[1] Syed Shameem*

[2] Kalisetty Venkatesh

[3] Latif Shaik

[4] Medavarapu T N D Sri Harsha

[5] BailundoLuis Rablay Lopes

Estimating Malware Impact on Network Traffic Analysis by Using Wireshark

Abstract: - With the increasing prevalence of advanced cyber threats, there is a growing need for effective cybersecurity measures that can detect and visualize malware attacks. This research introduces an integrated approach that combines malware detection techniques with geospatial visualization methods to enhance the identification and analysis of cyber-attacks. By analyzing packets in the HTTP protocol, we identify suspicious file transfers and compute their hash values for further examination. To assess the threat level of these transferred files, we utilize the Virus Total platform to conduct comprehensive scans for malware. At the same time, by utilizing geolocation data, we map out both the origins and destinations of these attacks, providing valuable spatial context for understanding global patterns in cyber threats. In addition to enabling the identification of potentially harmful files, the proposed approach also provides a comprehensive visualization of how these threats are spread geographically. Our findings contribute to advancing cybersecurity strategies by facilitating proactive threat mitigation and enhancing incident response capabilities. The integration of malware detection, hash analysis, and geospatial visualization emphasizes the importance of adopting a multidimensional approach in strengthening network security infrastructure.

Keywords: Malware Detection, Geospatial Visualization, HTTP Protocol, Virus Total, Network Traffic Analysis, Cybersecurity.

I. Introduction

The ever-changing landscape of cybersecurity presents a continuous challenge that requires constant innovation and adaptability. Among these challenges, malware attacks have become a significant threat to network security. These attacks combine sophistication with widespread occurrence, compromising the digital defenses organizations implement to safeguard their sensitive data, infrastructure, and digital resources. Malware detection methods [1] have traditionally relied on signature scanning, anomaly detection, and sandboxing. However, due to the constant evolution of malware tactics, there is a need for more comprehensive approaches that consider both technical characteristics and geographical aspects. Understanding the geographic origins and destinations of malware attacks can provide valuable insights into cyber threats worldwide, help identify vulnerabilities in network infrastructures, and enhance overall security measures.

This research unveils a multifaceted approach that combines advanced malware detection with geospatial visualization techniques, providing a comprehensive perspective on the analysis of cyber threats. Our methodology revolves around the thorough examination of packet captures, focusing on HTTP protocol-based file transfers. It involves extracting suspicious files and calculating their hash values to facilitate in-depth scrutiny [2]. Additionally, we leverage the capabilities of the Virus Total platform, which utilizes multiple antivirus engines for rigorous malware scanning. Through this collaborative effort, we accurately assess the threat level associated with these files. The significance of this approach lies in its integration of technology and geospatial awareness. By incorporating geolocation data, we can accurately map the spatial distribution of malware attacks. This provides a comprehensive understanding of the threat landscape beyond technical indicators, revealing the dynamic spatial context where these threats originate.

The importance of our research lies in its holistic approach to combating malware threats. By combining cutting-edge malware detection, comprehensive file analysis, and the invaluable dimension of geospatial visualization, we chart a new course for a proactive and effective cybersecurity strategy. Our approach prioritizes not only the identification of malicious files but also the critical spatial context that shapes their distribution and impact. As this unfolds, we delve into the technical intricacies of our methodology, present

[1, 2, 3, 4, 5, 6] Department of ECE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Green fields, A.P-522302, INDIA

* Corresponding Author Email: shameemsyed@kluniversity.in

empirical findings, and illuminate the ramifications of our work for enhancing network security infrastructure. By combining technology and geography, this study expands the traditional limitations of cybersecurity and introduces a new era of defense that incorporates data and spatial elements to enhance digital security.

II. Methodology

Within the context of our research, traffic analysis takes center stage as the foundational methodology for malware detection. This methodology relies on an exhaustive examination of network traffic to identify and understand malware behaviors and communication patterns [6]. Our methodology primarily centers on traffic analysis, which involves a meticulous inspection of network packets to discern and document the interactions of malware within the network environment. We capture, dissect, and analyze network traffic to identify the communication patterns, potentially malicious behaviors, and the presence of any command-and-control servers. This approach equips us with a comprehensive understanding of the malware's network-level activities, allowing us to detect, analyze, and respond to malware threats in real-time.

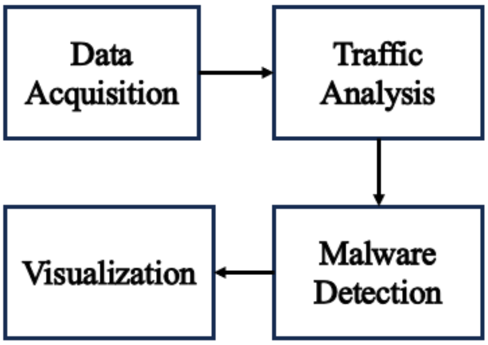
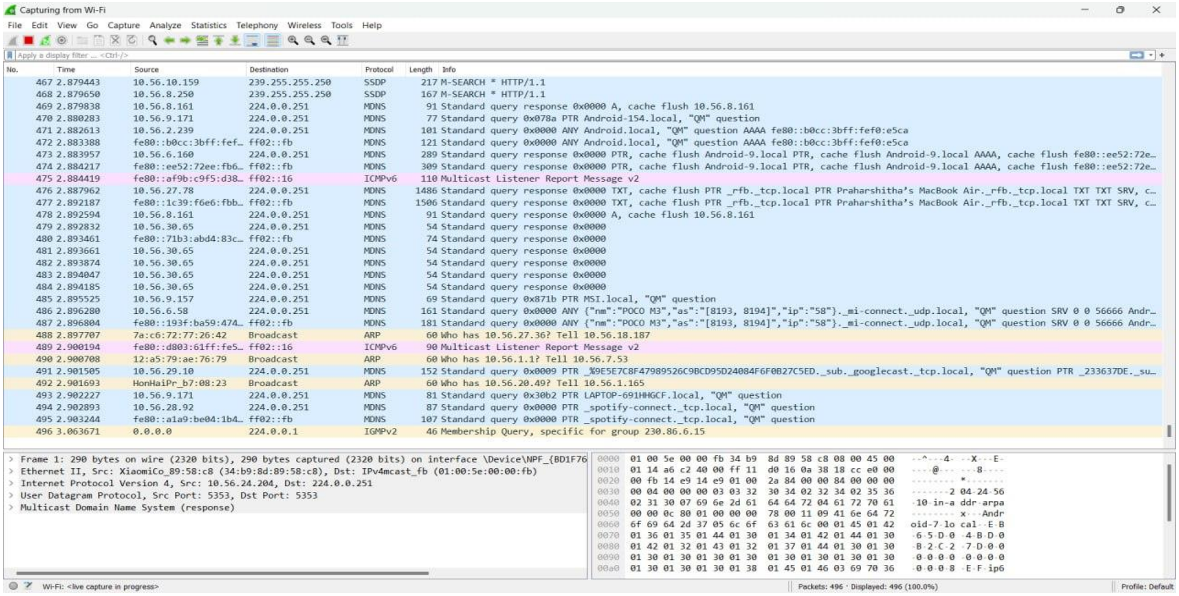


Fig. 1. Proposed Methodology.

By focusing exclusively on traffic analysis, we streamline our methodology as shown in fig. 1 to provide a precise and effective approach to malware detection.


