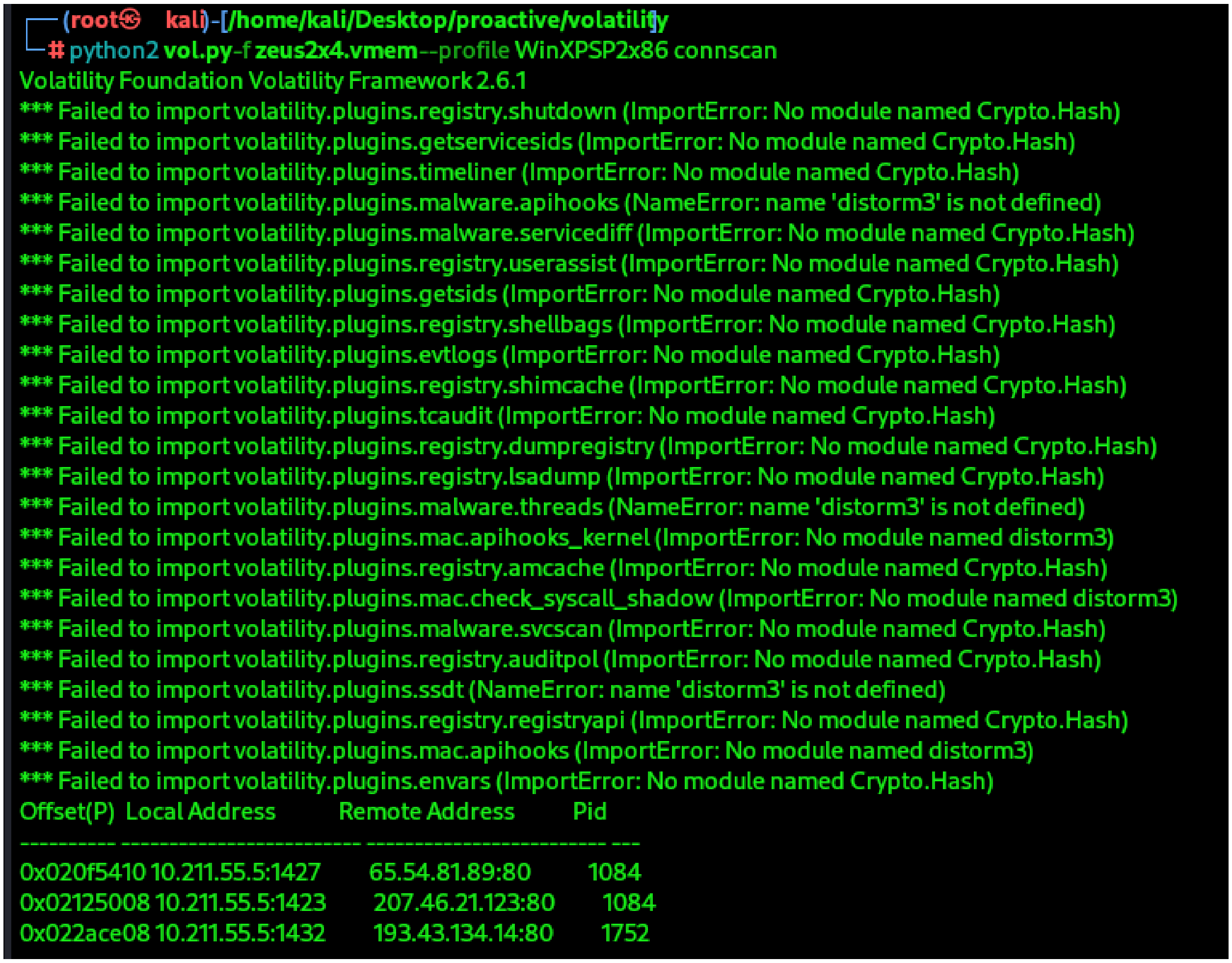
- looking at the processes using **python2 vol.py -f zeus2x4.vmem --profile WinXPSP2x86 pslist.** We don’t see anything suspicious from the processes listed here as the count and names of the processes look fine.



