A Comprehensive Study and In-depth Campa

zscaler.com/blogs/security-research/hibernating-qakbot-comprehensive-study-and-depth-campaign-analysis

Introduction

In the ever-evolving landscape of cyber threats, banking trojans continue to pose a significant risk to organizations worldwide. Among them, Qakbot, also known as QBot or Pinkslipbot, stands out as a highly sophisticated and persistent malware active since 2007, targeting businesses across different countries. With a primary focus on stealing financial data and login credentials from web browsers, Qakbot also serves as a backdoor to inject next-stage payloads like Cobalt Strike and ransomware. Its adaptability, evasive techniques, and global reach have made it a formidable adversary for cybersecurity professionals seeking to defend against its malicious activities.

As part of our commitment to monitoring active malware campaigns, Zscaler's ThreatLabz team conducts in-depth investigations to uncover the various attack chains employed by Qakbot. In this research article, we delve into the depths of Qakbot, conducting a comprehensive technical analysis to understand its behavior, attack vectors, and distribution methods. We explore its use of diverse file formats, encryption techniques, and attack chains to evade detection and maintain its foothold in infected systems. Our examination also uncovers patterns in its Command and Control (C2) infrastructure and provides valuable insights into its geographic distribution.

Recent campaigns have revealed Qakbot's adaptation to changing conditions. In January 2023, after Microsoft disabled Macros by default in all Office applications, Qakbot began abusing OneNote files as a means of spreading itself. For detailed insights into these campaigns and obfuscation techniques, readers can refer to Zscaler ThreatLabz's research blog on OneNote.

With the use of Zscaler Sandbox, we shed light on the threat scores and specific MITRE ATT&CK techniques triggered by Qakbot during our investigation. Armed with this knowledge, cybersecurity professionals can better equip themselves to counter this persistent malware and protect their networks from its malicious campaigns.

Interestingly, we observed a significant decline in Qakbot activity after June, indicating a potential pause in the threat actor's operations. It appears that the group behind Qakbot has temporarily reduced its activities, which could indicate various factors at play.

Throughout this article, we delve into the intricacies of Qakbot's attack chains, encryption methods, and its wide geographical reach. Our ultimate goal is to empower cybersecurity professionals to better defend against this sophisticated and persistent banking trojan. By fostering collaboration within the cybersecurity community and staying vigilant in monitoring emerging threats, we aim to collectively enhance the security posture of organizations worldwide and preserve the trust of users and businesses alike.

Top 5 Key Takeaways

1. Qakbot - A Persistent Banking Trojan: Qakbot, also known as QBot or Pinkslipbot, has been an active and persistent banking trojan since 2007. It continues to evolve over time, utilizing different techniques to infect users and compromise systems. Qakbot employs diverse attack chains, multiple file formats, and obfuscation methods to avoid detection from antivirus solutions and maintain its foothold in infected systems.

2. OneNote Campaign and Ongoing Activity: Following the OneNote campaign, Qakbot remains highly active, distributing its payload through various attack chains. Despite security measures and patches aimed at mitigating Qakbot's attacks, the threat actors continue to find novel ways to deliver their payload and exploit vulnerable Windows file formats. The malware employs different abusable file formats, including pdf, html, xhtml (eXtended HTML), wsf (Windows Script File), js (Javascript), ps (Powershell), xll (Excel add-in), hta (HTML Application), xmlhttp, etc., in its attack chain to infect users.

3. Global Reach and C2 Infrastructure: The analysis reveals Qakbot's wide geographic distribution, with C2 servers highly active in various countries. This highlights the malware's global reach and capability to target organizations worldwide.

4. Decline in Qakbot Activity: After observing a significant drop in Qakbot activity after June, it appears that the threat actor behind Qakbot has temporarily reduced its operations. The reasons for this decline remain unclear.

5. Collective Defense and Vigilance: Collaboration within the cybersecurity community, proactive monitoring, and adherence to best practices are crucial to effectively counter Qakbot's relentless pursuit. Strengthening security protocols and conducting security awareness training are essential in safeguarding against banking trojans like Qakbot and preserving the integrity of networks and sensitive data.

Analysis of Qakbot Attack Chains

This section presents distinct variations of the Qakbot banking trojan attack chain, examined across samples discovered between March and May of 2023. The case studies below specifically concentrate on how diverse file formats and techniques execute the Qakbot end payload on the victim's machine, instead of directly dropping and executing the malware.

Case Study 1: March 2023 - Evolving Qakbot Tactics: Exploiting File Formats for Deceptive Payload Delivery

At the outset of the year, Qakbot began spreading through OneNote files. Subsequently, in March, a shift was observed, as Qakbot transitioned to using PDF and HTML files as the initial attacking vectors to download further stage files, leading to the delivery of the final payload. These file formats are commonly utilized by numerous threat actors to infect users.

Multiple attack chains were observed, wherein Qakbot utilizes PDF files as the initial vector to download the next stage file, which contains an obfuscated JS (Javascript) file bearing names like "Invoice," "Attach," "Report," or "Attachments" to deceive users into executing the file. Upon running the JS file, Qakbot initially creates a registry key and adds the base64 encoded Powershell command into the registry key using the reg.exe command line tool, enabling the download and execution of the Qakbot DLL.

Attack Chain: MalSpam -> PDF -> URL -> JS -> PS -> Qakbot Payload

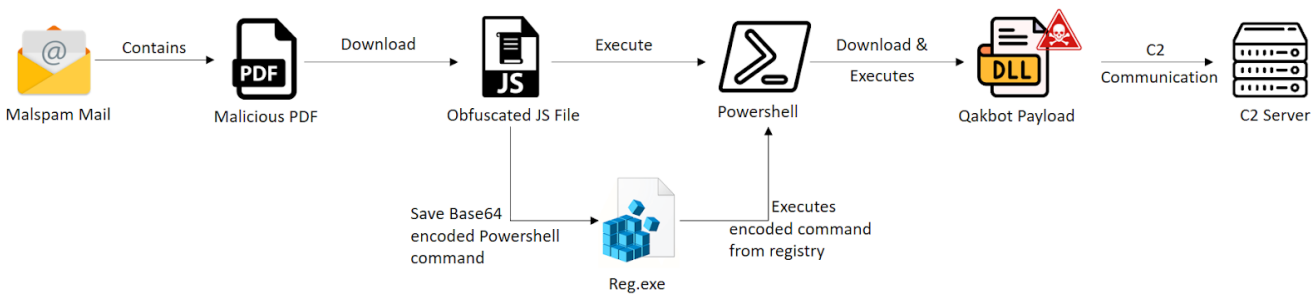

Figure 1 - Illustrates the attack chain involving a Malicious PDF as the initial attack vector.

Qakbot recently reverted to utilizing HTML smuggling as a means of delivering its initial attack payload. This technique was observed across numerous campaigns during the previous year. In March, the identification of several new malspam emails indicated that threat actors were leveraging Latin-themed HTML files to facilitate the download of zip archives. These archives contained an obfuscated JS file, initiating a sequence similar to the one depicted in Fig.1, ultimately leading to the delivery of the Qakbot payload.

The attack chain discovered in March follows the following progression: Malspam -> HTML -> URL -> ZIP -> JS -> PS -> Qakbot Payload

In this chain, malspam serves as the initial delivery method, targeting unsuspecting victims through deceptive emails. The HTML files play a pivotal role in exploiting HTML smuggling techniques, concealing malicious activities within seemingly innocuous web content.

Upon accessing the HTML files, URLs are triggered, initiating the download of zip archives containing the obfuscated JS file. The use of obfuscation ensures that the malicious code remains hidden from casual detection and analysis, enhancing the threat actors' ability to evade detection.

Subsequently, the JS file is executed, setting off a series of actions that culminate in the execution of a Powershell command (PS). The Powershell command is instrumental in obtaining and executing the final payload, which, in this case, is the notorious Qakbot banking trojan. During our campaign follow up we found this sample from Twitter handle @Pr0xylife and @Cryptolaemus1.

This resurgence of HTML smuggling by Qakbot highlights the significance of continuous monitoring and awareness of evolving malware tactics and shifting attack chains for detecting and countering such threats.

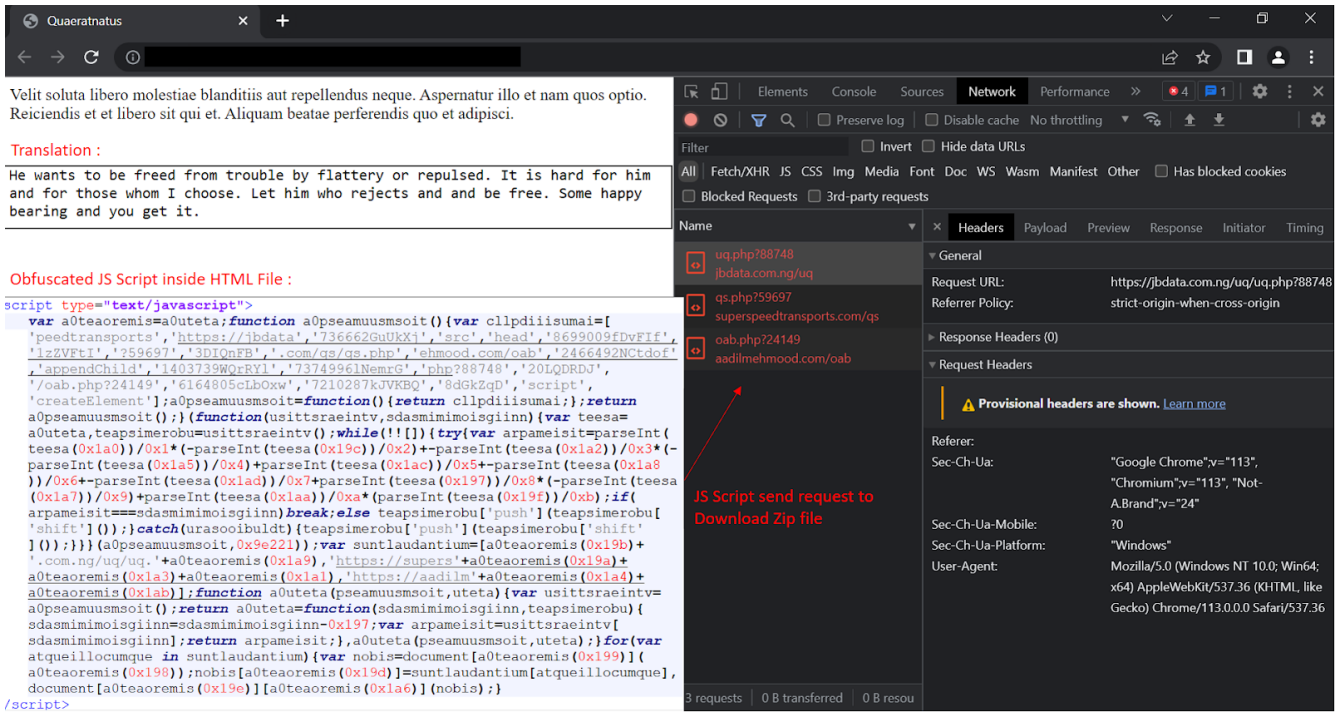

Figure 2 - Shows the attack chain with a Malicious HTML file as the initial attack vector.

Later, a similar attack chain was identified, where the initial attack vector involved a PDF file. This PDF file was designed to download a zip archive, which, in turn, contained an obfuscated WSF/HTA file. Upon execution, the WSF/HTA file ran a base64 encoded Powershell command, leading to the download and execution of the

final Qakbot payload.

The observed attack chain follows the following progression: Malspam -> PDF -> ZIP -> WSF/HTA -> PS -> Qakbot Payload

In this scenario, malspam continues to serve as the initial method of propagation, disseminating malicious content through email campaigns. The PDF file, acting as the attack vector, entices users to access its contents, ultimately triggering the download of a zip archive.