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Wi-Fi: <live capture in progress>
Packets: 496 · Displayed: 496 (100.0%)
Profile: Default

```
No.    Time      Source               Destination      Protocol Length Info
   467 2.879443  10.56.10.159         239.255.255.250  SSDP      217 M-SEARCH * HTTP/1.1
   468 2.879650  10.56.8.250          239.255.255.250  SSDP      167 M-SEARCH * HTTP/1.1
   469 2.879838  10.56.8.161          224.0.0.251      MDNS       91 Standard query response 0x0000 A, cache flush 10.56.8.161
   470 2.880283  10.56.9.171          224.0.0.251      MDNS       77 Standard query 0x078a PTR Android-154.local, "QM" question
   471 2.882613  10.56.2.239          224.0.0.251      MDNS      101 Standard query 0x0000 ANY Android.local, "QM" question AAAA fe80::b0cc:3bff:fef0:e5ca
   472 2.883388  fe80::b0cc:3bff:fef… ff02::fb         MDNS      121 Standard query 0x0000 ANY Android.local, "QM" question AAAA fe80::b0cc:3bff:fef0:e5ca
   473 2.883957  10.56.6.160          224.0.0.251      MDNS      289 Standard query response 0x0000 PTR, cache flush Android-9.local PTR, cache flush Android-9.local AAAA, cache flush fe80::ee52:72e…
   474 2.884217  fe80::ee52:72ee:fb6… ff02::fb         MDNS      309 Standard query response 0x0000 PTR, cache flush Android-9.local PTR, cache flush Android-9.local AAAA, cache flush fe80::ee52:72e…
   475 2.884419  fe80::af9b:c9f5:d38… ff02::16         ICMPv6    110 Multicast Listener Report Message v2
   476 2.887962  10.56.27.78          224.0.0.251      MDNS     1486 Standard query response 0x0000 TXT, cache flush PTR _rfb._tcp.local PTR Praharshitha’s MacBook Air._rfb._tcp.local TXT TXT SRV, c…
   477 2.892187  fe80::1c39:f6e6:fbb… ff02::fb         MDNS     1506 Standard query response 0x0000 TXT, cache flush PTR _rfb._tcp.local PTR Praharshitha’s MacBook Air._rfb._tcp.local TXT TXT SRV, c…
   478 2.892594  10.56.8.161          224.0.0.251      MDNS       91 Standard query response 0x0000 A, cache flush 10.56.8.161
   479 2.892832  10.56.30.65          224.0.0.251      MDNS       54 Standard query response 0x0000
   480 2.893461  fe80::71b3:abd4:83c… ff02::fb         MDNS       74 Standard query response 0x0000
   481 2.893661  10.56.30.65          224.0.0.251      MDNS       54 Standard query response 0x0000
   482 2.893874  10.56.30.65          224.0.0.251      MDNS       54 Standard query response 0x0000
   483 2.894047  10.56.30.65          224.0.0.251      MDNS       54 Standard query response 0x0000
   484 2.894185  10.56.30.65          224.0.0.251      MDNS       54 Standard query response 0x0000
   485 2.895525  10.56.9.157          224.0.0.251      MDNS       69 Standard query 0x871b PTR MSI.local, "QM" question
   486 2.896280  10.56.6.58           224.0.0.251      MDNS      161 Standard query 0x0000 ANY {"nm":"POCO M3","as":"[8193, 8194]","ip":"58"}._mi-connect._udp.local, "QM" question SRV 0 0 56666 Andr…
   487 2.896804  fe80::193f:ba59:474… ff02::fb         MDNS      181 Standard query 0x0000 ANY {"nm":"POCO M3","as":"[8193, 8194]","ip":"58"}._mi-connect._udp.local, "QM" question SRV 0 0 56666 Andr…
   488 2.897707  7a:c6:72:77:26:42    Broadcast        ARP        60 Who has 10.56.27.36? Tell 10.56.18.187
   489 2.900194  fe80::d803:61ff:fe5… ff02::16         ICMPv6     90 Multicast Listener Report Message v2
   490 2.900708  12:a5:79:ae:76:79    Broadcast        ARP        60 Who has 10.56.1.1? Tell 10.56.7.53
   491 2.901505  10.56.29.10          224.0.0.251      MDNS      152 Standard query 0x0009 PTR _%9E5E7C8F47989526C9BCD95D24084F6F0B27C5ED._sub._googlecast._tcp.local, "QM" question PTR _233637DE._su…
   492 2.901693  HonHaiPr_b7:08:23    Broadcast        ARP        60 Who has 10.56.20.49? Tell 10.56.1.165
   493 2.902227  10.56.9.171          224.0.0.251      MDNS       81 Standard query 0x30b2 PTR LAPTOP-691HHGCF.local, "QM" question
   494 2.902893  10.56.28.92          224.0.0.251      MDNS       87 Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question
   495 2.903244  fe80::a1a9:be04:1b4… ff02::fb         MDNS      107 Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question
   496 3.063671  0.0.0.0              224.0.0.1        IGMPv2     46 Membership Query, specific for group 230.86.6.15
```

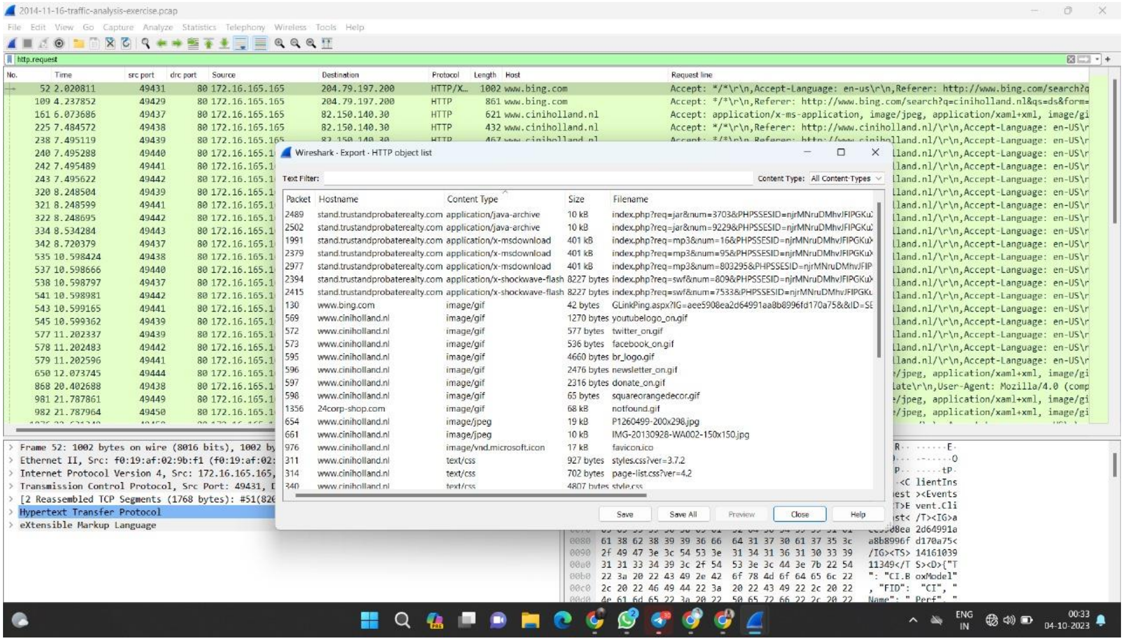
```
> Frame 1: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits) on interface \Device\NPF_{BD1F76
> Ethernet II, Src: XiaomiCo_89:58:c8 (34:b9:8d:89:58:c8), Dst: IPv4mcast_fb (01:00:5e:00:00:fb)
> Internet Protocol Version 4, Src: 10.56.24.204, Dst: 224.0.0.251
> User Datagram Protocol, Src Port: 5353, Dst Port: 5353
> Multicast Domain Name System (response)
```

```
0000  01 00 5e 00 00 fb 34 b9  8d 89 58 c8 08 00 45 00   ··^···4· ··X···E·
0010  01 14 a6 c2 40 00 ff 11  d0 16 0a 38 18 cc e0 00   ····@··· ···8····
0020  00 fb 14 e9 14 e9 01 00  2a 84 00 00 84 00 00 00   ········ *·······
0030  00 04 00 00 00 03 03 32  30 34 02 32 34 02 35 36   ·······2 04·24·56
0040  02 31 30 07 69 6e 2d 61  64 64 72 04 61 72 70 61   ·10·in-a ddr·arpa
0050  00 00 0c 80 01 00 00 00  78 00 11 09 41 6e 64 72   ········ x···Andr
0060  6f 69 64 2d 37 05 6c 6f  63 61 6c 00 01 45 01 42   oid-7·lo cal··E·B
0070  01 36 01 35 01 44 01 30  01 34 01 42 01 44 01 30   ·6·5·D·0 ·4·B·D·0
0080  01 42 01 32 01 43 01 32  01 37 01 44 01 30 01 30   ·B·2·C·2 ·7·D·0·0
0090  01 30 01 30 01 30 01 30  01 30 01 30 01 30 01 30   ·0·0·0·0 ·0·0·0·0
00a0  01 30 01 30 01 30 01 38  01 45 01 46 03 69 70 36   ·0·0·0·8 ·E·F·ip6
```

Fig. 2. Data Capture and Packet Analysis Setup.