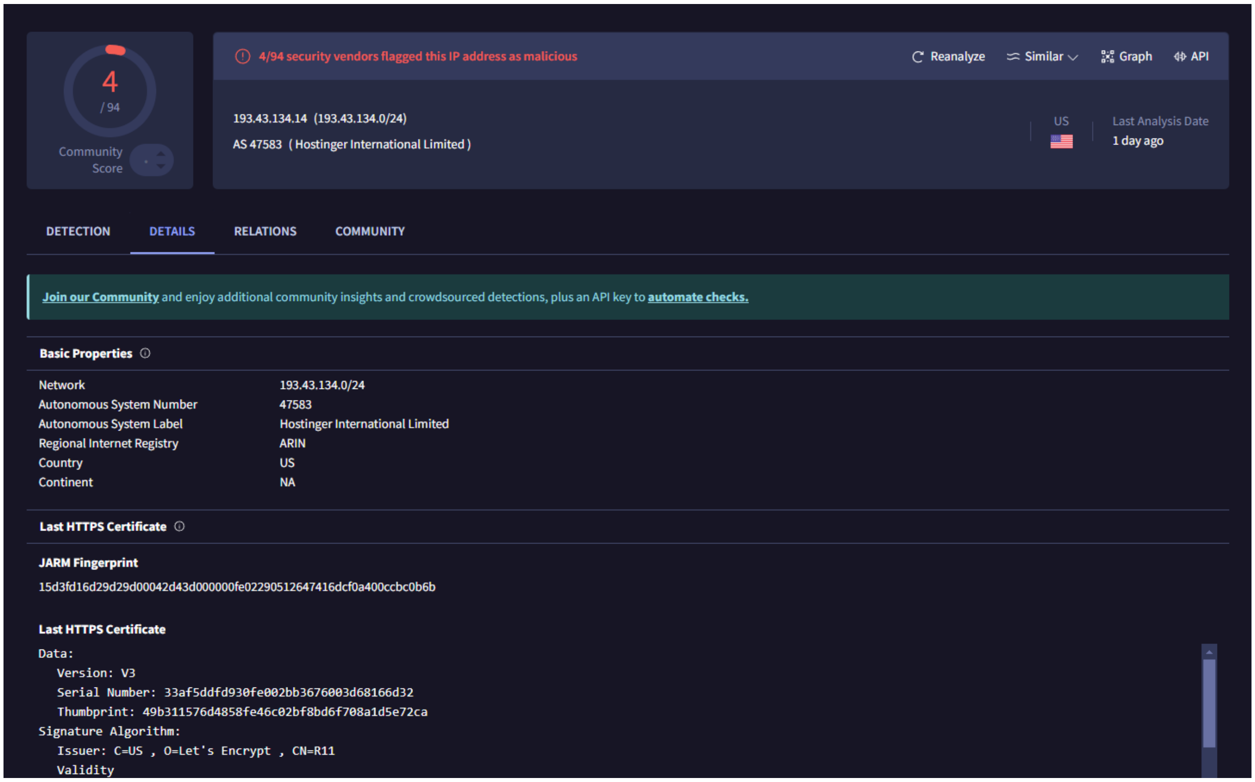
```
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x823c8a00 System                    4      0     57      671 ------      0
0x82292da0 smss.exe                596      4      3       19 ------      0 2010-09-02 12:25:18 UTC+0000
0x821f2978 csrss.exe               668    596     14      471      0      0 2010-09-02 12:25:21 UTC+0000
0x822c09f8 winlogon.exe            692    596     21      588      0      0 2010-09-02 12:25:22 UTC+0000
0x821a5da0 services.exe            744    692     15      279      0      0 2010-09-02 12:25:22 UTC+0000
0x822c8798 lsass.exe               756    692     24      437      0      0 2010-09-02 12:25:22 UTC+0000
0x82150b90 svchost.exe             912    744     20      202      0      0 2010-09-02 12:25:22 UTC+0000
0x822c8bf8 svchost.exe             992    744     10      277      0      0 2010-09-02 12:25:22 UTC+0000
0x82151da0 svchost.exe            1084    744     58     1327      0      0 2010-09-02 12:25:22 UTC+0000
0x821521b0 svchost.exe            1140    744      6       81      0      0 2010-09-02 12:25:22 UTC+0000
0x8214f488 svchost.exe            1192    744     13      175      0      0 2010-09-02 12:25:23 UTC+0000
0x8221e278 iscsiexe.exe           1436    744      6       78      0      0 2010-09-02 12:25:24 UTC+0000
0x82095500 spoolsv.exe            1616    744     13      140      0      0 2010-09-02 12:25:24 UTC+0000
0x821b2020 explorer.exe           1752   1720     22      520      0      0 2010-09-02 12:25:25 UTC+0000
0x822b96c0 SharedIntApp.ex        1900   1752      3       75      0      0 2010-09-02 12:25:25 UTC+0000
0x820ee580 prl_cc.exe             1908   1752     14      133      0      0 2010-09-02 12:25:25 UTC+0000
0x8212ada0 jusched.exe            1936   1752      1       43      0      0 2010-09-02 12:25:26 UTC+0000
0x82129370 svchost.exe             364    744      4       88      0      0 2010-09-02 12:25:33 UTC+0000
0x82089558 jqs.exe                 472    744      5      146      0      0 2010-09-02 12:25:33 UTC+0000
0x8208abf0 sqlservr.exe            488    744     25      306      0      0 2010-09-02 12:25:33 UTC+0000
0x82077da0 coherence.exe           572    744      4       51      0      0 2010-09-02 12:25:36 UTC+0000
0x82189530 prl_tools_servi         436    744      3       78      0      0 2010-09-02 12:25:36 UTC+0000
0x82086798 prl_tools.exe           632    436      9      107      0      0 2010-09-02 12:25:36 UTC+0000
0x821aa7e8 sqlwriter.exe           660    744      4       84      0      0 2010-09-02 12:25:36 UTC+0000
0x8213dda0 wscntfy.exe            2180   1084      3       48      0      0 2010-09-02 12:25:41 UTC+0000
0x81e8a368 alg.exe                2588    744      6      107      0      0 2010-09-02 12:25:44 UTC+0000
0x8205dda0 wuauclt.exe             940   1084      4      126      0      0 2010-09-02 12:26:40 UTC+0000
0x82001ad0 ImmunityDebugge        2972   1752      2       87      0      0 2010-09-08 19:14:36 UTC+0000
0x8207bda0 nifek_locked.ex        2204   2972      2       38      0      0 2010-09-08 19:14:36 UTC+0000
0x82282380 ImmunityDebugge        1932   1752      2       86      0      0 2010-09-08 19:23:02 UTC+0000
0x8223c020 vaelh.exe               952   1932      2       40      0      0 2010-09-08 19:23:02 UTC+0000
0x81ffb6d8 ImmunityDebugge        3788   1752      2      103      0      0 2010-09-08 22:39:40 UTC+0000
0x8219e5c8 anaxu.exe              3508   3788      2       54      0      0 2010-09-08 22:39:40 UTC+0000
0x81eab2f8 wuauclt.exe            3984   1084      8      325      0      0 2010-09-09 19:52:45 UTC+0000
0x82066478 ImmunityDebugge        2404   1752      2       85      0      0 2010-09-09 19:56:19 UTC+0000
0x81f4bb28 b98679df6defbb3        3772   2404      1       46      0      0 2010-09-09 19:56:19 UTC+0000
0x81e87da0 ihah.exe               3276   3772      1       45      0      0 2010-09-09 19:56:32 UTC+0000
0x82311648 rundll32.exe           3768   1084      1       53      0      0 2010-09-09 19:56:33 UTC+0000
```

Network connections

When analyzing the network connections using connscan where we get 3 different IP addresses. Scanning each one resulted in only one suspicious IP address 193.43.134.14 hooked to a process with ID 1752.



```
┌──(root㉿kali)-[/home/kali/Desktop/proactive/volatility]
└─# python2 vol.py -f zeus2x4.vmem --profile WinXPSP2x86 connscan
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.registry.shutdown (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getservicesids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.timeliner (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.servicediff (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.userassist (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getsids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shellbags (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.evtlogs (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shimcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.tcaudit (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.dumpregistry (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.registry.amcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.malware.svcscan (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.auditpol (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.ssdt (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.registry.registryapi (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.envars (ImportError: No module named Crypto.Hash)
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ---
0x020f5410 10.211.55.5:1427          65.54.81.89:80            1084
0x02125008 10.211.55.5:1423          207.46.21.123:80          1084
0x022ace08 10.211.55.5:1432          193.43.134.14:80          1752
```



4/94 security vendors flagged this IP address as malicious

193.43.134.14 (193.43.134.0/24)
AS 47583 (Hostinger International Limited)

Last Analysis Date
1 day ago

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Basic Properties

Network	193.43.134.0/24
Autonomous System Number	47583
Autonomous System Label	Hostinger International Limited
Regional Internet Registry	ARIN
Country	US
Continent	NA