Inside the zip archive, an obfuscated WSF/HTA file is concealed, obscuring its malicious intent and complicating detection efforts. Once executed, the WSF/HTA file initiates a base64 encoded Powershell command, a common technique used by threat actors to download and execute further payloads without leaving a conspicuous trail.

The culmination of this attack chain results in the delivery and execution of the Qakbot banking trojan against the targeted system and its users.

Figure 3 - Features a Malicious PDF as the initial attack vector in the attack chain, accompanied by WSF and HTA files.

In another discovery made by ThreatLabz researchers, a variant of the Qakbot malware was observed employing a stealthy attack chain with the use of Microsoft Excel add-ins (XLL) as the initial vector. Microsoft Office add-ins are DLL files with distinct extensions based on the application they are designed for. While Microsoft Word add-ins use the '.wll' extension, Excel add-ins utilize the '.xll' extension.

The choice of using XLL files as the initial attacking vector is strategic for threat actors due to their ease of use. Unlike Word add-ins that must be placed in specific trusted locations depending on the Office version, XLL files are automatically loaded and opened by the Excel application, simplifying the delivery process for the attackers.

Moreover, XLL files possess unique characteristics that differentiate them from regular DLLs. They can have export functions that are invoked by the Excel Add-In manager when triggered by Excel. Upon launching an XLL file, Excel activates the export functions defined by the XLL interface, such as **xlAutoOpen** and **xlAutoClose**, similar to **Auto_Open** and **Auto_Close** in VBA macros. This mechanism is exploited by the attackers to load the malicious payload seamlessly, evading security measures and detection.

The attack chain follows a sequence where the threat actor utilizes a .xll file in the initial phase. When a user opens this .xll file, it proceeds to drop two files, "**1.dat**" and "**2.dat**," into the **'\Users\User\AppData\Roaming\'** directory. The "**1.dat**" file contains a 400-byte header of the PE file, while the "**2.dat**" file holds the remaining data of the PE file. These two files are then combined to create the "**3.dat**" file, which contains the actual Qakbot payload. Additionally, the attackers establish scheduled tasks to execute the Qakbot payload every 10 minutes, ensuring its persistence on the victim's machine.

The observed attack chain follows the following progression: Malspam -> ZIP -> XLL > Qakbot Payload

This attack chain sample underscores the ever-evolving nature of Qakbot, which continuously adapts its tactics and techniques to avoid detection and infiltrate systems. By utilizing XLL files and implementing sophisticated techniques to hide and deliver its payload, Qakbot continues to pose a significant threat to users and organizations.

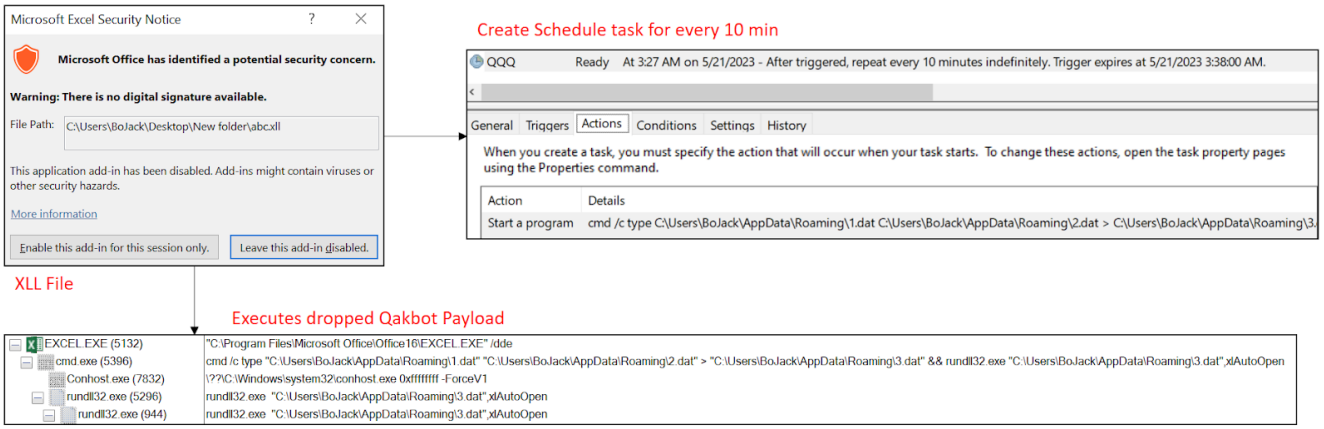

Figure 4 - Shows the attack chain involving Malicious XLL files as the initial attack vector.

Case Study 2: April 2023 - Adapting Qakbot: Unraveling the XMLHTTP Experiment in the Attack Chain

In April, researchers noted more significant changes in the Qakbot attack chain, as the samples revealed the malware continued to experiment with different file formats to infect users.

In this evolved attack chain, the WSF (Windows Script File) contains a hex-encoded **XMLHTTP** request to download the Qakbot payload, replacing the previous base64 encoded PowerShell command.

The observed attack chain follows the following progression: Malspam -> PDF -> ZIP -> WSF -> XMLHTTP -> Qakbot Payload

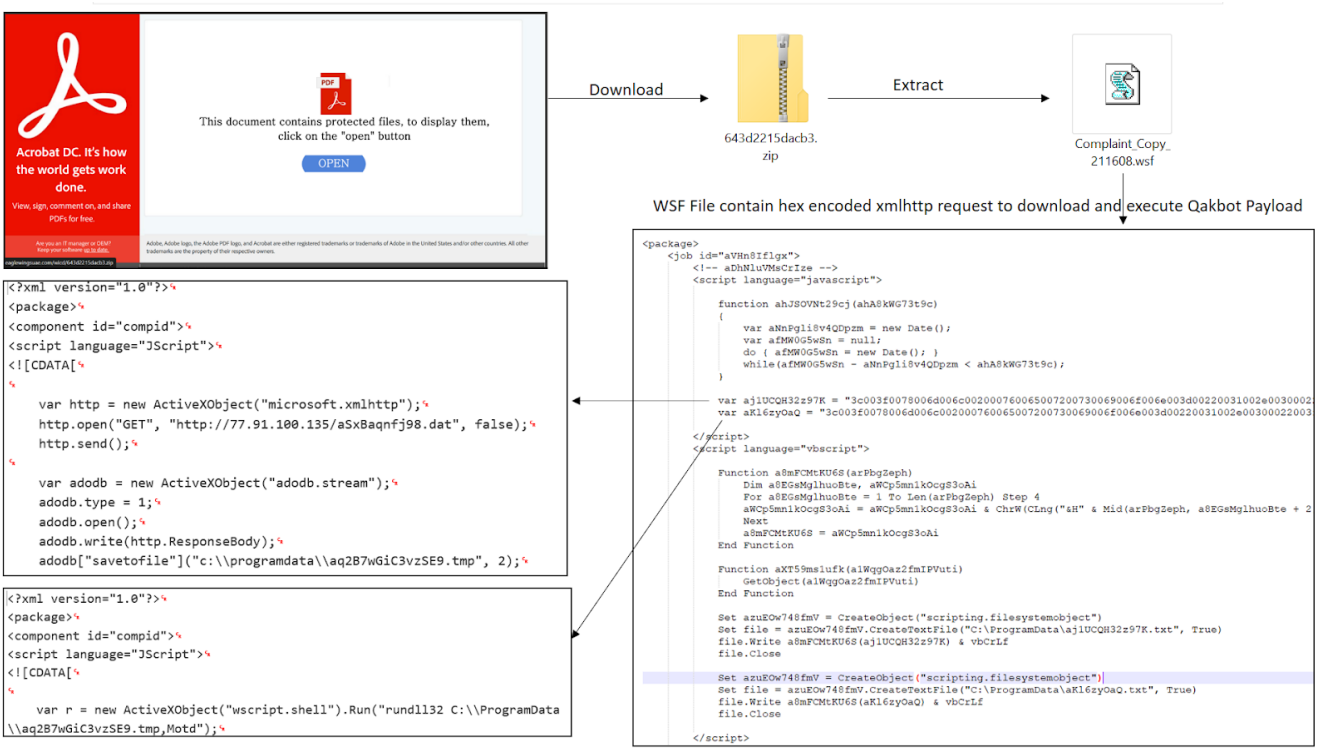

Figure 5 - Depicts the attack chain utilizing the XMLHTTP file.

Towards the end of April, Qakbot's persistent use of OneNote files as the initial attack vector was still evident in its latest campaign. OneNote files served as an effective disguise, luring unsuspecting users into opening and executing the embedded contents. The attackers leveraged the familiarity and widespread use of OneNote files to increase the chances of successful infections.

Within this attack chain, the OneNote file contains an embedded MSI (Microsoft Installer) file. This MSI file was designed to trick users by posing as a legitimate Microsoft Azure installer, exploiting victims' trust in these familiar software installations and delivering the Qakbot payload.

The MSI file was purposely crafted to include several components, enhancing its evasive capabilities and making it difficult for security systems to detect its true intent. Among these components, a self-deletion PowerShell script was incorporated, allowing the malware to erase its tracks after execution, reducing the chances of detection and analysis.

Furthermore, the MSI file contained a configuration file that held essential information, including the path to execute a WSF (Windows Script File) script. This WSF script served as a critical link in the attack chain, acting as an intermediary to facilitate the download and execution of the Qakbot payload.

To ensure further obfuscation and evasion, the WSF script was hex-encoded, making it challenging for traditional security measures to interpret its true purpose. This encoded script was responsible for executing an **XMLHTTP** request, a technique used to download the actual Qakbot payload from a remote server.

Through this intricate sequence of deception and evasion, attackers aim to successfully deliver the Qakbot payload onto victim machines. By continuously adapting their attack techniques and leveraging familiar file formats, the threat actors behind Qakbot seek to stay one step ahead of security defenses and professionals.

The observed attack chain follows the following progression: Malspam -> OneNote -> MSI -> WSF -> XMLHTTP -> Qakbot Payload

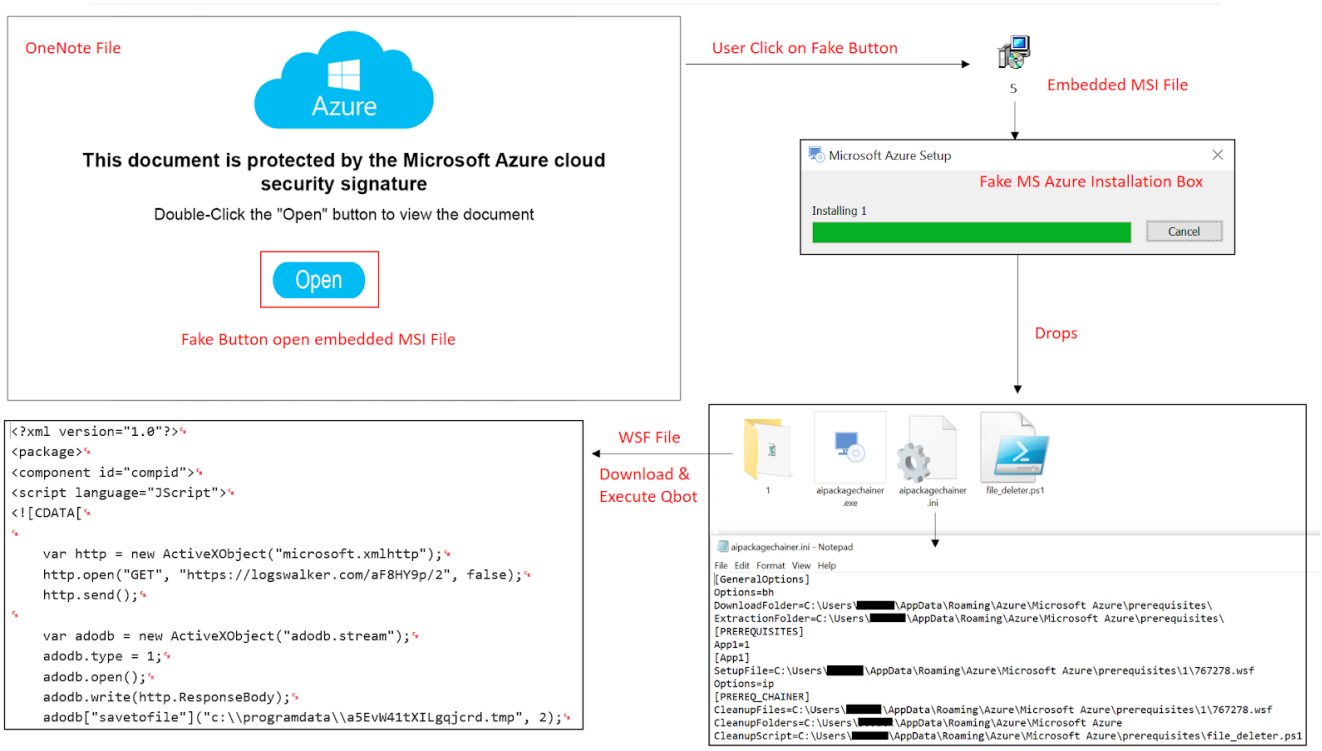

Figure 6 - Evolving Attack Chain: Leveraging Malicious OneNote and MSI Files as Initial Attack Vector.