In the preliminary stage of our investigation, we established a solid data acquisition methodology by carefully collecting and capturing packets. To ensure the thoroughness and precision of our study, we utilized Wireshark [7], a reputable tool for packet analysis as shown in fig. 2. Our capture was configured to target HTTP traffic and carried out in a controlled environment with network traffic mirroring capabilities. This crucial preparatory measure enabled us to intercept and record network packets that primarily encompassed HTTP transactions—a widely used medium for file transfers in contemporary networking practices. To collect data, we employed Wireshark's capture filters and focused on capturing HTTP traffic [8]. This approach enabled us to selectively isolate packets related to file transfers and other file-related transactions for further analysis. By carefully selecting the network traffic components of interest, we were able to maintain the integrity and efficiency of our dataset.

In order to detect potential threats, we implemented a meticulous approach within our broader methodology. This involved the careful examination of file transfers within captured network traffic. To accomplish this, we utilized Wireshark, a well-known packet analysis tool renowned for its robust capabilities. Specifically, Wireshark was configured to analyze HTTP traffic which is commonly used for file transfers in network communication. Our process focused on identifying and delineating specific HTTP transactions related to both downloads and uploads. By conducting protocol analysis in Wireshark, we were able to identify transactions that displayed characteristics indicative of file transfers such as the presence of binary content within HTTP packets. Additionally, our approach included extracting and preserving relevant data associated with these files for further analysis purposes. To enhance the accuracy and effectiveness of our cybersecurity strategy, we initially focused on isolating specific transactions within the network traffic. This targeted approach allowed us to optimize subsequent malware analysis and geographical visualization efforts, ultimately leading to a more comprehensive examination of potential threats in the network traffic.



2014-11-16-traffic-analysis-exercise.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http.request

```
No.   Time       src port  drc port Source          Destination     Protocol Length Host                Request line
   52 2.020811   49431     80 172.16.165.165   204.79.197.200  HTTP/X…   1002 www.bing.com        Accept: */*\r\n,Accept-Language: en-us\r\n,Referer: http://www.bing.com/search?q
  109 4.237852   49429     80 172.16.165.165   204.79.197.200  HTTP       861 www.bing.com        Accept: */*\r\n,Referer: http://www.bing.com/search?q=ciniholland.nl&qs=ds&form=
  161 6.073686   49437     80 172.16.165.165   82.150.140.30   HTTP       621 www.ciniholland.nl  Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gi
  225 7.484572   49438     80 172.16.165.165   82.150.140.30   HTTP       432 www.ciniholland.nl  Accept: */*\r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
  238 7.495119   49439     80 172.16.165.165
  240 7.495288   49440     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  242 7.495489   49441     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  243 7.495622   49442     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  320 8.248504   49439     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  321 8.248599   49441     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  322 8.248695   49442     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  334 8.534284   49443     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  342 8.720379   49437     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  535 10.598424  49438     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  537 10.598666  49440     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  538 10.598797  49437     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  541 10.598981  49442     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  543 10.599165  49441     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  545 10.599362  49439     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  577 11.202337  49439     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  578 11.202483  49442     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  579 11.202596  49441     80 172.16.165.1                                                      lland.nl/\r\n,Accept-Language: en-US\r
  650 12.073745  49444     80 172.16.165.1                                                      e/jpeg, application/xaml+xml, image/gi
  868 20.402688  49438     80 172.16.165.1                                                      late\r\n,User-Agent: Mozilla/4.0 (comp
  981 21.787861  49449     80 172.16.165.1                                                      e/jpeg, application/xaml+xml, image/gi
  982 21.787964  49450     80 172.16.165.1                                                      e/jpeg, application/xaml+xml, image/gi
```

```
> Frame 52: 1002 bytes on wire (8016 bits), 1002 by
> Ethernet II, Src: f0:19:af:02:9b:f1 (f0:19:af:02:
> Internet Protocol Version 4, Src: 172.16.165.165,
> Transmission Control Protocol, Src Port: 49431, D
> [2 Reassembled TCP Segments (1768 bytes): #51(826
> Hypertext Transfer Protocol
> eXtensible Markup Language
```

```
0080  61 38 62 38 39 39 36 66  64 31 37 30 61 37 35 3c   a8b8996f d170a75<
0090  2f 49 47 3e 3c 54 53 3e  31 34 31 36 31 30 33 39   /IG><TS> 14161039
00a0  31 31 33 34 39 3c 2f 54  53 3e 3c 44 3e 7b 22 54   11349</T S><D>{"T
00b0  22 3a 20 22 43 49 2e 42  6f 78 4d 6f 64 65 6c 22   ": "CI.B oxModel"
00c0  2c 20 22 46 49 44 22 3a  20 22 43 49 22 2c 20 22   , "FID":  "CI", "
```

Wireshark · Export · HTTP object list

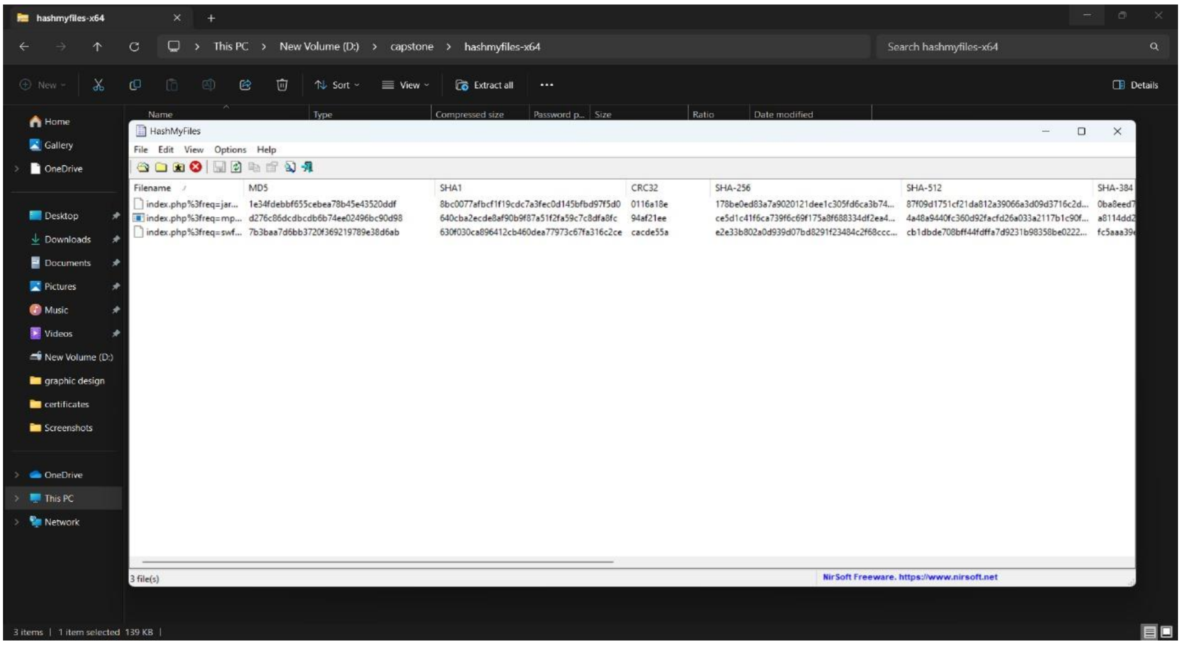
Text Filter:
Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
2489	stand.trustandprobaterealty.com	application/java-archive	10 kB	index.php?req=jar&num=3703&PHPSSESID=njrMNruDMhvJFIPGKu
2502	stand.trustandprobaterealty.com	application/java-archive	10 kB	index.php?req=jar&num=9229&PHPSSESID=njrMNruDMhvJFIPGKu
1991	stand.trustandprobaterealty.com	application/x-msdownload	401 kB	index.php?req=mp3&num=16&PHPSSESID=njrMNruDMhvJFIPGKu
2379	stand.trustandprobaterealty.com	application/x-msdownload	401 kB	index.php?req=mp3&num=95&PHPSSESID=njrMNruDMhvJFIPGKu
2977	stand.trustandprobaterealty.com	application/x-msdownload	401 kB	index.php?req=mp3&num=803295&PHPSSESID=njrMNruDMhvJFIP
2394	stand.trustandprobaterealty.com	application/x-shockwave-flash	8227 bytes	index.php?req=swf&num=809&PHPSSESID=njrMNruDMhvJFIPGKu
2415	stand.trustandprobaterealty.com	application/x-shockwave-flash	8227 bytes	index.php?req=swf&num=7533&PHPSSESID=njrMNruDMhvJFIPGKu
130	www.bing.com	image/gif	42 bytes	GLinkPing.aspx?IG=aee5908ea2d64991aa8b8996fd170a75&&ID=SE
569	www.ciniholland.nl	image/gif	1270 bytes	youtubelogo_on.gif
572	www.ciniholland.nl	image/gif	577 bytes	twitter_on.gif
573	www.ciniholland.nl	image/gif	536 bytes	facebook_on.gif
595	www.ciniholland.nl	image/gif	4660 bytes	br_logo.gif
596	www.ciniholland.nl	image/gif	2476 bytes	newsletter_on.gif
597	www.ciniholland.nl	image/gif	2316 bytes	donate_on.gif
598	www.ciniholland.nl	image/gif	65 bytes	squareorangedecor.gif
1356	24corp-shop.com	image/gif	68 kB	notfound.gif
654	www.ciniholland.nl	image/jpeg	19 kB	P1260499-200x298.jpg
661	www.ciniholland.nl	image/jpeg	10 kB	IMG-20130928-WA002-150x150.jpg
976	www.ciniholland.nl	image/vnd.microsoft.icon	17 kB	favicon.ico
311	www.ciniholland.nl	text/css	927 bytes	styles.css?ver=3.7.2
314	www.ciniholland.nl	text/css	702 bytes	page-list.css?ver=4.2
340	www.ciniholland.nl	text/css	4807 bytes	style.css