Last HTTPS Certificate

JARM Fingerprint

15d3fd16d29d29d00042d43d000000fe02290512647416dcf0a400ccbc0b6b

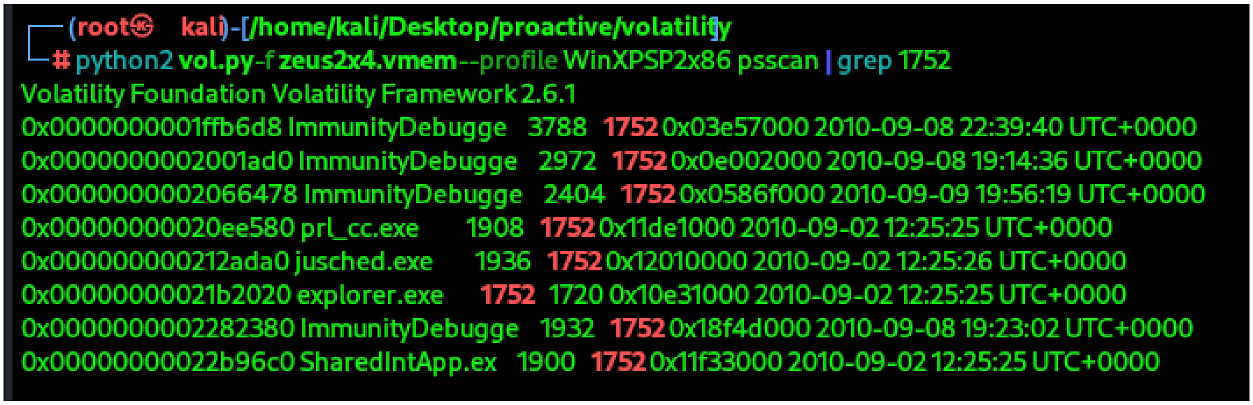
Last HTTPS Certificate

```
Data:
   Version: V3
   Serial Number: 33af5ddfd930fe002bb3676003d68166d32
   Thumbprint: 49b311576d4858fe46c02bf8bd6f708a1d5e72ca
Signature Algorithm:
   Issuer: C=US , O=Let's Encrypt , CN=R11
   Validity
```

The IP shows malicious activity on virustotal.

Malicious Process

Now to check which process was communicating with this IP address by grepping the output of psscan. We can see that the process that made the suspicious network connection is “explorer.exe” of PID1752.

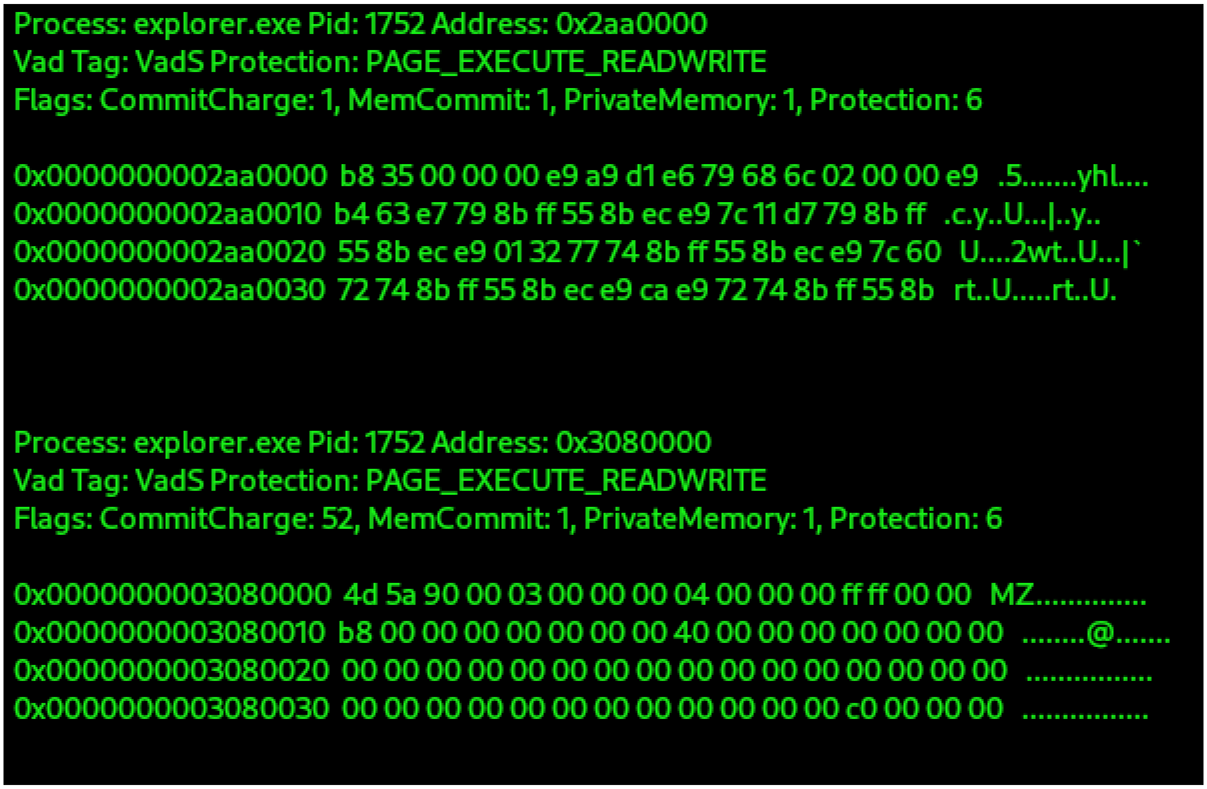


```
┌──(root㉿kali)-[/home/kali/Desktop/proactive/volatility]
└─# python2 vol.py -f zeus2x4.vmem --profile WinXPSP2x86 psscan | grep 1752
Volatility Foundation Volatility Framework 2.6.1
0x000000001ffb6d8 ImmunityDebugge   3788  1752 0x03e57000 2010-09-08 22:39:40 UTC+0000
0x0000000002001ad0 ImmunityDebugge   2972  1752 0x0e002000 2010-09-08 19:14:36 UTC+0000
0x0000000002066478 ImmunityDebugge   2404  1752 0x0586f000 2010-09-09 19:56:19 UTC+0000
0x00000000020ee580 prl_cc.exe        1908  1752 0x11de1000 2010-09-02 12:25:25 UTC+0000
0x000000000212ada0 jusched.exe       1936  1752 0x12010000 2010-09-02 12:25:26 UTC+0000
0x00000000021b2020 explorer.exe      1752  1720 0x10e31000 2010-09-02 12:25:25 UTC+0000
0x0000000002282380 ImmunityDebugge   1932  1752 0x18f4d000 2010-09-08 19:23:02 UTC+0000
0x00000000022b96c0 SharedIntApp.ex   1900  1752 0x11f33000 2010-09-02 12:25:25 UTC+0000
```

Code injection

Nothing about the process is suspicious. The code might be injected into the process. We'll use malfind plugin to check.