Case Study 3: May 2023: Qakbot Explores Advanced Defense Evasion Tactics

Throughout the month of May, researchers closely monitored Qakbot's activities and observed the threat actor's efforts to experiment with innovative Defense Evasion Tactics aimed at infecting users and evading detection. Alongside changes in the attack chain, Qakbot introduced sophisticated techniques, including Indirect Command Execution using conhost.exe and DLL Side-Loading, further complicating its detection and removal.

In this attack chain, Qakbot takes advantage of conhost.exe as a proxy binary to bypass defensive measures. By employing conhost.exe, Qakbot attempts to outwit security counter-measures that restrict the use of typical command-line interpreters. This enables the threat actor to execute commands using various Windows utilities, creating a clever diversion and making it more challenging for security tools to identify and mitigate the threat effectively.

The attack sequence starts with malspam, where malicious emails are distributed to unsuspecting victims. These emails often contain malicious attachments disguised as innocent files, luring users into opening them. The threat actors use PDF files packed within ZIP archives, which, when accessed, lead to the execution of WSF files via XMLHTTP.

To further obscure its activities, Qakbot then leverages conhost.exe, employing it as an intermediary to carry out specific commands. This tactic is part of Qakbot's strategy to operate stealthily within the compromised system, remaining undetected by conventional security mechanisms that may primarily focus on detecting direct malicious code execution.

The ultimate goal of this attack chain is to deliver the Qakbot payload, allowing the malware to infiltrate the victim's system, steal sensitive information, and potentially carry out other malicious activities, including espionage and financial theft.

The observed attack chain follows the following progression: Malspam -> PDF -> ZIP -> WSF -> XMLHTTP -> conhost.exe -> Qakbot Payload

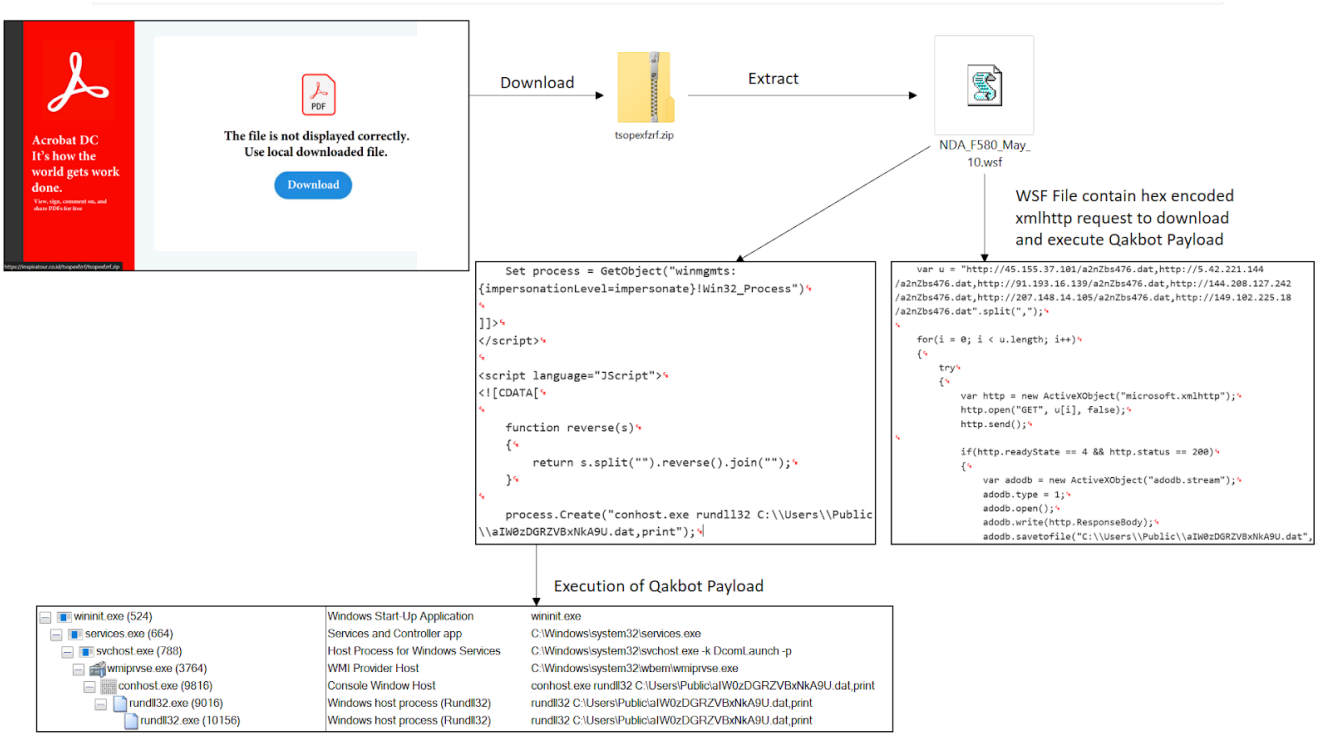

Figure 7 - Demonstrates Qakbot's utilization of Indirect Command Execution with conhost.exe.

In this intricate attack chain, the initial vector is a ZIP file that conceals an executable (EXE) file. Upon execution, the EXE file loads a hidden dynamic-link library (DLL) that employs a curl command to download the final Qakbot payload. This attack chain also involves the use of DLL side loading technique, adding another layer of complexity to the attack.

The threat actor initiates this attack through malspam, sending deceptive emails containing URLs that lead to the delivery of the ZIP file. Once the user accesses the ZIP file and executes the embedded EXE file, the attack unfolds, triggering the loading of the concealed DLL. This DLL utilizes a curl command to download the final Qakbot payload from a remote server.

By incorporating DLL side loading, the threat actor creates a diversion, making it more challenging for security measures to detect the malicious activities. This advanced technique allows the malware to execute code indirectly and evade traditional detection mechanisms, adding an extra layer of sophistication to the attack.

The attack sequence follows: Malspam -> URL -> ZIP -> EXE -> DLL -> CURL -> Qakbot Payload

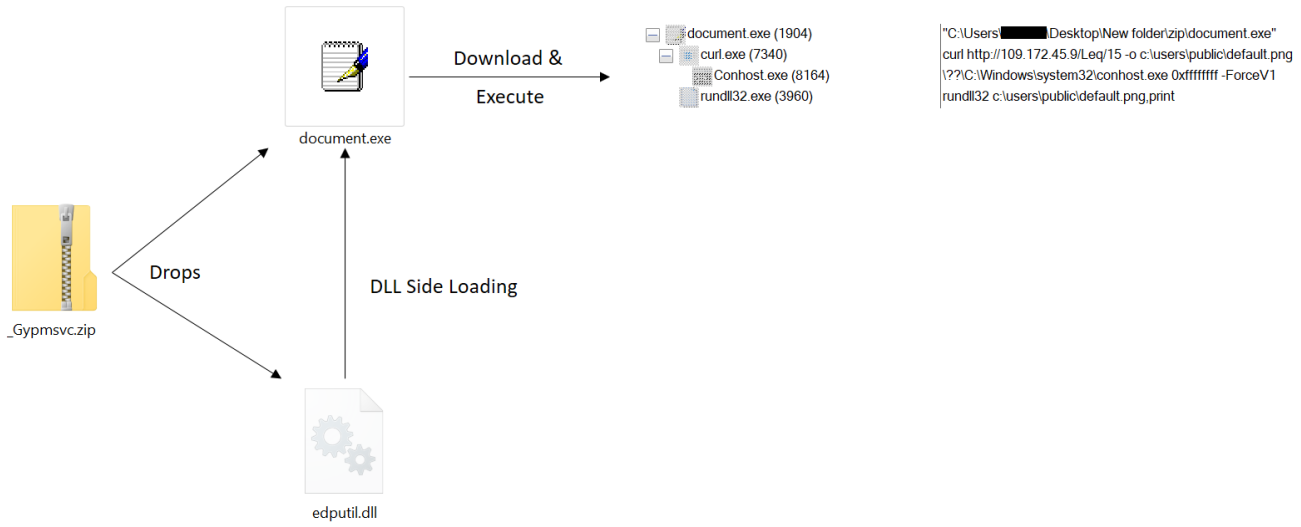

Figure 8 - Depicts Qakbot's utilization of DLL Side Loading in its attack chain.

On May 17th, several Pikabot samples were distributed using tactics, techniques, and procedures (TTPs) similar to those of Qakbot within the Zscaler Cloud. This discovery is valuable as it highlights a potential link or copycat scenario and provides insights into Pikabot malware behavior and distribution methods. The resemblance between Pikabot and Qakbot, including similarities in their behavior and internal campaign identifiers, suggests a possible connection between the two. However, there is not yet sufficient evidence to definitively link these malware families to the same threat actor.

Understanding the similarities and differences between Pikabot and Qakbot is critical for cybersecurity professionals to effectively respond to these threats. The identification of new malware variants helps security teams stay ahead of evolving attack trends, enabling them to adjust their defense strategies accordingly. By closely monitoring the behavior and distribution patterns of these malware families, security experts can enhance their threat intelligence and improve their ability to detect and mitigate such attacks in the future.

Threatlabz's ongoing technical analysis of Pikabot will provide further insights into its capabilities and potential impact on organizations. Keeping abreast of such developments and conducting thorough examinations of new malware variants is crucial for safeguarding networks, systems, and sensitive data from cyber threats. As the investigation progresses, security professionals can better assess the potential risks posed by Pikabot and formulate effective mitigation measures to protect against its infiltration and harmful activities.