Fig. 3. Precise File Data Extraction.

In the process of extracting file data, we delved into the packet captures obtained through the Wireshark packet analysis tool as shonw in fig. 3. The focus of this phase was on isolating and preserving the content of file transfers within the HTTP traffic. As HTTP transactions inherently encapsulate data payloads in their response packets, we meticulously identified and extracted these packets to unveil the content of the transmitted files. The approach ensured that no data was left behind, maintaining the integrity of the files as they were initially transferred over the network. A precise packet analysis was conducted to ensure the accurate extraction of file data. This involved a comprehensive examination of the HTTP response packets to pinpoint the exact boundaries of the file content. By inspecting the packet payload and content-type headers, we were able to differentiate between various file types, such as executables, images, documents, and

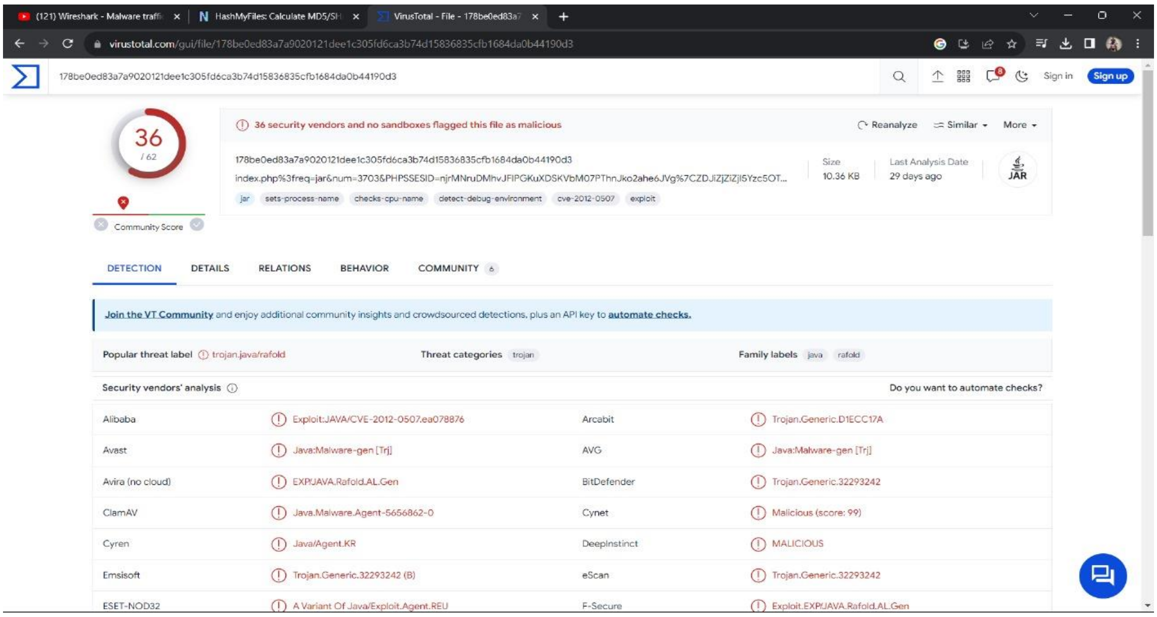
compressed archives. Once the file data was accurately isolated, it was saved in a binary format that preserved the original structure and content, ensuring that it could be subjected to further analysis, including hashing and malware scans. Throughout the extraction process, a fundamental consideration was the preservation of data integrity. The file content was captured without any alteration, ensuring that the extracted files were true replicas of their original counterparts. This attention to data integrity was essential not only for accurate hashing but also for conducting reliable malware scans. By upholding the authenticity and completeness of the file data, our methodology laid the groundwork for comprehensive security analysis and geospatial visualization, providing a robust foundation for evaluating the global distribution of malware attacks.



Filename	MD5	SHA1	CRC32	SHA-256	SHA-512	SHA-384
index.php%3freq=jar...	1e34fdebbf655cebea78b45e43520ddf	8bc0077afbcf1f19cdc7a3fec0d145bfbd97f5d0	0116a18e	178be0ed83a7a9020121dee1c305fd6ca3b74...	87f09d1751cf21da812a39066a3d09d3716c2d...	0ba8eed7
index.php%3freq=mp...	d276c86dcdbcdb6b74ee02496bc90d98	640cba2ecde8af90b9f87a51f2fa59c7c8dfa8fc	94af21ee	ce5d1c41f6ca739f6c69f175a8f688334df2ea4...	4a48a9440fc360d92facfd26a033a2117b1c90f...	a8114dd2
index.php%3freq=swf...	7b3baa7d6bb3720f369219789e38d6ab	630f030ca896412cb460dea77973c67fa316c2ce	cacde55a	e2e33b802a0d939d07bd8291f23484c2f68ccc...	cb1dbde708bff44fdffa7d9231b98358be0222...	fc5aaa39

Fig. 4. Hash File Generation.

Utilizing standard hashing algorithms, including MD5, SHA-1, and SHA-256 [9], unique hash values were generated for each of the extracted files as shown in fig. 4. These hash values, calculated in a robust and consistent manner, served the dual purpose of verifying the integrity of the files and facilitating subsequent analytical procedures. The resulting hash values were retained for reference throughout the study, allowing for the precise identification and evaluation of individual files within the context of malware analysis and geospatial visualization.


(121) Wireshark - Malware traffi
HashMyFiles: Calculate MD5/SH
VirusTotal - File - 178be0ed83a7
virustotal.com/gui/file/178be0ed83a7a9020121dee1c305fd6ca3b74d15836835cfb1684da0b44190d3
178be0ed83a7a9020121dee1c305fd6ca3b74d15836835cfb1684da0b44190d3
Sign in
Sign up
36
/ 62
Community Score
Reanalyze
Similar
More
JAR
DETECTION
DETAILS
RELATIONS
BEHAVIOR
COMMUNITY 6

36 security vendors and no sandboxes flagged this file as malicious

178be0ed83a7a9020121dee1c305fd6ca3b74d15836835cfb1684da0b44190d3
index.php%3freq=jar&num=3703&PHPSSESID=njrMNruDMhvJFIPGKuXDSKVbM07PThnJko2ahe6JVg%7CZDJiZjZiZjI5Yzc5OT...