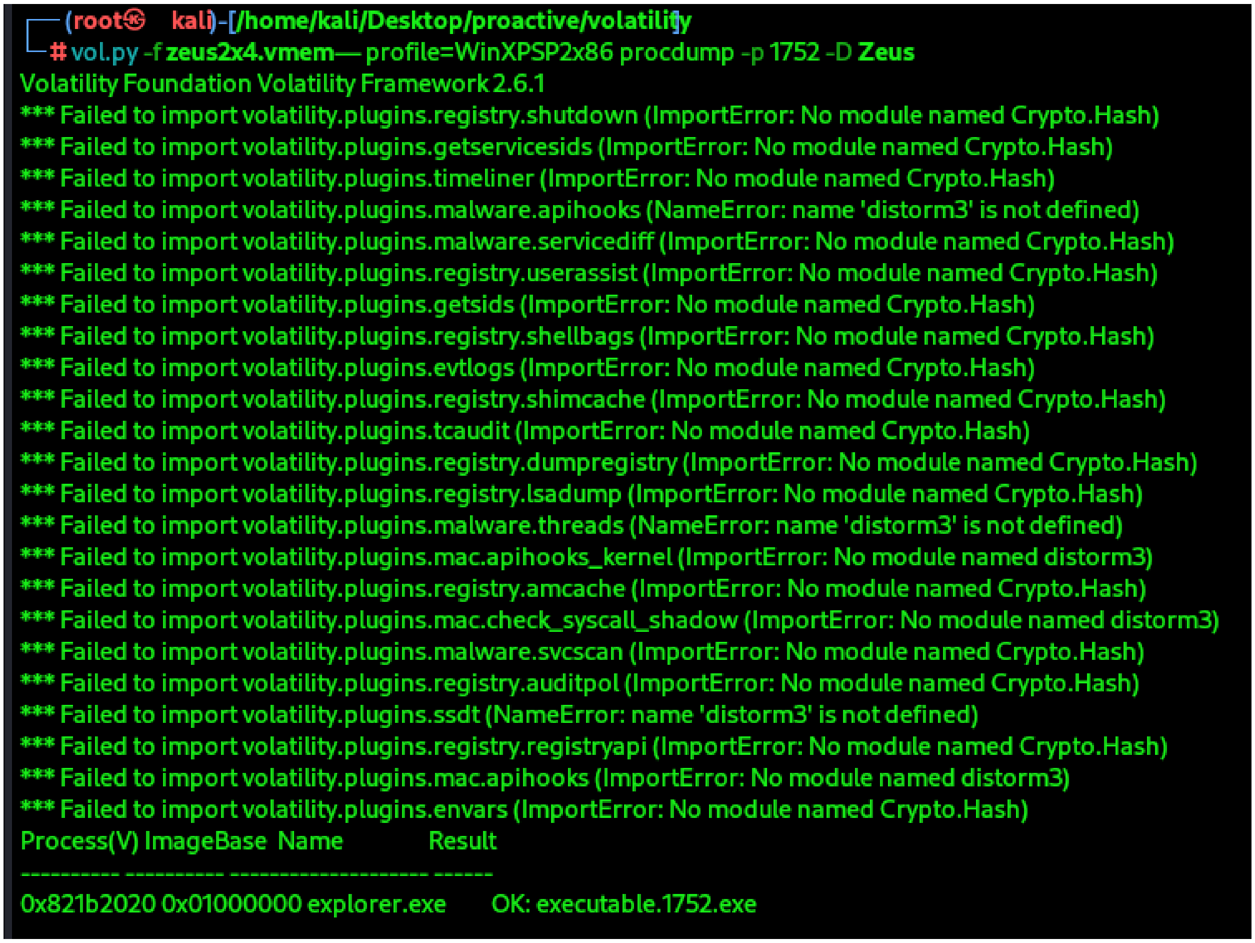
Using **python2 vol.py -f zeus2x4.vmem —profile WinXPSP2x86 malfind -p 1752**



```
Process: explorer.exe Pid: 1752 Address: 0x2aa0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x0000000002aa0000  b8 35 00 00 00 e9 a9 d1 e6 79 68 6c 02 00 00 e9   .5.......yhl....
0x0000000002aa0010  b4 63 e7 79 8b ff 55 8b ec e9 7c 11 d7 79 8b ff   .c.y..U...|..y..
0x0000000002aa0020  55 8b ec e9 01 32 77 74 8b ff 55 8b ec e9 7c 60   U....2wt..U...|`
0x0000000002aa0030  72 74 8b ff 55 8b ec e9 ca e9 72 74 8b ff 55 8b   rt..U.....rt..U.



Process: explorer.exe Pid: 1752 Address: 0x3080000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 52, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x0000000003080000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x0000000003080010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x0000000003080020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000000003080030  00 00 00 00 00 00 00 00 00 00 00 00 c0 00 00 00   ................
```