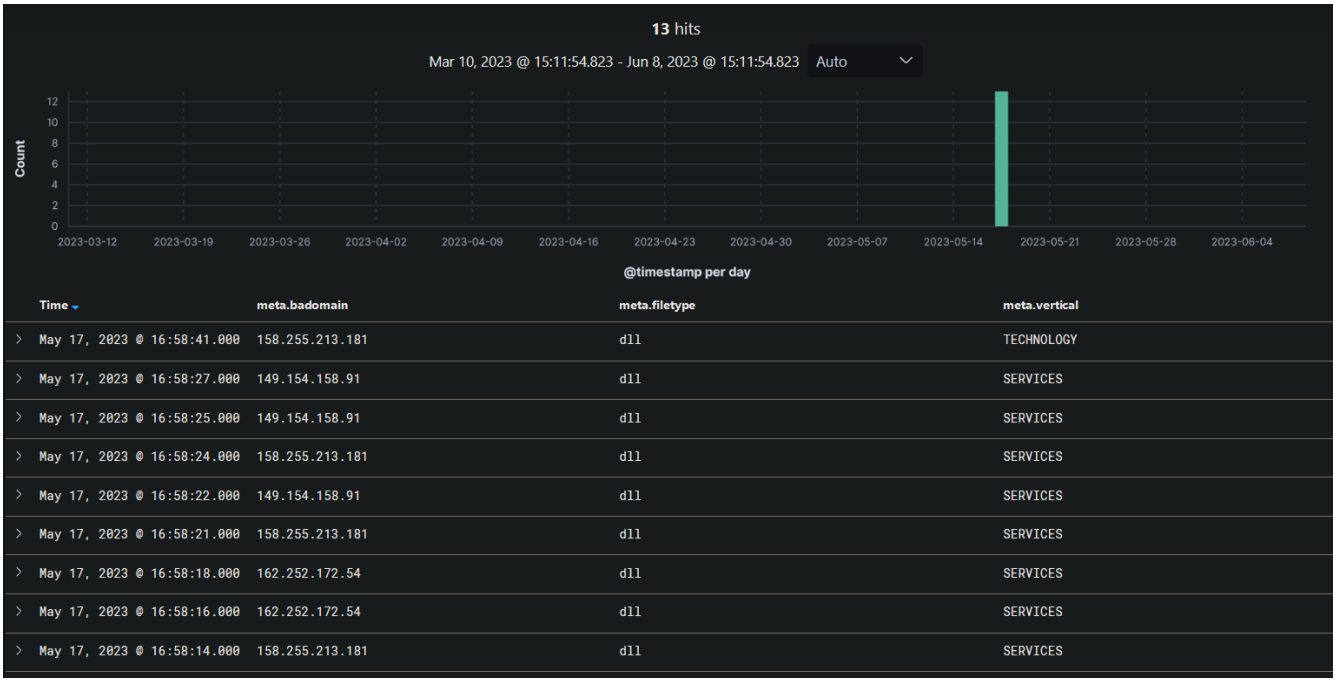

Figure 9 - Shows the distribution of Pikabot, discovered in Zscaler Cloud.

Technical Analysis Summary

The analysis of various Qakbot campaigns revealed that despite different campaign strategies, all Qakbot samples retained a consistent core. Notably, the threat actor used different compilers in each campaign, resulting in changes to the binary's opcodes while maintaining the same depack algorithm. This technique aims to evade static detection mechanisms like YARA, making it more challenging for security analysts to identify and mitigate the malware.

Following execution, the Qakbot malware checks if it is running under the Windows Defender Sandbox environment using the **GetFileAttributeW()** function. Specifically, it searches for the presence of any directory named "**C:\INTERNAL__empty**," and if detected, Qakbot terminates itself. This behavior showcases the malware's efforts to evade analysis within sandboxed environments and highlights its sophistication.

```
100010A8    397D 0C          cmp dword ptr ss:[ebp+C],edi
100010AB    0F85 D0000000    jne p.10001181
100010B1    E8 DF7F0000      call p.10009095                  Heapcra
100010B6    E8 B2830000      call p.1000946D                  allocate
100010BB    8B45 08          mov eax,dword ptr ss:[ebp+8]     [ebp+8]:L"C:\\INTERNAL\\__empty"
100010BE    50               push eax
100010BF    A3 9C0F0210      mov dword ptr ds:[10020F9C],eax  10020F9C:&"MZE"
100010C4    893D 980F0210    mov dword ptr ds:[10020F98],edi
100010CA    E8 85310100      call p.10014254
100010CF    C70424 F30E0000  mov dword ptr ss:[esp],EF3       [esp]:EntryPoint
100010D6    E8 9A7F0000      call p.10009075
100010DB    59               pop ecx
100010DC    50               push eax
100010DD    8945 08          mov dword ptr ss:[ebp+8],eax     [ebp+8]:L"C:\\INTERNAL\\__empty"
100010E0    FF15 64A10110    call dword ptr ds:[<&GetFileAttributesW> ]
100010E6    83F8 FF          cmp eax,FFFFFFFF
100010E9    8D45 08          lea eax,dword ptr ss:[ebp+8]     [ebp+8]:L"C:\\INTERNAL\\__empty"
100010EC    50               push eax
100010ED    74 0D            je p.100010FC                    if(getfileattrib2==-1)
100010EF    E8 EC8B0000      call p.10009CE0
100010F4    59               pop ecx
100010F5    33C0             xor eax,eax
100010F7    E9 80000000      jmp p.1000117C
100010FC    E8 DF8B0000      call p.10009CE0
10001101    BA 44010000      mov edx,144
```

Figure 10 - Verification of Windows Defender Sandbox execution.

Additionally, the unpacking of the Qakbot malware is relatively straightforward, utilizing the **VirtualAlloc()** API to allocate memory space and execute itself. The unpacked payload reveals two different components within the Bitmap section: **COMPONENT_07** and **COMPONENT_08**. **COMPONENT_07** contains the encrypted campaign ID, while **COMPONENT_08** contains the encrypted Qakbot command-and-control server (C2) configurations.

Qakbot samples tend to use the following resources:

- Bitmap
- RCData

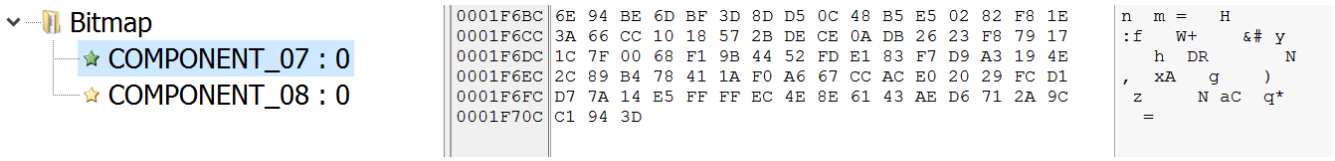

Figure 11 - Component_07: Encrypted Campaign ID.

The screenshot in Figure 11 shows the encrypted content of Component_07, which appears to contain the campaign ID used by Qakbot. This encrypted data is a crucial part of the malware's internal campaign identification process, and decrypting it may provide valuable insights into the threat actor's campaigns and targeting strategies.

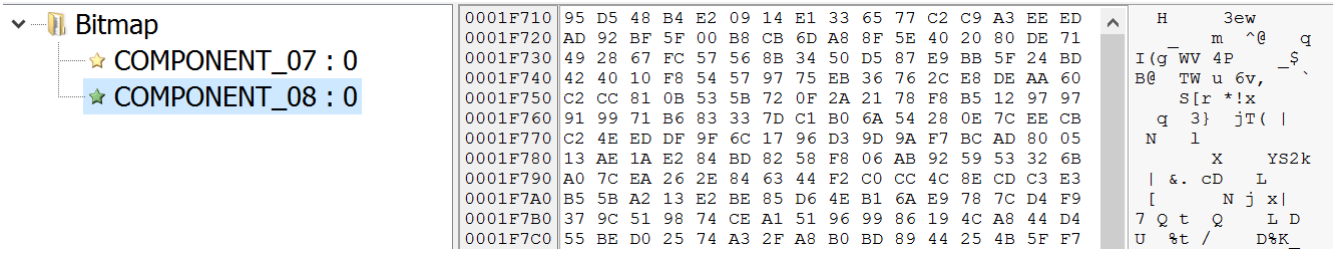

Figure 12 - Component_08: Encrypted QakBot C2 configuration.

The screenshot in Figure 12 shows the encrypted content of Component_08, which appears to hold the encrypted QakBot command-and-control (C2) server configuration information. Decrypting this component may reveal critical information about the communication channels and C2 infrastructure utilized by the QakBot malware and provide essential insights into the threat actor's operations.

Of note, Qakbot employs XOR encryption with two different offsets to encrypt significant strings. The encrypted data is strategically placed in the **.DATA** section of the unpacked payload binary file, enhancing its concealment and making it more challenging for analysts to interpret the content. The decryption loop relies on a separator byte as the termination condition, adding flexibility to the decryption process.

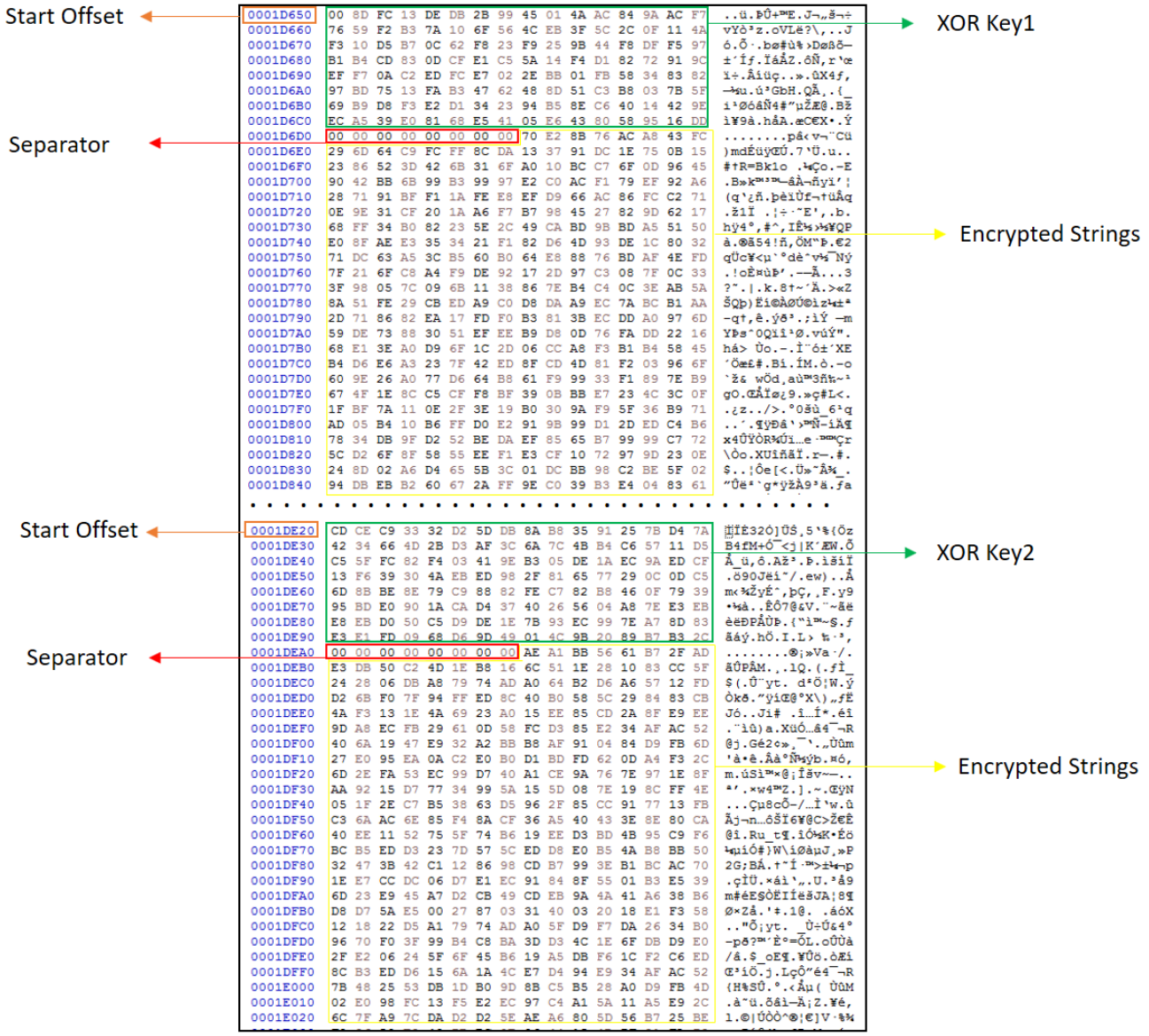

Figure 13 -Shows the encoded strings residing in the .DATA section of the Qakbot malware.