Size: 10.36 KB
Last Analysis Date: 29 days ago

jar sets-process-name checks-cpu-name detect-debug-environment cve-2012-0507 exploit

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

Popular threat label: trojan.java/rafold
Threat categories: trojan
Family labels: java rafold

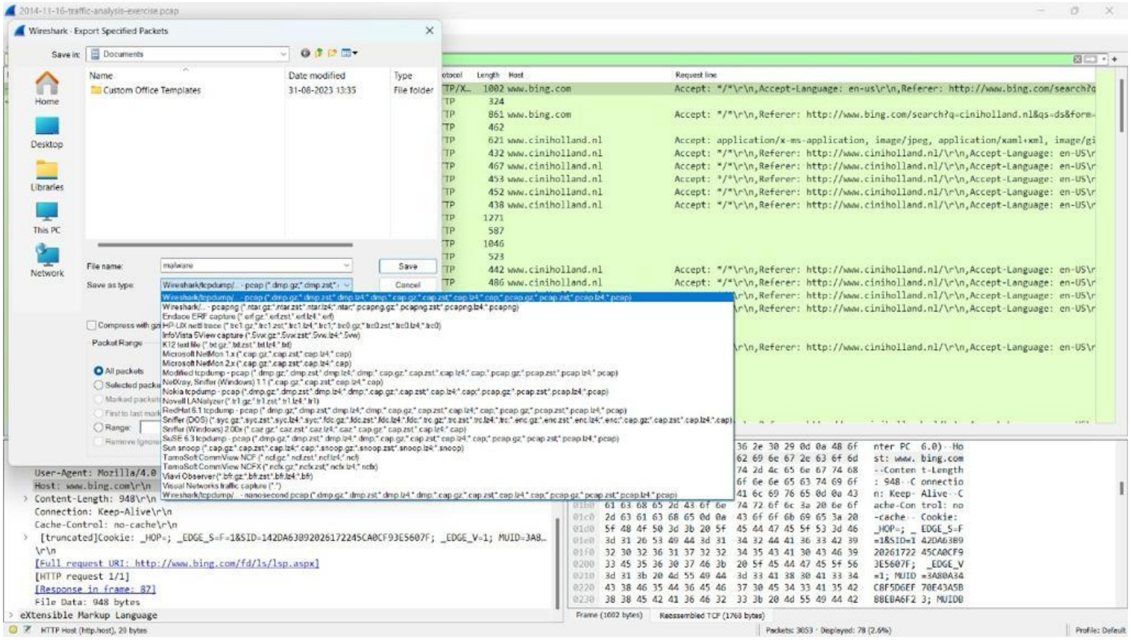
Security vendors' analysis

Do you want to automate checks?

Alibaba	Exploit:JAVA/CVE-2012-0507.ea078876	Arcabit	Trojan.Generic.D1ECC17A
Avast	Java:Malware-gen [Trj]	AVG	Java:Malware-gen [Trj]
Avira (no cloud)	EXP/JAVA.Rafold.AL.Gen	BitDefender	Trojan.Generic.32293242
ClamAV	Java.Malware.Agent-5656862-0	Cynet	Malicious (score: 99)
Cyren	Java/Agent.KR	DeepInstinct	MALICIOUS
Emsisoft	Trojan.Generic.32293242 (B)	eScan	Trojan.Generic.32293242
ESET-NOD32	A Variant Of Java/Exploit.Agent.REU	F-Secure	Exploit.EXP/JAVA.Rafold.AL.Gen

Fig. 5. virus detection from hash files

VirusTotal plays a pivotal role in assessing the potential threat level of the extracted files [10]. VirusTotal, a widely recognized online service, offers a robust platform for malware analysis by subjecting files to scrutiny by multiple antivirus engines. The platform leverages a comprehensive and continuously updated database of malware signatures and characteristics, making it a potent tool for identifying potentially malicious files. Upon extracting the files from network traffic, each file is individually uploaded to the VirusTotal platform for analysis. The platform initiates an array of antivirus scans, checking the file against the signature databases of numerous antivirus engines, ranging from well-known security vendors to open-source solutions.Each engine evaluates the file for known malware attributes, employing a diverse set of detection techniques, including signature matching, behavioral analysis, and heuristics. VirusTotal aggregates the results of these scans into a report that includes the number of engines that flagged the file as potentially malicious, along with additional metadata about the file. The report serves as a critical determinant of the file's threat level. Based on this analysis, our research establishes an informed classification of the files, distinguishing those potentially harboring malware from those that exhibit no such characteristics. VirusTotal's extensive coverage and multifaceted scanning approach add depth and reliability to the malware analysis component of our methodology, enabling a comprehensive assessment of the files' potential threat and facilitating a more nuanced understanding of the risks involved in the network traffic under scrutiny. Incorporating geospatial visualization into our methodology forms a pivotal aspect of our research. Geospatial visualization serves as the means through which we add a critical layer of context to our analysis. This component enables us to transcend conventional data inspection by harnessing the power of geography, an invaluable dimension that sheds light on the spatial distribution of cyber threats. Through the mapping of the geographical locations associated with both the source and destination IP [11] addresses found within our network traffic data, we unlock the potential to visually represent the global scope of malicious file transfers. This spatial context provides security professionals with a unique vantage point, allowing them to observe the dispersal of threats in real-world terms. Such insight proves instrumental in identifying localized vulnerabilities, tracing the origins of attacks, and making informed decisions in incident response and threat mitigation strategies. The synergy between malware detection and geospatial visualization forms the bedrock of our integrated approach, offering a multifaceted perspective on the analysis of cyber threats.



2014-11-16-traffic-analysis-exercise.pcap

Wireshark · Export Specified Packets

Save in: Documents

Name	Date modified	Type
Custom Office Templates	31-08-2023 13:35	File folder

File name: malware

Save as type: Wireshark/tcpdump/... - pcap (*.dmp.gz;*.dmp.zst;*...

```
Wireshark/tcpdump/... - pcap (*.dmp.gz;*.dmp.zst;*.dmp.lz4;*.dmp;*.cap.gz;*.cap.zst;*.cap.lz4;*.cap;*.pcap.gz;*.pcap.zst;*.pcap.lz4;*.pcap)
Wireshark/... - pcapng (*.ntar.gz;*.ntar.zst;*.ntar.lz4;*.ntar;*.pcapng.gz;*.pcapng.zst;*.pcapng.lz4;*.pcapng)
Endace ERF capture (*.erf.gz;*.erf.zst;*.erf.lz4;*.erf)
HP-UX nettl trace (*.trc1.gz;*.trc1.zst;*.trc1.lz4;*.trc1;*.trc0.gz;*.trc0.zst;*.trc0.lz4;*.trc0)
InfoVista 5View capture (*.5vw.gz;*.5vw.zst;*.5vw.lz4;*.5vw)
K12 text file (*.txt.gz;*.txt.zst;*.txt.lz4;*.txt)
Microsoft NetMon 1.x (*.cap.gz;*.cap.zst;*.cap.lz4;*.cap)
Microsoft NetMon 2.x (*.cap.gz;*.cap.zst;*.cap.lz4;*.cap)
Modified tcpdump - pcap (*.dmp.gz;*.dmp.zst;*.dmp.lz4;*.dmp;*.cap.gz;*.cap.zst;*.cap.lz4;*.cap;*.pcap.gz;*.pcap.zst;*.pcap.lz4;*.pcap)
NetXray, Sniffer (Windows) 1.1 (*.cap.gz;*.cap.zst;*.cap.lz4;*.cap)
Nokia tcpdump - pcap (*.dmp.gz;*.dmp.zst;*.dmp.lz4;*.dmp;*.cap.gz;*.cap.zst;*.cap.lz4;*.cap;*.pcap.gz;*.pcap.zst;*.pcap.lz4;*.pcap)
Novell LANalyzer (*.tr1.gz;*.tr1.zst;*.tr1.lz4;*.tr1)
RedHat 6.1 tcpdump - pcap (*.dmp.gz;*.dmp.zst;*.dmp.lz4;*.dmp;*.cap.gz;*.cap.zst;*.cap.lz4;*.cap;*.pcap.gz;*.pcap.zst;*.pcap.lz4;*.pcap)
Sniffer (DOS) (*.syc.gz;*.syc.zst;*.syc.lz4;*.syc;*.fdc.gz;*.fdc.zst;*.fdc.lz4;*.fdc;*.trc.gz;*.trc.zst;*.trc.lz4;*.trc;*.enc.gz;*.enc.zst;*.enc.lz4;*.enc;*.cap.gz;*.cap.zst;*.cap.lz4;*.cap)
Sniffer (Windows) 2.00x (*.caz.gz;*.caz.zst;*.caz.lz4;*.caz;*.cap.gz;*.cap.zst;*.cap.lz4;*.cap)
SuSE 6.3 tcpdump - pcap (*.dmp.gz;*.dmp.zst;*.dmp.lz4;*.dmp;*.cap.gz;*.cap.zst;*.cap.lz4;*.cap;*.pcap.gz;*.pcap.zst;*.pcap.lz4;*.pcap)
Sun snoop (*.cap.gz;*.cap.zst;*.cap.lz4;*.cap;*.snoop.gz;*.snoop.zst;*.snoop.lz4;*.snoop)
TamoSoft CommView NCF (*.ncf.gz;*.ncf.zst;*.ncf.lz4;*.ncf)
TamoSoft CommView NCFX (*.ncfx.gz;*.ncfx.zst;*.ncfx.lz4;*.ncfx)
Viavi Observer (*.bfr.gz;*.bfr.zst;*.bfr.lz4;*.bfr)
Visual Networks traffic capture (*.*)
Wireshark/tcpdump/... - nanosecond pcap (*.dmp.gz;*.dmp.zst;*.dmp.lz4;*.dmp;*.cap.gz;*.cap.zst;*.cap.lz4;*.cap;*.pcap.gz;*.pcap.zst;*.pcap.lz4;*.pcap)
```