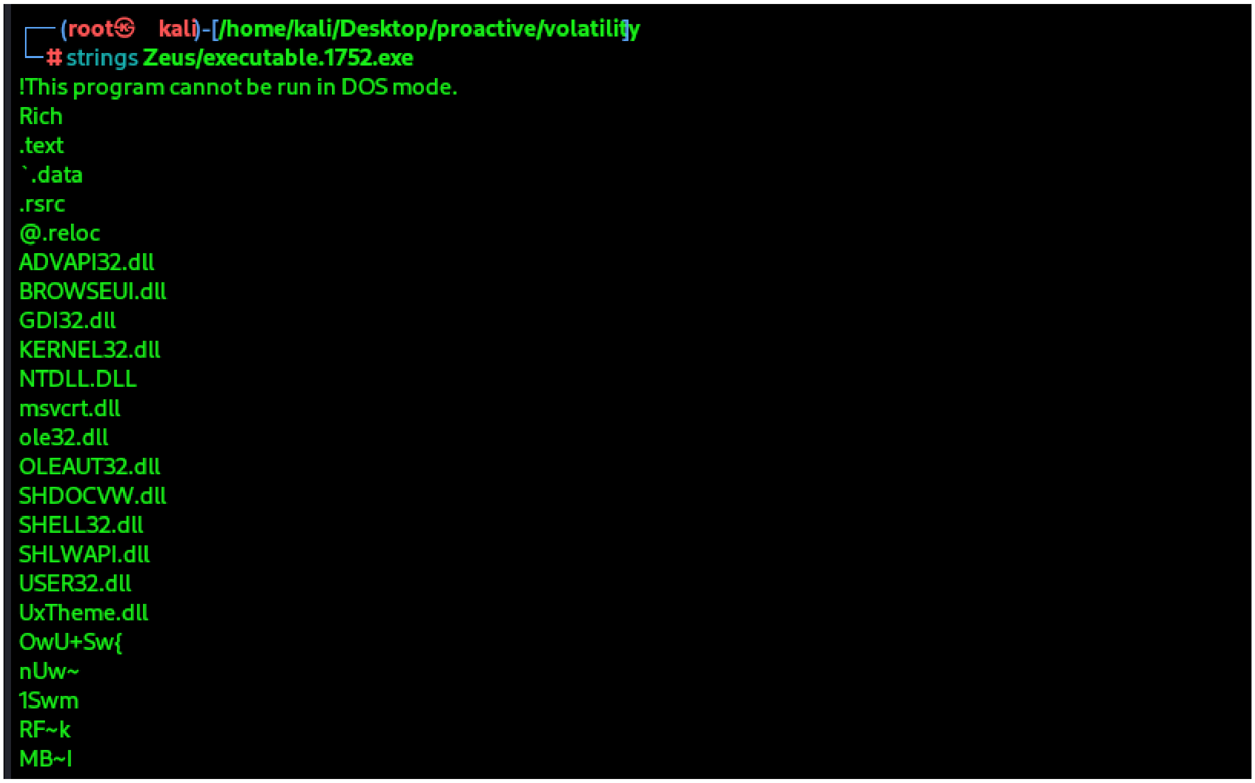
By looking at the result of the explorer.exe online it shows that this process has MZ header and protection of PAGE_EXECUTE_READWRITE, which means that this memory region is marked as executable, and it can also be both read from and written to. Memory regions shouldn't be executable and writable at the same time.

We’ll try dumping this process information

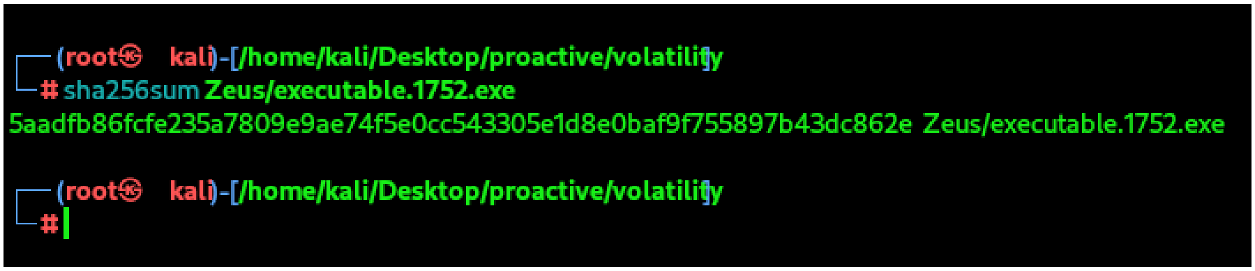


```
┌──(root㉿ kali)-[/home/kali/Desktop/proactive/volatility]
└─# vol.py -f zeus2x4.vmem— profile=WinXPSP2x86 procdump -p 1752 -D Zeus
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.registry.shutdown (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getservicesids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.timeliner (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.servicediff (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.userassist (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getsids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shellbags (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.evtlogs (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shimcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.tcaudit (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.dumpregistry (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.registry.amcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.malware.svcscan (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.auditpol (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.ssdt (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.registry.registryapi (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.envars (ImportError: No module named Crypto.Hash)
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x821b2020 0x01000000 explorer.exe         OK: executable.1752.exe
```

We can get the strings and by getting the sha256 checksum we can look it up on virustotal as follows:

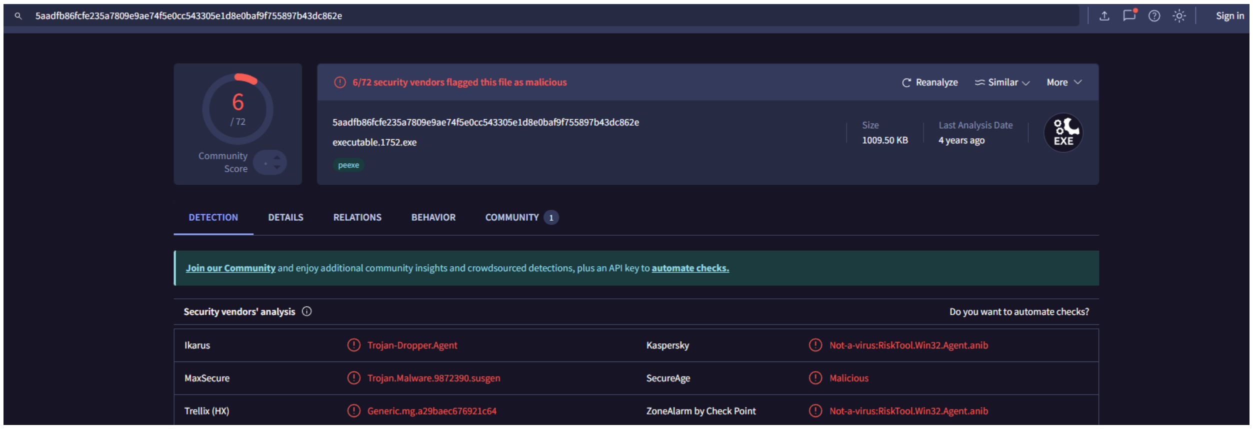


```
┌──(root㉿kali)-[/home/kali/Desktop/proactive/volatility]
└─# strings Zeus/executable.1752.exe
!This program cannot be run in DOS mode.
Rich
.text
`.data
.rsrc
@.reloc
ADVAPI32.dll
BROWSEUI.dll
GDI32.dll
KERNEL32.dll
NTDLL.DLL
msvcrt.dll
ole32.dll
OLEAUT32.dll
SHDOCVW.dll
SHELL32.dll
SHLWAPI.dll
USER32.dll
UxTheme.dll
OwU+Sw{
nUw~
1Swm
RF~k
MB~I
```



```
┌──(root㉿kali)-[/home/kali/Desktop/proactive/volatility]
└─# sha256sum Zeus/executable.1752.exe
5aadfb86fcfe235a7809e9ae74f5e0cc543305e1d8e0baf9f755897b43dc862e  Zeus/executable.1752.exe

┌──(root㉿kali)-[/home/kali/Desktop/proactive/volatility]
└─# 
```

We can check the hash on virustotal and see that it actually is malicious.