The use of encoding techniques in this section adds an extra layer of obfuscation. The decrypted strings contain critical information about Qakbot's anti-AV functionality and other malicious activities it performs. These decoded strings offer insights into the malware's behavior, showcasing the various techniques employed to avoid detection and hamper analysis efforts.

```
%s %04x.%u %04x.%u res: %s seh_test: %u consts_test: %d vmdetected: %d createprocess: %d
runas
\System32\WindowsPowerShell\v1.0\powershell.exe
net localgroup
Self check
schtasks.exe /Create /RU "NT AUTHORITY\SYSTEM" /SC ONSTART /TN %u /TR "%s" /NP /F
route print
Self check ok!
net share
Self test FAILED!!!
powershell.exe
netstat -nao
cmd
%u;%u;%u;
 /c ping.exe -n 6 127.0.0.1 &  type "%s\System32\calc.exe" > "%s"
error res='%s' err=%d len=%u
whoami /all
nltest /domain_trusts /all_trusts
SELF_TEST_1
Component_07
microsoft.com,google.com,cisco.com,oracle.com,verisign.com,broadcom.com,yahoo.com,xfinity.com,irs.gov,linkedin.com
ProfileImagePath
/t5
cmd.exe /c set
powershell.exe -encodedCommand
"%s\system32\schtasks.exe" /Create /ST %02u:%02u /RU "NT AUTHORITY\SYSTEM" /SC ONCE /tr "%s" /Z /ET %02u:%02u /tn %s
%s \"$%s = \\\"%s\\\\; & $%s\"
net view
arp -a
Microsoft
bUdiuy81gYguty@4frdRdpfko(eKmudeuMncueaN
SoNuce]ugdiB3c[doMuce2s81*uXmcvP
Self test OK.
schtasks.exe /Delete /F /TN %u
nslookup -querytype=ALL -timeout=12 _ldap._tcp.dc._msdcs.%s
Component_08
```

Next stage Decryption key

Figure 14 - Depicts the decrypted strings, along with the next stage decryption key.

These decrypted strings contain valuable information regarding Qakbot's functionalities and internal operations. The next stage decryption key is a critical component that leads to the unraveling of additional layers of encryption and provides insights into Qakbot's intricate behavior.

```
SELECT * FROM Win32_Processor
https
Content-Type: application/x-www-form-urlencoded
SELECT * FROM Win32_OperatingSystem
SOFTWARE\Wow6432Node\Microsoft\Windows Defender\Spynet
frida-winjector-helper-32.exe;frida-winjector-helper-64.exe;tcpdump.exe;windump.exe;ethereal.exe;wireshark.exe;ettercap.exe;rtsniff.exe;packetcapture.exe;capturenet
.exe;qak_proxy;dumpcap.exe;CFF
Explorer.exe;not_rundll32.exe;ProcessHacker.exe;tcpview.exe;filemon.exe;procmon.exe;idaq64.exe;loaddll32.exe;PETools.exe;ImportREC.exe;LordPE.exe;SysInspector.exe;p
roc_analyzer.exe;sysAnalyzer.exe;sniff_hit.exe;joeboxcontrol.exe;joeboxserver.exe;ResourceHacker.exe;x64dbg.exe;Fiddler.exe;sniff_hit.exe;sysAnalyzer.exe;BehaviorDu
mper.exe;processdumperx64.exe;anti-virus.EXE;sysinfoX64.exe;sctoolswrapper.exe;sysinfoX64.exe;FakeExplorer.exe;apimonitor-x86.exe;idaq.exe
SOFTWARE\Microsoft\Windows Defender\Exclusions\Paths
%SystemRoot%\System32\wermgr.exe
%SystemRoot%\System32\wextract.exe
SOFTWARE\Microsoft\Microsoft AntiMalware\SpyNet
reg.exe ADD "HKLM\%s" /f /t %s /v "%s" /d "%s"
rundll32.exe
Win32_Bios
Win32_DiskDrive
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\.\%coot\cimv2")
Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
AvastSvc.exe;aswEngSrv.exe;aswToolsSvc.exe;afwServ.exe;aswidsagent.exe;AvastUI.exe
avgcsrvx.exe;avgsvcx.exe;avgcsrva.exe
SentinelServiceHost.exe;SentinelStaticEngine.exe;SentinelAgent.exe;SentinelStaticEngineScanner.exe;SentinelUI.exe
Win32_PhysicalMemory
fmon.exe
SELECT * FROM AntiVirusProduct
image/gif
image/pjpeg
X555
C:\INTERNAL\__empty
dwengine.exe;dwarkdaemon.exe;dwwatcher.exe
SOFTWARE\Microsoft\Windows\CurrentVersion\Run
%SystemRoot%\SysWOW64\explorer.exe
SOFTWARE\Microsoft\Windows Defender\SpyNet
mcshield.exe
MBAMService.exe;mbamgui.exe
CynetEPS.exe;CynetMS.exe;CynetConsole.exe
wmic process call create 'expand "%S" "%S"'
```

Figure 15 - Decrypted strings contain Anti-AV and Anti-Analysis strings.

The SHA-1 of the hardcoded key recovered from the **.DATA** section remains static across different campaigns, and it serves as the RC4 key to decrypt encoded data in the resource section. Additionally, the SHA-1 is used for validation purposes to ensure the accuracy of the decryption process.

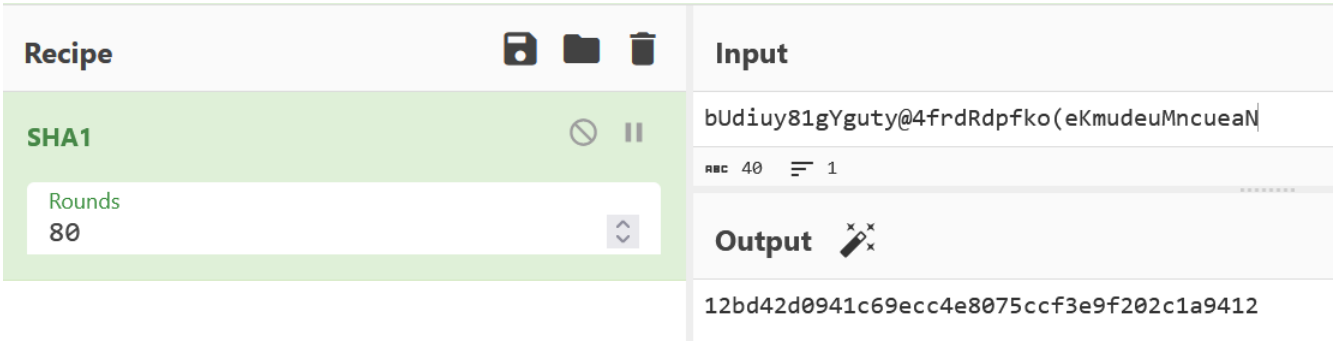

Figure 16 - SHA-1 hash of encrypted key.

Moreover, Qakbot uses SHA-1 validation to decrypt the encoded configuration present in the resource section of the unpacked binary. The decrypted configuration contains critical information such as new RC4 keys and Qakbot campaign IDs.

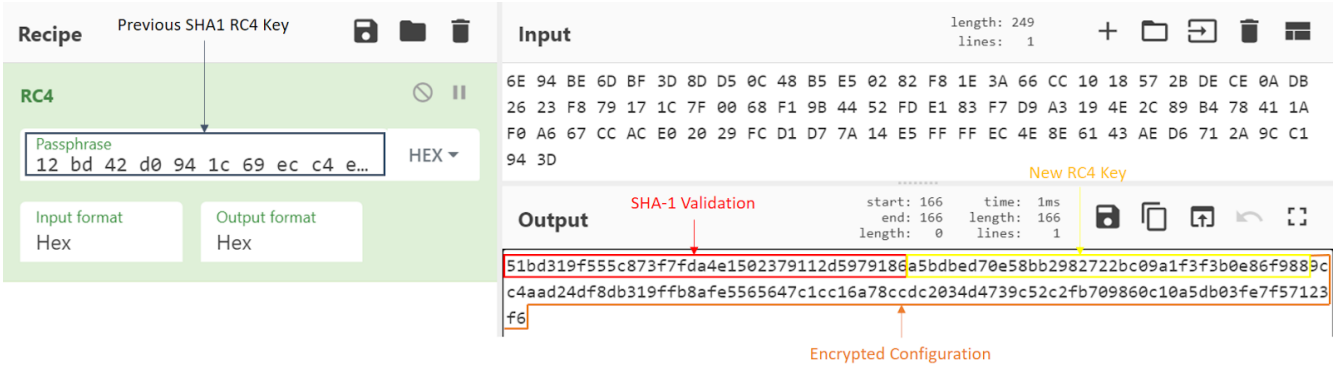

Fig.17 - SHA-1 validation + New RC4 Key + Qakbot Campaign ID

The SHA-1 validation of the New RC4 key and the Encrypted Configuration matches with the first 20 bytes obtained from the decrypted data in the previous step (Figure 17).

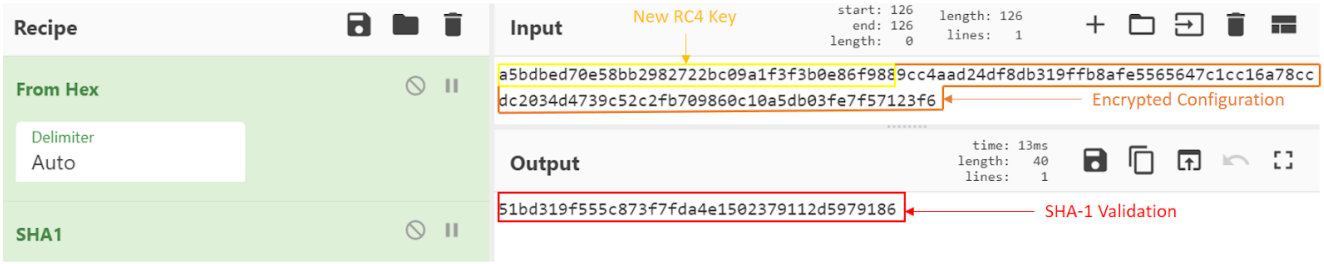

Figure 18 - SHA-1 validation.

The first 20 bytes in the data represent the SHA-1 validation, a cryptographic process used for data integrity verification. These bytes serve as a hash value that allows systems to confirm the authenticity and integrity of the data being processed.

Following the SHA-1 validation, the subsequent 20 to 40 bytes are indicative of the new encryption key. Encryption keys are essential in securing data and ensuring that only desired parties can access and interpret the encrypted information.

Beyond the 40th position in the data, we encounter the encrypted configurations. These configurations likely contain critical instructions, settings, or data that the malware utilizes during its execution and malicious activities.

This data structure encompasses essential components of the Qakbot malware's operation, encompassing validation, encryption, and critical configurations necessary for executing its malicious objectives.

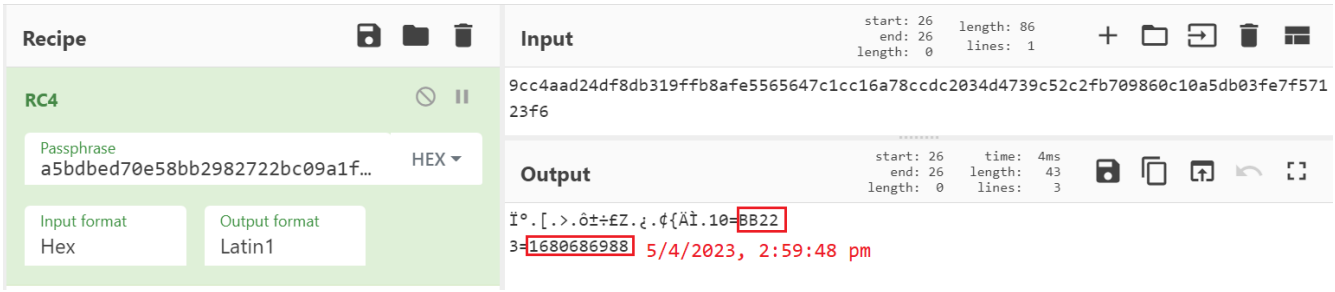

Figure 19 - Qakbot CampaignID.

Following the initial RC4 decryption process, the second round of RC4 decryption occurs on **Component_08**, as shown in Figure 12 of the resource section.