☐ Compress with gz

Packet Range

● All packets
○ Selected packe
○ Marked packets
○ First to last mark
○ Range:
☐ Remove Ignore

```
otocol  Length Host                 Request line
TP/X...   1002 www.bing.com         Accept: */*\r\n,Accept-Language: en-us\r\n,Referer: http://www.bing.com/search?q
TP         324
TP         861 www.bing.com         Accept: */*\r\n,Referer: http://www.bing.com/search?q=ciniholland.nl&qs=ds&form=
TP         462
TP         621 www.ciniholland.nl   Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gi
TP         432 www.ciniholland.nl   Accept: */*\r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
TP         467 www.ciniholland.nl   Accept: */*\r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
TP         453 www.ciniholland.nl   Accept: */*\r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
TP         452 www.ciniholland.nl   Accept: */*\r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
TP         438 www.ciniholland.nl   Accept: */*\r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
TP        1271
TP         587
TP        1046
TP         523
TP         442 www.ciniholland.nl   Accept: */*\r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
TP         486 www.ciniholland.nl   Accept: */*\r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
                                    \r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
                                    \r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
                                    \r\n,Referer: http://www.ciniholland.nl/\r\n,Accept-Language: en-US\r
```

```
    User-Agent: Mozilla/4.0
    Host: www.bing.com\r\n
  > Content-Length: 948\r\n
    Connection: Keep-Alive\r\n
    Cache-Control: no-cache\r\n
  >  [truncated]Cookie: _HOP=; _EDGE_S=F=1&SID=142DA63092026172245CA0CF93E5607F; _EDGE_V=1; MUID=3A8…
    \r\n
    [Full request URI: http://www.bing.com/fd/ls/lsp.aspx]
    [HTTP request 1/1]
    [Response in frame: 87]
    File Data: 948 bytes
> eXtensible Markup Language
```

```
                                36 2e 30 29 0d 0a 48 6f   nter PC  6.0)··Ho
                                62 69 6e 67 2e 63 6f 6d   st: www. bing.com
                                74 2d 4c 65 6e 67 74 68   ··Conten t-Length
                                6f 6e 6e 65 63 74 69 6f   : 948··C onnectio
                                41 6c 69 76 65 0d 0a 43   n: Keep- Alive··C
01b0  61 63 68 65 2d 43 6f 6e   74 72 6f 6c 3a 20 6e 6f   ache-Con trol: no
01c0  2d 63 61 63 68 65 0d 0a   43 6f 6f 6b 69 65 3a 20   -cache·· Cookie: 
01d0  5f 48 4f 50 3d 3b 20 5f   45 44 47 45 5f 53 3d 46   _HOP=; _ EDGE_S=F
01e0  3d 31 26 53 49 44 3d 31   34 32 44 41 36 33 42 39   =1&SID=1 42DA63B9
01f0  32 30 32 36 31 37 32 32   34 35 43 41 30 43 46 39   20261722 45CA0CF9
0200  33 45 35 36 30 37 46 3b   20 5f 45 44 47 45 5f 56   3E5607F;  _EDGE_V
0210  3d 31 3b 20 4d 55 49 44   3d 33 41 38 30 41 33 34   =1; MUID =3A80A34
0220  43 38 46 35 44 36 45 46   37 30 45 34 33 41 35 42   C8F5D6EF 70E43A5B
0230  38 38 45 42 41 36 46 32   33 3b 20 4d 55 49 44 42   88EBA6F2 3; MUIDB
```

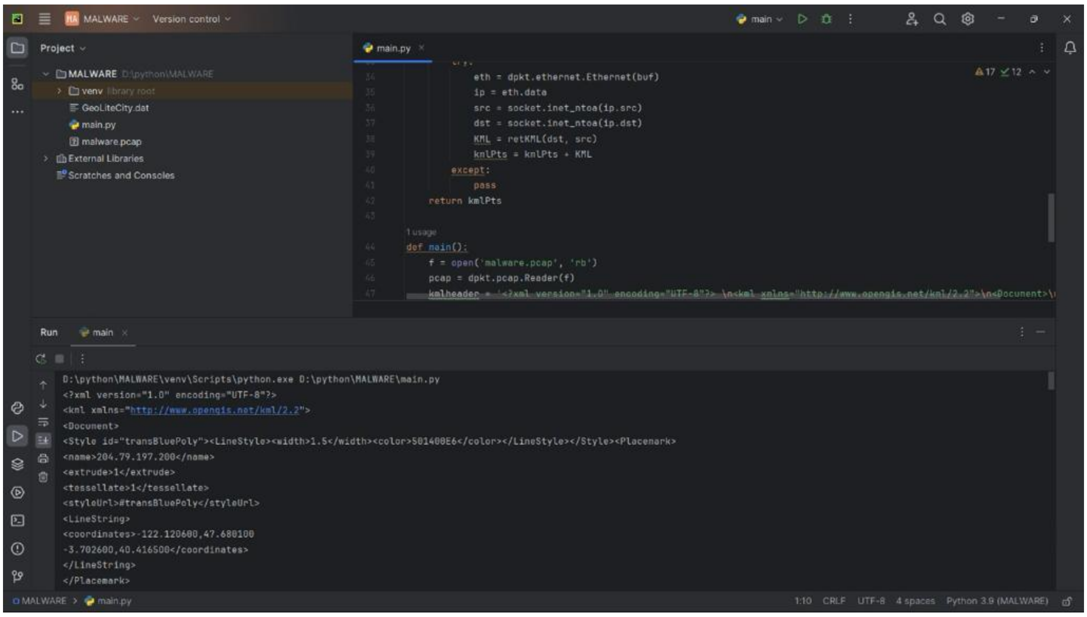
Frame (1002 bytes) | Reassembled TCP (1768 bytes)

HTTP Host (http.host), 20 bytes | Packets: 3053 · Displayed: 78 (2.6%) | Profile: Default

Fig. 6. Saving packets into pcap file

In our geospatial analysis, we have a crucial step where we turn pcap files (which contain data packets in HTML format) into a format called Keyhole Markup Language (KML). This process is handled by a special Python script. It's a bit complex but very important because it helps us connect raw network data to a map view.Here's how it works: The Python script carefully looks inside the pcap files, figuring out things like IP addresses and where they are located on Earth (latitude and longitude). It does this using libraries like pygeoip and dpkt. This creates a detailed dataset that shows where the malware has been.The KML file that comes out of this process is like a dynamic map, showing us the path the malware has taken. Each point on the map represents where the malware was at a certain time, adding a time aspect to our analysis.But this isn't

just about technical stuff; it's also about telling a story. Each point on the map is a part of the malware's journey, helping us understand how it spread and who it might have targeted.After we get the KML file, we carefully inspect it using a text editor like Notepad++. This lets us understand its structure and extract important information. Notepad++ is helpful because it has features that make it easy to work with this kind of data.By going through this process, we make sure we have all the right geographical information for our analysis. This information then gets used in our visualization tools to create detailed maps showing how the malware moved around. This careful inspection ensures that our analysis is accurate and reliable.