5aadfb86fcfe235a7809e9ae74f5e0cc543305e1d8e0baf9f755897b43dc862e

6/72 security vendors flagged this file as malicious

5aadfb86fcfe235a7809e9ae74f5e0cc543305e1d8e0baf9f755897b43dc862e

executable.1752.exe

Size
1009.50 KB

Last Analysis Date
4 years ago

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Security vendors' analysis

Do you want to automate checks?

Ikarus	Trojan-Dropper.Agent	Kaspersky	Not-a-virus:RiskTool.Win32.Agent.anib
MaxSecure	Trojan.Malware.9872390.susgen	SecureAge	Malicious
Trellix (HX)	Generic.mg.a29baec676921c64	ZoneAlarm by Check Point	Not-a-virus:RiskTool.Win32.Agent.anib

Conclusion:

- This memory dump is infected contains a malware
- Initial connection to the C2 server was made to the IP 193.43.134.14
- Malware is hooked to the explorer.exe process with ID 1752

Zeus Banking Trojan Detection With YARA

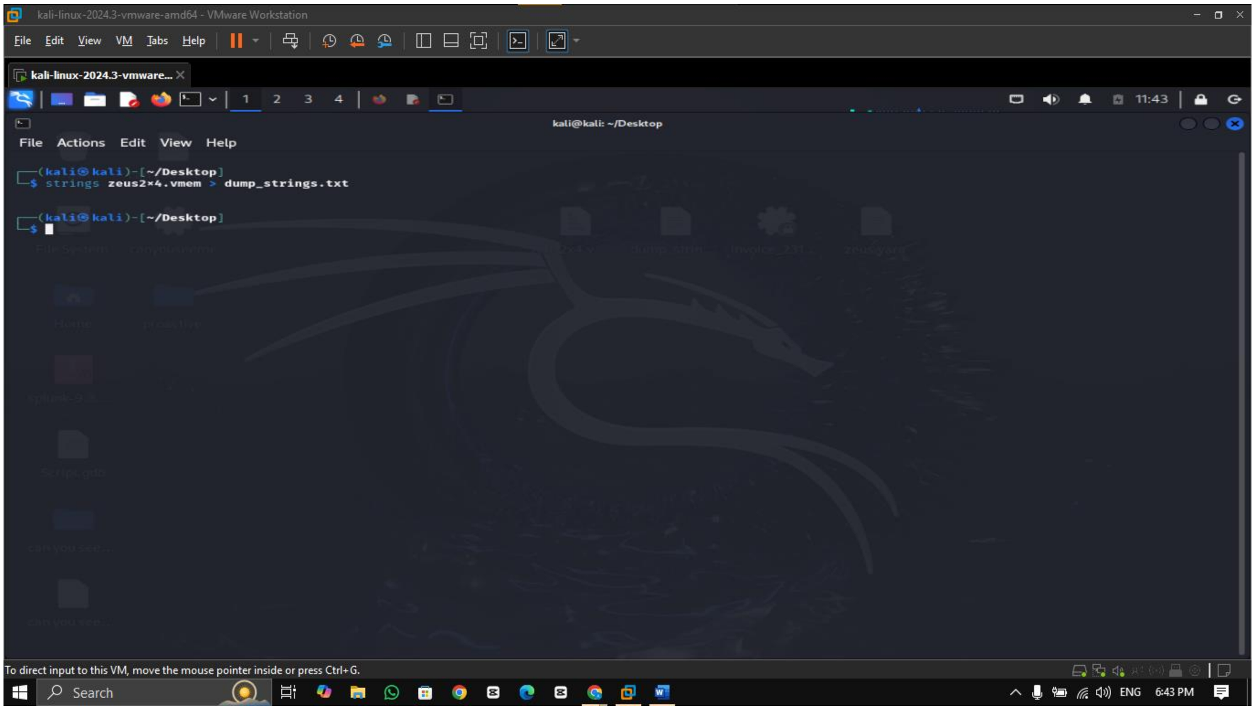
- **Objective:**

  Detect Zeus with YARA Signatures:

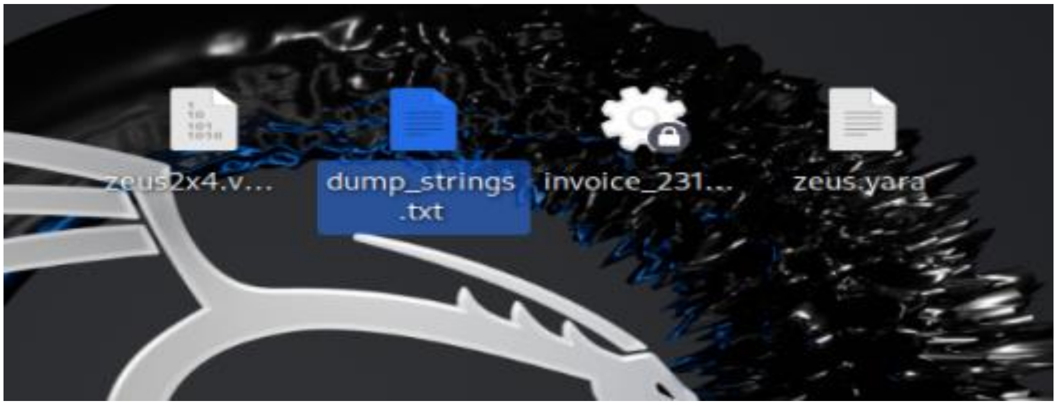
  - Write custom YARA rules to detect
    Zeus-related patterns in binaries, configuration files, and memory dumps.
  - Scan the infected system and memory dumps
    with YARA to identify Zeus artifacts.