Component_08 is the encrypted section that likely contains the Qakbot command-and-control (C2) server configuration. Conducting the second RC4 decryption on this component may unveil critical information about the communication channels, domains, or IP addresses used by the malware to establish communication with its C2 infrastructure. Analyzing this decrypted data is essential in understanding the command and control infrastructure of Qakbot.

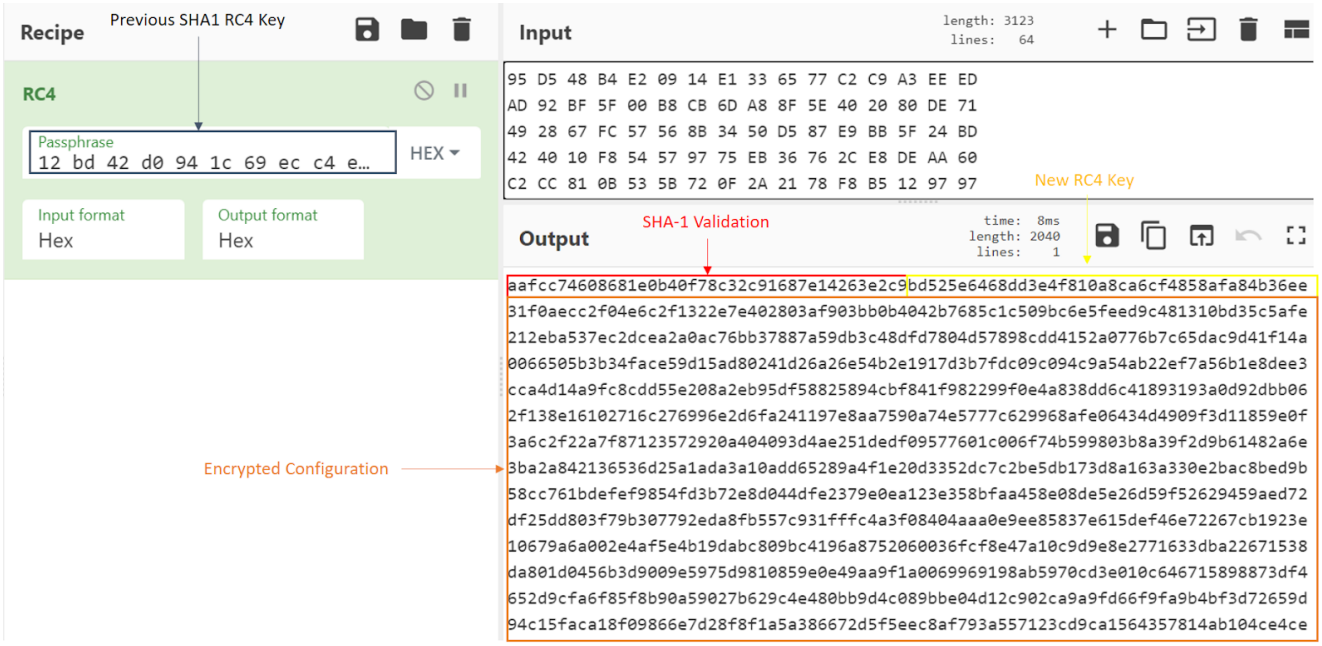

Figure 20 - SHA-1 hash validation, new RC4 key, Qakbot C2 configuration.

With the application of the new RC4 key, the decryption process enables access to the command-and-control (C2) configuration of the Qakbot malware. The decrypted configuration data is presented in hexadecimal format, with a starting separator value of "01". The subsequent four bytes are converted into decimal values

byte by byte, followed by an additional two bytes that indicate the ports used to establish connections to the C2 servers.

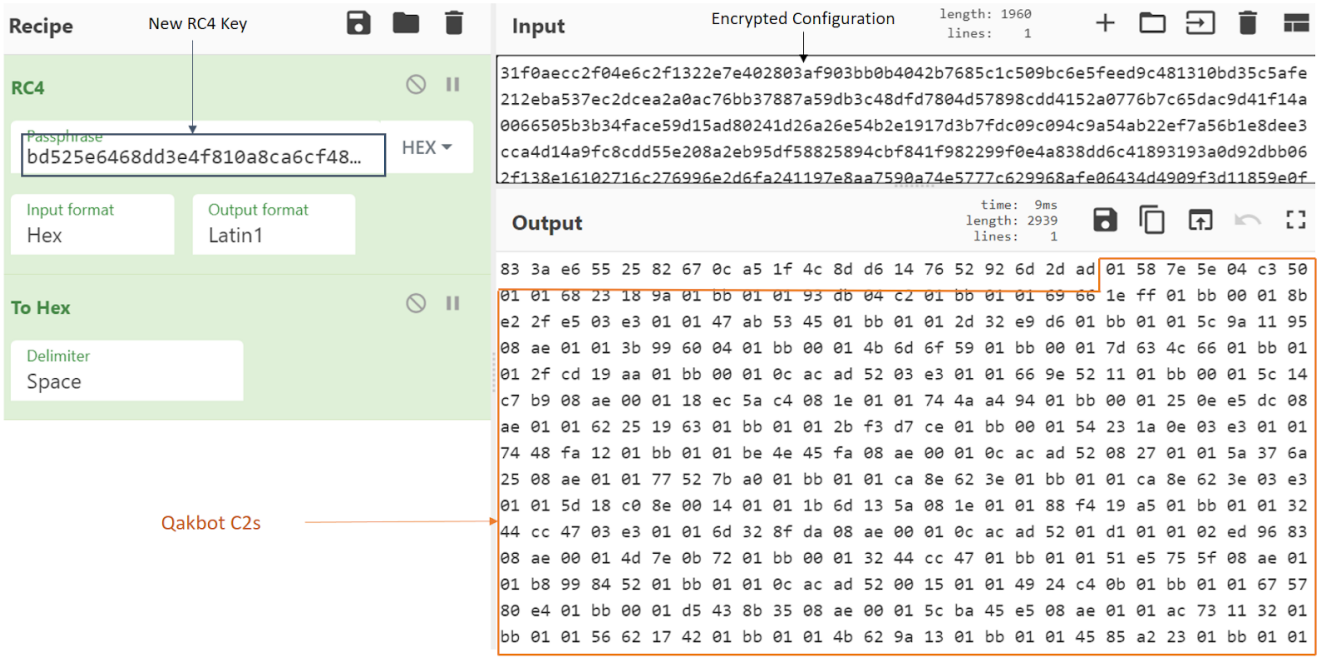

Figure 21 - Qakbot's decrypted Command-and-Control (C2) configuration.

Upon decrypting the command-and-control (C2) configuration of Qakbot, a distinct pattern emerges in how the IP addresses and ports are separated. These values are initially represented in hexadecimal format, and Qakbot converts each byte of these values into their corresponding decimal equivalents to obtain the C2 addresses.

For example:

- IP: 58 7E 5E 04 (hex) -> 88.126.94.4 (decimal)

- Port: C3 50 (hex) -> 50000 (decimal)

By converting these values from hexadecimal to decimal, Qakbot obtains the IP addresses and ports, which are essential in establishing connections with its command-and-control servers.

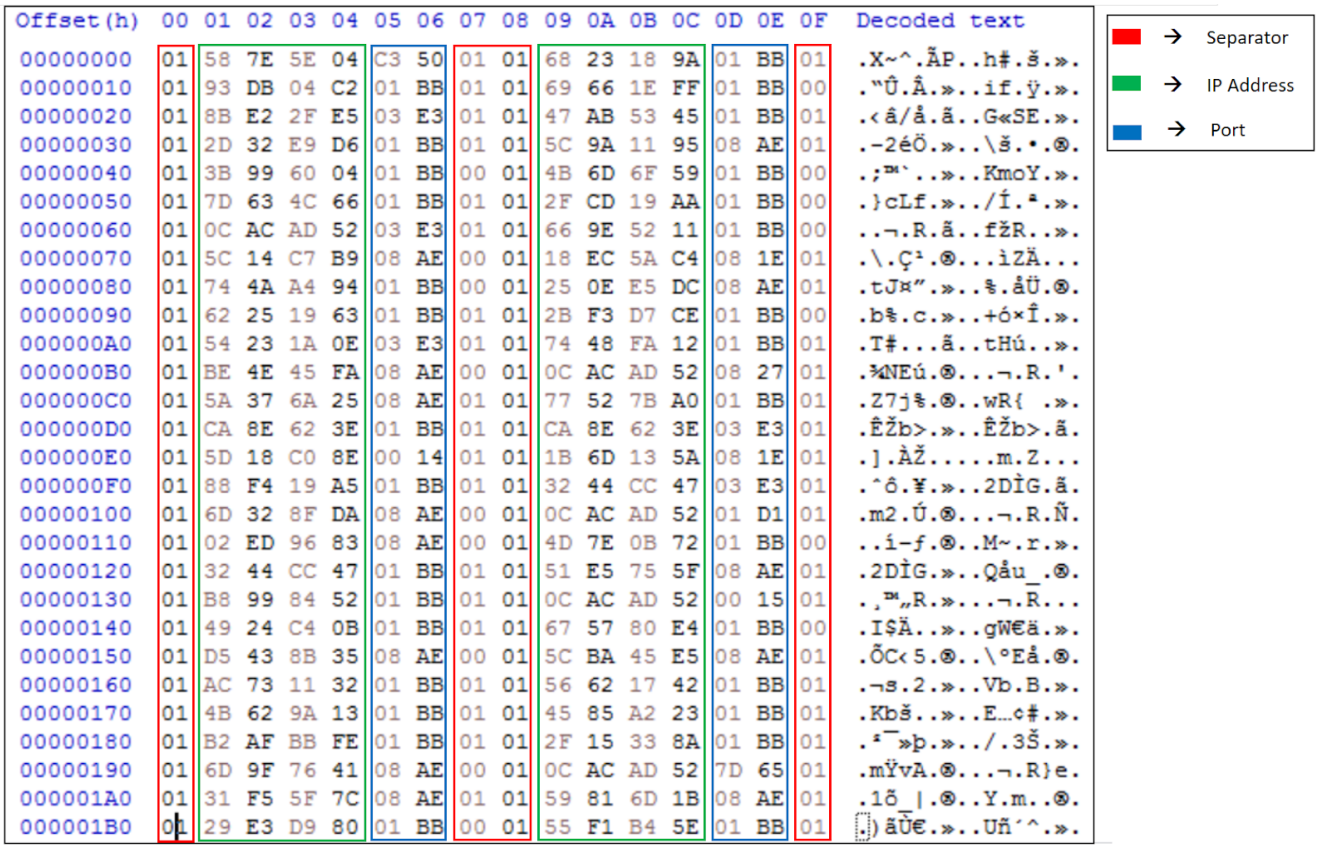

Figure 22 - Qakbot's Command-and-Control (C2) configuration.

Overall, the technical analysis provides essential insights into Qakbot's behavior, evasion techniques, and the significance of analyzing its unique components to effectively counter and mitigate this persistent threat. Understanding the malware's strategies empowers security professionals to develop robust defense measures and stay proactive in safeguarding networks and systems from Qakbot and other evolving malware.

Network Analysis

Conducting a thorough examination of the Qakbot Command and Control (C2) infrastructure, we observed the top five countries where Qakbot C2s are most active. These countries include the United States (US), Great Britain (GB), India (IN), Canada (CA), and France (FR).

This analysis highlights the global reach and widespread distribution of Qakbot's C2 servers, indicating the significant geographic presence of the malware's command centers. Understanding the distribution of C2 servers in different countries is crucial for devising targeted defense strategies and collaborating with international cybersecurity partners to combat the threat effectively.

country	total_hostnames	lat	long
US	87	39.783730	-100.445882
GB	25	54.702355	-3.276575
IN	23	22.351115	78.667743
CA	12	61.066692	-107.991707
FR	11	46.603354	1.888333

Table 1 - Displays the top 5 countries where Qakbot Command and Control (C2) servers are most active.