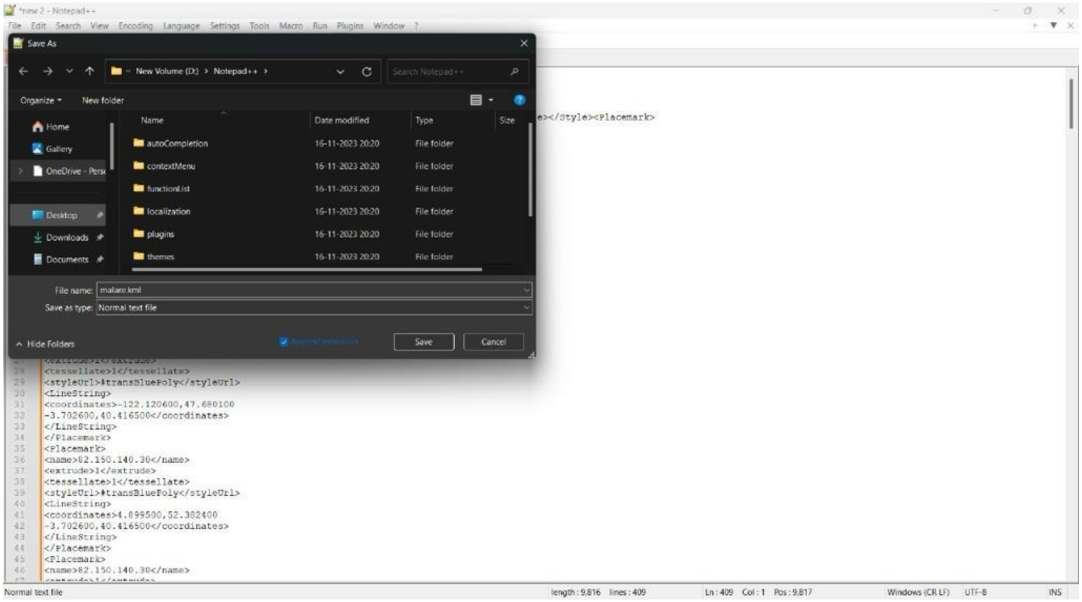


```
            eth = dpkt.ethernet.Ethernet(buf)
            ip = eth.data
            src = socket.inet_ntoa(ip.src)
            dst = socket.inet_ntoa(ip.dst)
            KML = retKML(dst, src)
            kmlPts = kmlPts + KML
        except:
            pass
    return kmlPts

1 usage
def main():
    f = open('malware.pcap', 'rb')
    pcap = dpkt.pcap.Reader(f)
    kmlheader = '<?xml version="1.0" encoding="UTF-8"?> \n<kml xmlns="http://www.opengis.net/kml/2.2">\n<Document>\
```

```
D:\python\MALWARE\venv\Scripts\python.exe D:\python\MALWARE\main.py
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<Style id="transBluePoly"><LineStyle><width>1.5</width><color>501400E6</color></LineStyle></Style><Placemark>
<name>204.79.197.200</name>
<extrude>1</extrude>
<tessellate>1</tessellate>
<styleUrl>#transBluePoly</styleUrl>
<LineString>
<coordinates>-122.120600,47.680100
-3.702600,40.416500</coordinates>
</LineString>
</Placemark>
```

Fig. 7. coverting pcap file into kml using python script


*new 2 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Save As
New Volume (D:) > Notepad++ >
Search Notepad++
Organize
New folder
Home
Gallery
OneDrive - Perso
Desktop
Downloads
Documents
Hide Folders
Save
Cancel
Normal text file
length : 9,816 lines : 409
Ln : 409 Col : 1 Pos : 9,817
Windows (CR LF) UTF-8 INS

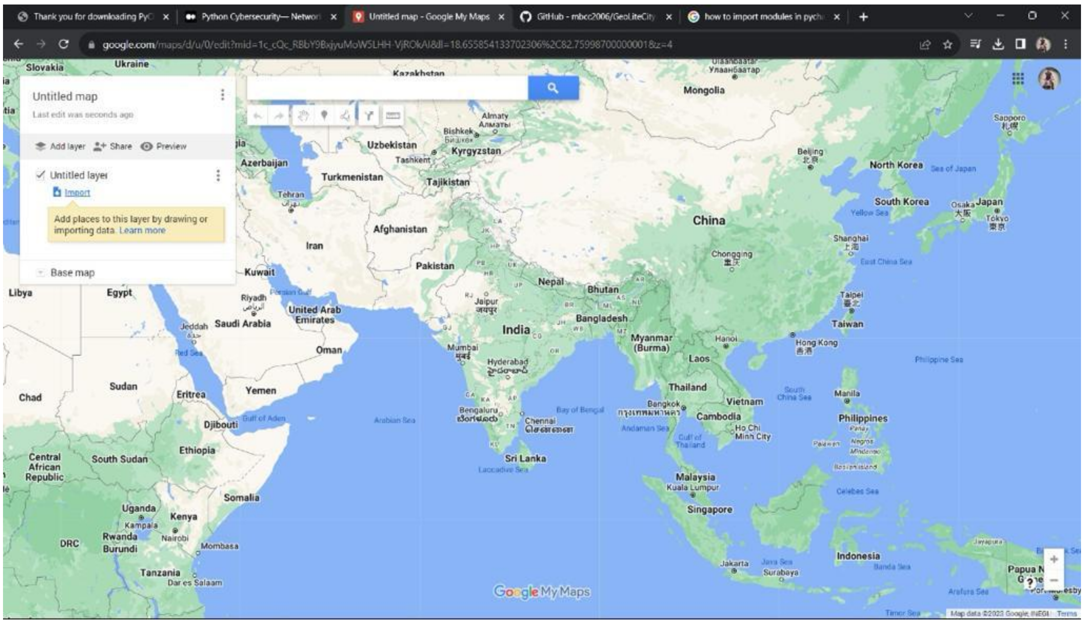
Name	Date modified	Type	Size
autoCompletion	16-11-2023 20:20	File folder	
contextMenu	16-11-2023 20:20	File folder	
functionList	16-11-2023 20:20	File folder	
localization	16-11-2023 20:20	File folder	
plugins	16-11-2023 20:20	File folder	
themes	16-11-2023 20:20	File folder	

File name: malare.kml
Save as type: Normal text file

```
e></Style><Placemark>
<tessellate>1</tessellate>
<styleUrl>#transBluePoly</styleUrl>
<LineString>
<coordinates>-122.120600,47.680100
-3.702600,40.416500</coordinates>
</LineString>
</Placemark>
<Placemark>
<name>82.150.140.30</name>
<extrude>1</extrude>
<tessellate>1</tessellate>
<styleUrl>#transBluePoly</styleUrl>
<LineString>
<coordinates>4.899500,52.382400
-3.702600,40.416500</coordinates>
</LineString>
</Placemark>
<Placemark>
<name>82.150.140.30</name>
```

Fig. 8. KML Inspection and Notepad++ Extraction.