- **Steps:**

1. Using Strings to Extract Data from the memory dump file:


kali-linux-2024.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
kali-linux-2024.3-vmware...
kali@kali: ~/Desktop
File Actions Edit View Help
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Search

```
┌──(kali㉿kali)-[~/Desktop]
└─$ strings zeus2x4.vmem > dump_strings.txt

┌──(kali㉿kali)-[~/Desktop]
└─$ 
```



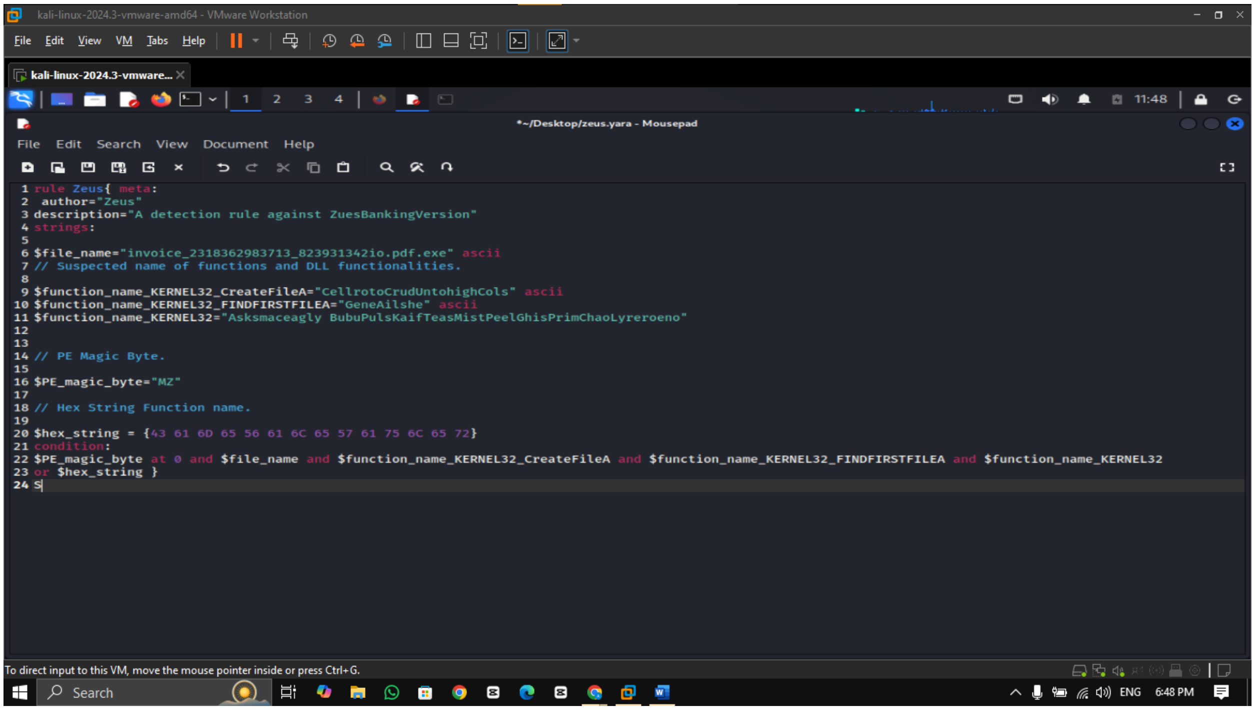
2. Searching for Suspected Common Strings:

Once the strings are extracted, the next step is to search for suspected common strings or patterns that could indicate suspicious activity. This involves looking for signatures, keywords, or patterns that are indicative of known malicious behavior or functions.

3. Creating YARA Rules Based on Suspected Functions Calls, Magic Bytes, and Strings:

After identifying suspected strings or patterns, the next step is to create YARA rules to automate the detection of these suspicious behaviors. YARA rules can be based on the following elements:

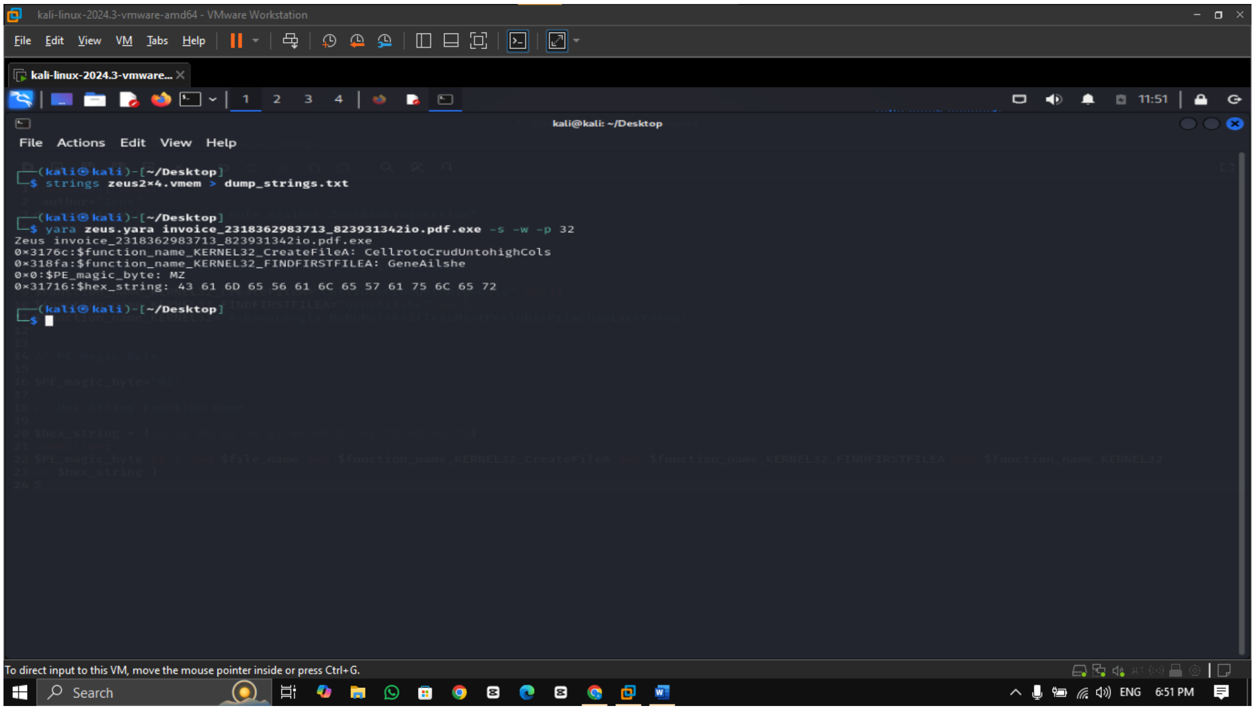
- **Function calls** – If suspicious function calls were detected in the extracted strings, YARA rules can be created to match these function names.
- **Magic bytes** – Specific byte sequences that are known to indicate file formats or data structures associated with malware.
- **Strings** – Custom strings that match specific patterns found in the extracted strings.


kali-linux-2024.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
kali-linux-2024.3-vmware...
*~/Desktop/zeus.yara - Mousepad
File Edit Search View Document Help
11:48
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Search
ENG 6:48 PM