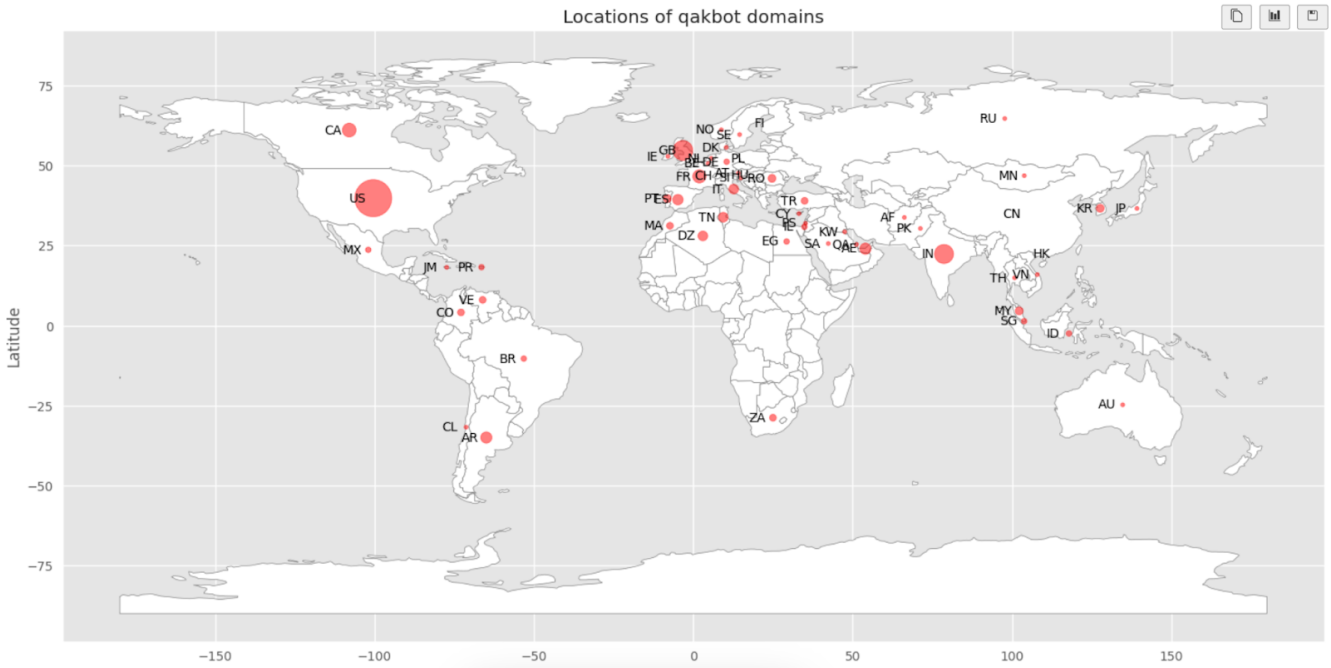

Figure 23 - Showcases the distribution of Qakbot Command and Control (C2) servers.

Upon further analysis of the Command and Control (C2) servers, we observed that the transaction count of Qakbot C2s was significantly higher during March and April, than at the beginning of the year. This indicates a surge in the malware's activities during that period, and it may suggest that the threat actor(s) behind Qakbot were particularly active in executing campaigns during this timeframe.

datetime	hostname	count	reqsize	respsize
2023-01-19	16	575	0.560	14.352
2023-02-02	14	610	0.000	0.000
2023-03-13	11	900	1908.432	1747.095
2023-04-06	53	1132	6695.416	2178.435
2023-05-15	42	77	8.319	559.605

Table 2 - Displays the Qakbot transaction count month over month from January to May of 2023.

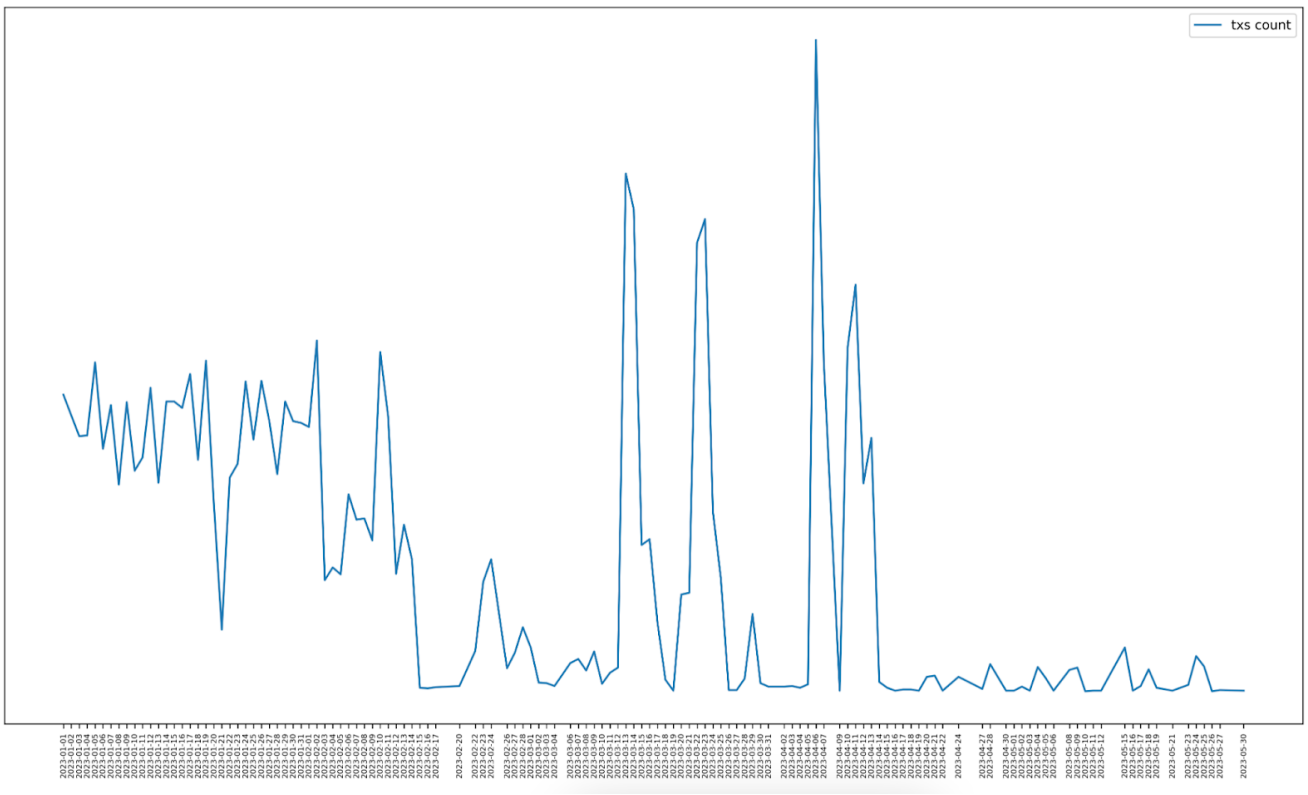

Figure 24 - Illustrates spikes in the transaction count of Qakbot Command and Control (C2) activity by date.

In March 2023, Germany experienced a significant surge in Qakbot Command and Control (C2) activity. During this period, major Qakbot C2s from the United States (US), Netherlands (NL), and France (FR) were directed towards Germany, indicating a targeted campaign against the country. A similar trend was observed in April 2023, although with a reduced volume of data transferred compared to March. The data suggests a concentrated effort by threat actors to target Germany during these months, potentially signaling specific motivations or objectives in that region.

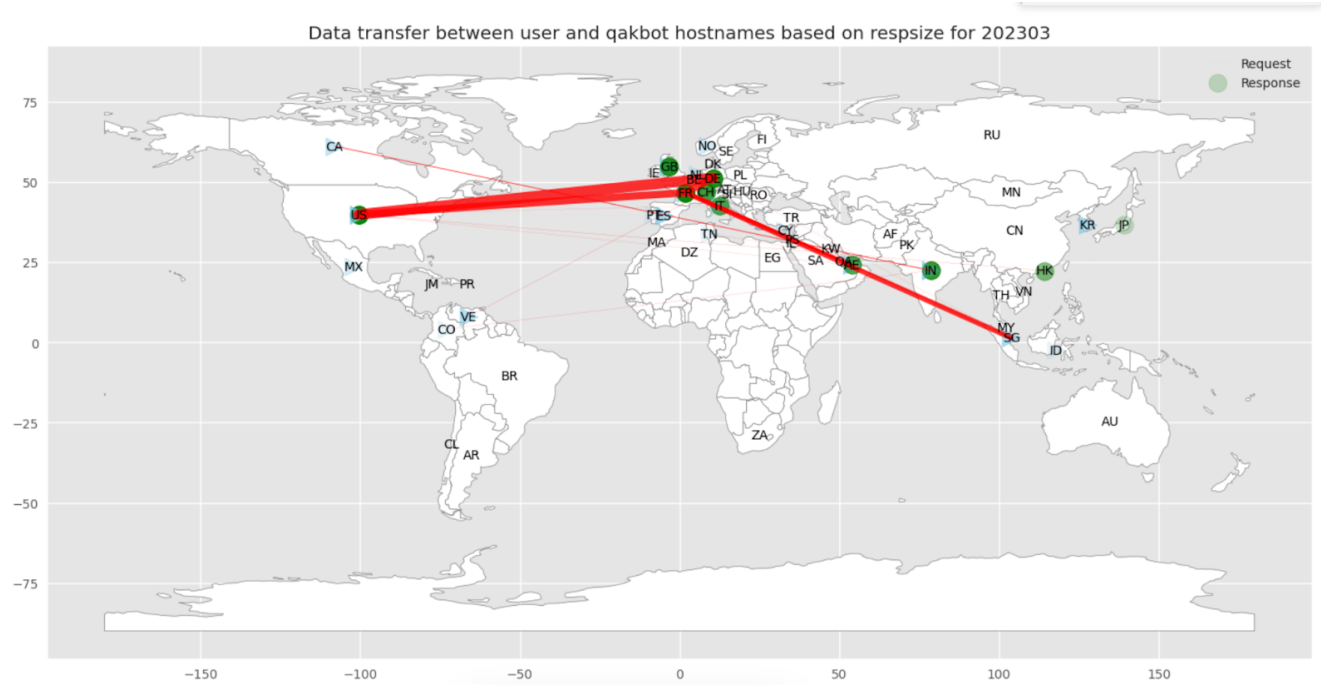

Figure 25 - Illustrates the Qakbot Command and Control (C2) activity specifically targeting Germany in March 2023.

In April 2023, our observations revealed noteworthy Command and Control (C2) activity originating from Argentina (AR) and targeting the United States (US) with substantial data transfer. Additionally, Qakbot C2s in Italy (IT) were observed targeting Brazil (BR). These activities indicate an interconnected network of C2 servers and highlight the global nature of Qakbot's operations, with specific regions targeting each other for potential malicious activities.