The data transformed from Python into Keyhole Markup Language (KML) represents a crucial juncture in our research, capturing the geographical trajectory of malware propagation. As a visual representation of our analysis, the KML file encapsulates the intricate journey undertaken by the malware, with each coordinate marking a significant point in its path. This spatial narrative not only provides insights into the spread of malware but also adds a temporal dimension to our understanding.Upon generating the KML data, we meticulously transferred it into Notepad++, a versatile text editor chosen for its feature-rich environment and suitability for handling KML documents. Within Notepad++, we performed a careful inspection of the KML document structure, navigating through its components to ensure accuracy and integrity. This phase of scrutiny involved parsing and organizing the essential elements extracted from the KML file.Once the inspection and organization were complete, we saved the curated KML file within Notepad++, preserving the enriched geographical insights crucial for our analysis. This file serves as a treasure trove of spatial data, laying the groundwork for our dynamic geospatial visualization and further exploration of malware propagation patterns.In summary, the transformation of Python data into KML format, followed by meticulous scrutiny and extraction within Notepad++, culminated in the creation of a comprehensive KML file. This document not only captures the geographical nuances of malware propagation but also represents a critical component of our research workflow, enabling us to unravel the spatial tale of cyber threats.



Add places to this layer by drawing or importing data. Learn more

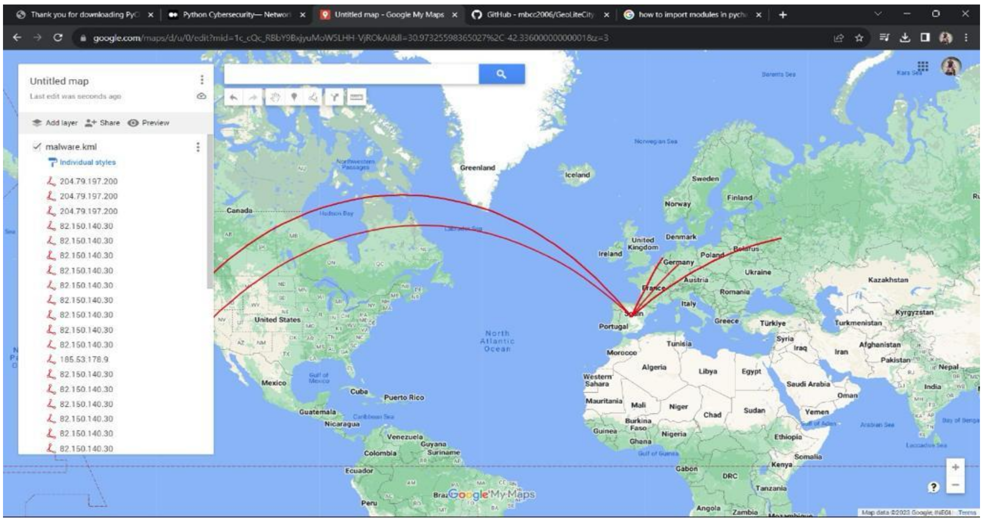
Fig. 9. Adding data to google maps

Upon completion of the KML file generation using Python and GeoliteCity, the data was seamlessly integrated into Google Maps for enhanced visualization and analysis. Leveraging GeoliteCity's database, the Python script meticulously decoded pcap files, extracting vital information such as IP addresses and their corresponding geographical coordinates. This enriched dataset formed the foundation for the creation of the KML file, which encapsulated the spatial trajectory of malware propagation.Importing the KML file into Google Maps facilitated dynamic mapping of the malware's journey, offering a comprehensive spatial representation of its propagation patterns. By overlaying the KML data onto the Google Maps interface, users gained valuable insights into the geographical distribution and movement of the malware. The integration of KML data with Google Maps' versatile features allowed for interactive exploration, enabling users to zoom in on specific locations, toggle visibility of layers, and customize the visualization to suit their analytical needs.This integration of Python-generated KML data with Google Maps not only provided a

visually compelling representation of malware propagation but also empowered users with actionable insights for cybersecurity analysis and strategic decision-making.

III. Result

Our combined efforts in visualizing geographical data and analysing malware using Virus Total have uncovered valuable insights into how malware spreads. By mapping out the paths taken by malware on a map, we not only see where it has been but also understand which areas face higher risks. This helps us identify patterns in how attacks are targeted, spot where they might be coming from, and pinpoint regions with more cybersecurity threats. The information we get from Virus Total adds another layer of detail to our understanding. It tells us more about how malware behaves over time and who it might be targeting. By putting together the geographical data and malware analysis, we're not just looking at pretty maps; we're gaining practical knowledge that helps us strengthen our defences, react to new threats quickly, and actively protect our networks. In essence, our approach isn't just about seeing data visually; it's about using that insight to make smarter decisions and keep our systems safe from cyber threats.



Untitled map

Last edit was seconds ago

malware.kml

Individual styles

204.79.197.200
204.79.197.200
204.79.197.200
82.150.140.30
82.150.140.30
82.150.140.30
82.150.140.30
82.150.140.30
82.150.140.30
82.150.140.30
82.150.140.30
82.150.140.30
185.53.178.9
82.150.140.30
82.150.140.30
82.150.140.30
82.150.140.30
82.150.140.30
82.150.140.30

Fig. 10. Dynamic Geospatial Visualization on Google Maps.

IV. Conclusion

In the fast-paced world of cybersecurity, our unique approach combines advanced techniques in malware detection, thorough file analysis, and visual mapping to give us a deep understanding of how malware spreads. We don't just look at data; we turn it into detailed maps that show us where malware is going and how it behaves.Our process starts by converting HTML packet data into detailed maps, which isn't just about making things look pretty—it's about gaining valuable insights. We carefully follow a step-by-step process, moving from one format to another until we have detailed maps that show us the path of malware across different locations.But we don't stop there. We also analyze the malware using VirusTotal, a powerful tool that helps us understand more about its behavior and potential risks. By combining the data from VirusTotal with our maps, we get a complete picture of the threat landscape. We can see where malware is most active, where it might be coming from, and how it's evolving over time.Our maps aren't just static images; they tell a story. They highlight areas with high levels of threat, point out where attacks might be originating, and show us how cyber threats are changing. This helps cybersecurity professionals make informed decisions, strengthen their defenses, and stay ahead of emerging threats.In a constantly shifting cybersecurity landscape, our approach isn't just about looking back at what happened; it's about being ready for what might come next. By integrating different techniques and staying one step ahead, we're equipped to tackle the complex challenges posed by modern cyber threats.

References

[1] Vinod, P., Jaipur, R., Laxmi, V. and Gaur, M., 2009, March. Survey on malware detection methods. In Proceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09) (pp. 74-79).

[2] Jusas, V., Birvinskas, D. and Gahramanov, E., 2017. Methods and tools of digital triage in forensic context: Survey and future directions. Symmetry, 9(4), p.49.

[3] W.M. Shbair, T. Cholez, J. Francois, I. Chrisment," Improving SNI-based HTTPs security monitoring", IEEE International Conference on Distributed Computing Systems Workshops, pp.72-77, 2016.

[4] Tomas Komarek and Petr Somol "End-node Fingerprinting for Malware Detection on HTTPS Data" In Proceedings of ARES '17, Reggio Calabria, Italy pp. 1-7, 2017.

[5] Anderson, B., McGrew,D, "Identifying Encrypted Malware Traffic with Contextual Flow Data". In ACM Workshop on Artificial Intelligence and Security, pp: 35-46, 2016.

[6] Kohout, J., Komárek, T., Čech, P., Bodnár, J. and Lokoč, J., 2018. Learning communication patterns for malware discovery in HTTPs data. Expert Systems with Applications, 101, pp.129-142.

[7] Beale, J., Orebaugh, A. and Ramirez, G., 2006. Wireshark & Ethereal network protocol analyzer toolkit. Elsevier.

[8] Saxena, P. and Sharma, S.K., 2017. Analysis of network traffic by using packet sniffing tool: Wireshark. Int. J. Adv. Res. Ideas Innov. Technol, 3(6), pp.804-808.

[9] Roshdy, R., Fouad, M. and Aboul-Dahab, M., 2013. Design and Implementation a new Security Hash Algorithm based on MD5 and SHA-256. International Journal of Engineering Sciences & Emerging Technologies, 6(1), pp.29-36.

[10] Choo, E., Nabeel, M., De Silva, R., Yu, T. and Khalil, I., 2022. A large scale study and classification of virustotal reports on phishing and malware urls. arXiv preprint arXiv:2205.13155.

[11] Padmanabhan, V.N. and Subramanian, L., 2001, August. An investigation of geographic mapping techniques for Internet hosts. In Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (pp. 173-185).