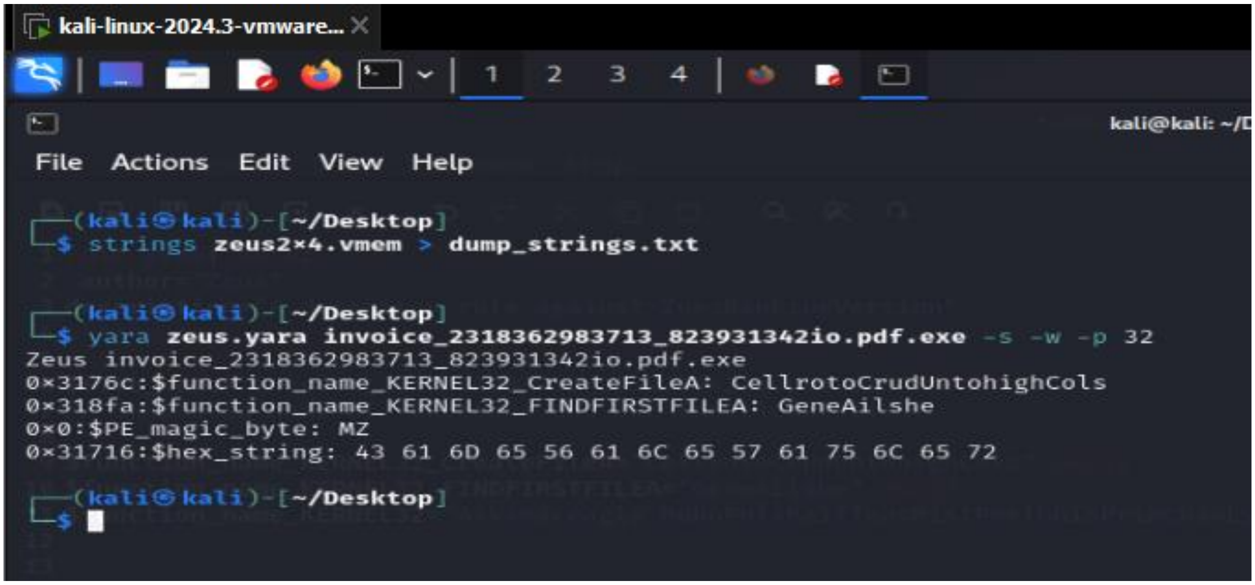
```
rule Zeus{ meta:
 author="Zeus"
description="A detection rule against ZuesBankingVersion"
strings:

$file_name="invoice_2318362983713_823931342io.pdf.exe" ascii
// Suspected name of functions and DLL functionalities.

$function_name_KERNEL32_CreateFileA="CellrotoCrudUntohighCols" ascii
$function_name_KERNEL32_FINDFIRSTFILEA="GeneAilshe" ascii
$function_name_KERNEL32="Asksmaceagly BubuPulsKaifTeasMistPeelGhisPrimChaoLyreroeno"


// PE Magic Byte.

$PE_magic_byte="MZ"

// Hex String Function name.

$hex_string = {43 61 6D 65 56 61 6C 65 57 61 75 6C 65 72}
condition:
$PE_magic_byte at 0 and $file_name and $function_name_KERNEL32_CreateFileA and $function_name_KERNEL32_FINDFIRSTFILEA and $function_name_KERNEL32
or $hex_string }
S
```

4. Running YARA on the malicious file:

The final step is to run the created YARA rules on the malicious file to check for any matches that would indicate the presence of malicious content. YARA will scan the file, using the created rules, and report any potential indicators of compromise.


kali-linux-2024.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
kali-linux-2024.3-vmware...
1 2 3 4
11:51
kali@kali: ~/Desktop
File Actions Edit View Help
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Search
ENG 6:51 PM

```
┌──(kali㉿kali)-[~/Desktop]
└─$ strings zeus2x4.vmem > dump_strings.txt

┌──(kali㉿kali)-[~/Desktop]
└─$ yara zeus.yara invoice_2318362983713_823931342io.pdf.exe -s -w -p 32
Zeus invoice_2318362983713_823931342io.pdf.exe
0x3176c:$function_name_KERNEL32_CreateFileA: CellrotoCrudUntohighCols
0x318fa:$function_name_KERNEL32_FINDFIRSTFILEA: GeneAilshe
0x0:$PE_magic_byte: MZ
0x31716:$hex_string: 43 61 6D 65 56 61 6C 65 57 61 75 6C 65 72

┌──(kali㉿kali)-[~/Desktop]
└─$
```



```
┌──(kali㉿kali)-[~/Desktop]
└─$ strings zeus2x4.vmem > dump_strings.txt

┌──(kali㉿kali)-[~/Desktop]
└─$ yara zeus.yara invoice_2318362983713_823931342io.pdf.exe -s -w -p 32
Zeus invoice_2318362983713_823931342io.pdf.exe
0x3176c:$function_name_KERNEL32_CreateFileA: CellrotoCrudUntohighCols
0x318fa:$function_name_KERNEL32_FINDFIRSTFILEA: GeneAilshe
0x0:$PE_magic_byte: MZ
0x31716:$hex_string: 43 61 6D 65 56 61 6C 65 57 61 75 6C 65 72

┌──(kali㉿kali)-[~/Desktop]
└─$
```

- Command Flag:

-s: Displays matching strings.
-w: Enables warnings (useful for debugging rules).
-p 32: Sets the maximum process recursion depth to 32. This is helpful when scanning complex binaries.

5. Output explanation:

Match 1:

$function_name_KERNEL32_CreateFileA: The name of the matched string in your YARA rule. In this case, it indicates a function call to CreateFileA from the Windows KERNEL32.dll library.

- **CreateFileA**: This function is often used in malware to create, open, or manipulate files.
- **CellrotoCrudUntohighCols**: This is the actual value or string found at the offset, potentially used in the malicious operation.

Match 2:

$function_name_KERNEL32_FINDFIRSTFILEA: This appears to refer to the name of a Windows API function. Specifically, **FindFirstFileA** is a function in the KERNEL32.dll library used to find the first file in a directory that matches a given pattern. The "A" at the end typically refers to the ANSI version of the function (as opposed to **FindFirstFileW**, which would be the wide-character version).

Match 3:

$PE_magic_byte: A string defined in your YARA rule, representing the PE (Portable Executable) file signature.

- **MZ:** The magic bytes of a Windows executable, confirming that this file is a PE binary.

Match 4:

$hex_string: The name of the string in the YARA rule, which matches the hex values in the file.

- **43 61 6D 65 56 61 6C 65 57 61 75 6C 65 72**: Hexadecimal representation of the ASCII string CameValeWaule.
- This could be an encoded or obfuscated string used in the malware.