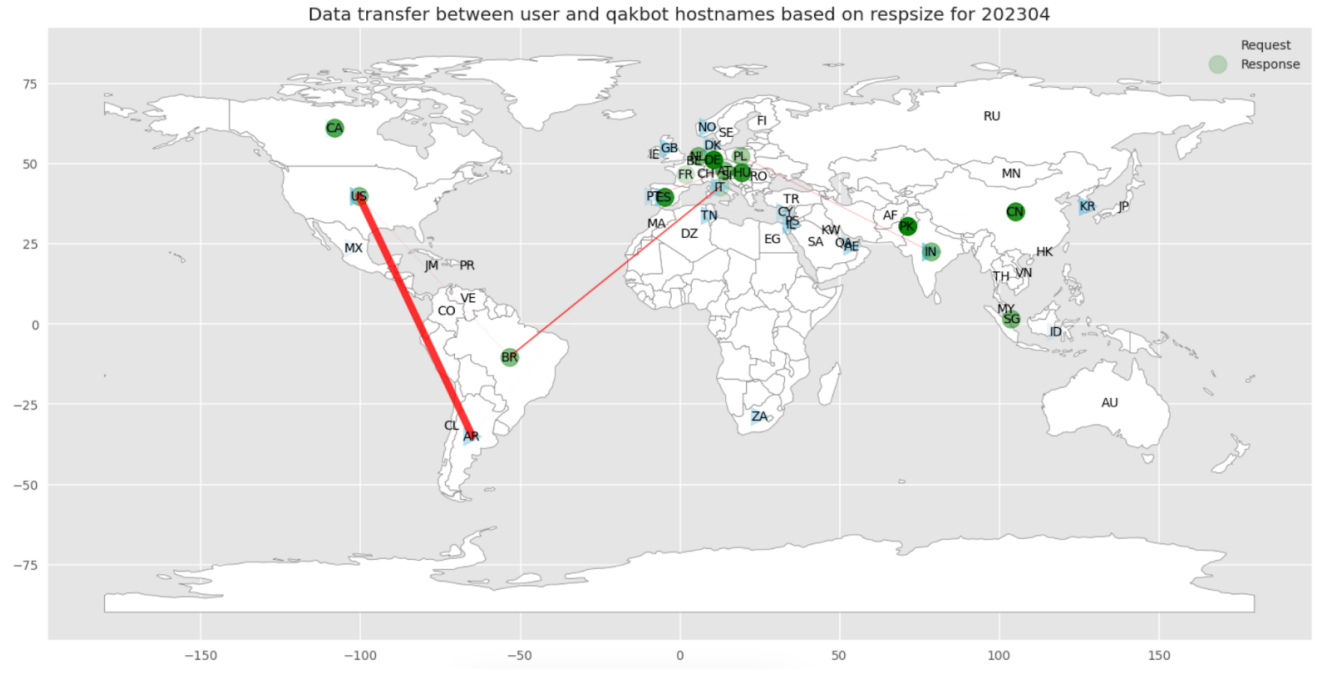

Figure 26 - Depicts the Qakbot Command and Control (C2) activity during April 2023.

Conclusion

In conclusion, Qakbot is a highly sophisticated banking trojan malware, strategically targeting businesses across different countries. This elusive threat employs multiple file formats and obfuscation methods within its attack chain, enabling it to evade detection from conventional antivirus engines. Operating through a phishing campaign, Qakbot continuously adapts to new distribution mechanisms to more effectively infect users.

Through its experimentation with diverse attack chains, it becomes evident that the Threat Actor behind Qakbot is continuously refining its strategies. However, after June, a significant drop in Qakbot campaigns is observed, suggesting a possible pause in their activities. Zscaler's Threat Labs team extensively analyzed the behavior of various files associated with Qakbot, utilizing the MITRE ATT&CK framework to assess threat scores and triggered techniques. The team remains vigilant, continuously monitoring the campaign, and is prepared to unveil any new findings they may discover.

To combat such threats effectively, organizations must remain vigilant and adopt best practices, including implementing multi-layered security defenses and conducting security awareness training. By staying proactive and collaborative, the cybersecurity community can thwart Qakbot's relentless pursuit of infiltrating and compromising systems, ensuring a safer digital landscape for individuals and enterprises worldwide.

Zscaler Sandbox Coverage

During the investigation of this campaign, Zscaler Sandbox played a crucial role in analyzing the behavior of various files. Through this sandbox analysis, the threat scores and specific MITRE ATT&CK techniques triggered were identified, as illustrated in the screenshots provided below. This comprehensive approach empowers cybersecurity professionals with critical insights into the malware's behavior, enabling them to effectively detect and counter the threats posed by this campaign.

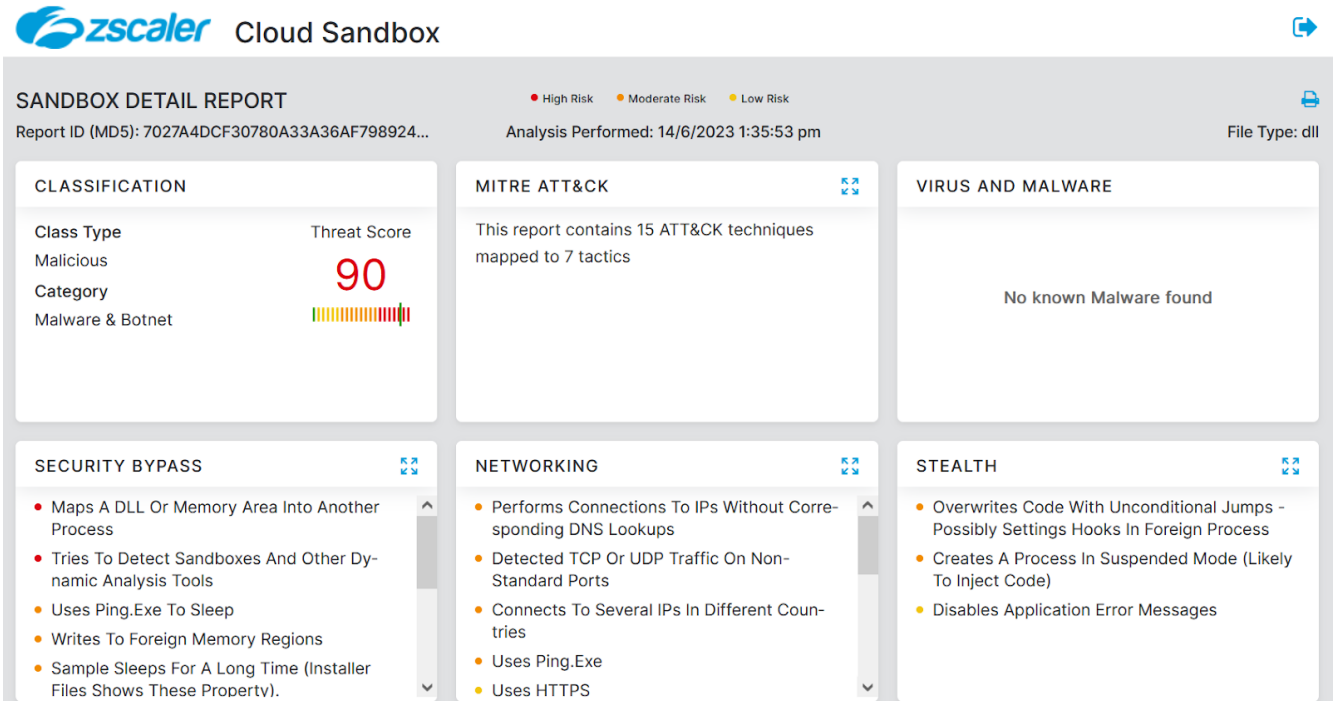

Figure 27 - Zscaler Sandbox report detecting and analyzing recent Qakbot malware campaign.

Zscaler's multilayered cloud security platform detects payloads with following threat names:

- Win32.Banker.Qakbot

MITRE ATT&CK Techniques:

Tactic	Technique ID	Technique Name
Initial Access	T1566	Phishing
Execution	T1204	User Execution
	T1059	Command and Scripting Interpreter
	T1047	Windows Management Instrumentation
Persistence	T1053.005	Scheduled Task
	T1547.001	Registry Run Keys / Startup Folder
Privilege Escalation	T1053.005	Scheduled Task
Defense Evasion	T1027	Obfuscated Files or Information
	T1070.004	File Deletion
	T1112	Modify Registry
	T1202	Indirect Command Execution
	T1574.002	DLL Side-Loading
	T1574.001	DLL Search Order Hijacking
	T1564.001	Hidden Files and Directories
	T1055	Process Injection
	T1218	System Binary Proxy Execution
Credential Access	T1003	OS Credential Dumping
	T1555.003	Credentials from Web Browsers
Discovery	T1016	System Network Configuration Discovery
Command and Control	T1071	Application Layer Protocol
	T1095	Non-Application Layer Protocol

Indicators of Compromise (IoCs):

Case Study 1 - March 2023

Description	MD5	Network

Malicious PDF Download JS File	c986136d713f71449ad8ba970379d306	85.239.52[.]29/ONT[.]php
Obfuscated JS file download Qakbot	3607ad95e33dd12803af676597df5c6a	45.66.248[.]9/qBSTwc/aw
Qakbot Payload	770453c5d3ed689a451d55e947764742	-
Description	**MD5**	**Network**
Malicious HTML file download Zip file	755a25e36cbf87b7e4415de2fdf0f9e3	https[:]//jbdata.com.ng/uq/uq[.]php?88748 https[:]//superspeedtransports.com/qs/qs.php?59697 https[:]//aadilmehmood.com/oab/oab[.]php?24149
Downloaded Zip File	1a90b0c2129b8a552b6ec751ef1e6caa	-
Extracted JS File	e2a21a2a7f5d2d85c0bcda95d6d0fc03	https[:]//azarmadar[.]com/aUqL/120
Qakbot Payload	74ee45a7dc4ca40eaaf817dc5959328d	-
Description	**MD5**	**Network**
Malicious PDF	dd27c04bc998f69467c2c81c53a111ab	http[:]//gurtek.com[.]tr/exi/exi.php
Downloaded Zip File	789e3789de0eb630000adb1a2ed27d7e	-
Extracted WSF File	e94c5f36ec0cccccb231e1cd04f2a646	https[:]//graficalevi.com[.]br/0p6P/vLSyX
Qakbot Payload	19c1526182fe5ed0f1abfafc98d84df9	-
Description	**MD5**	**Network**
Malicious PDF	cccda4837024a71fa74ceb420b5e854e	https[:]//iquodigital[.]com/eps/delectusfuga.php
Download Zip	2bc1cbc8c8f54245ca0fefb49c229f77	-

Extracted HTA File	2394742a2c6fa05327cf1d48767af727	https[:]//zainco[.]net/OdOU/5k4ll56eOFo
Qakbot Payload	fb5ca6825e52d72a2010c8474ddaaa41	-

Description	**MD5**	**Network**
Zip File	91fb1dcf5a6222262fd7fa77019bb1e4	-
XLL File	68781578b0b58e21177c7b71f9b85567	-
Qakbot	ff58f9cf0740aead678d9e36c0782894	-

Case Study 2 - April 2023

Description	**MD5**	**Network**
PDF File	2342ee9c7520abef3700b0fddf825c71	http[:]//eaglewingsuae[.]com/wicd/643d2215dacb3.zip
Zip File	03c8cd94f624ae6074c8facb973d4b9d	-
WSF File	65f256e4ce4013742f2b59d869b6c663	http[:]//77.91.100[.]135/aSxBaqnfj98.dat
Qakbot	4deae2c9f1f455670f2e091ce7e0b4e1	-

Description	**MD5**	**Network**
OneNote File	77079f381ac044ad7a3df18607657f74	-
MSI File	8056b3bafd82ce7e6156f1b3f314db52	-
Cleanup PS1 File	e1031ce77dde7a368159a9dd0ed7e6d4	-
WSF File	cb93c679ed14fe409df9a6cb564e488f	https[:]//logswalker[.]com/aF8HY9p/2
Qakbot	ce0d0ef75f3d7da7ba434a2017905132	-

Case Study 3 - May 2023

Description	**MD5**	**Network**

PDF File	f42544fe0db583e4b836e4b8cfc52802	https[:]//inspiratour[.]co[.]id/tsopexfzrf/tsopexfzrf.zip
ZIP File	842fb152664671ca137b8ae390900fa6	-
WSF File	934feee5657b08faec80a29cd2a77acc	http[:]//45.155.37[.]101/a2nZbs476.dat http[:]//149.102.225[.]18/a2nZbs476.dat http[:]//207.148.14[.]105/a2nZbs476.dat http[:]//5.42.221[.]144/a2nZbs476.dat
Qakbot	2b652290e80db5de823a915145eff417	-

Description	**MD5**	**Network**
ZIP File	55027a65b1889b0642dbce8f39f4ba74	-
Side Loading DLL	48f68450df1ca26e3fb1d7c07d0fd836	http[:]//109.172.45[.]9/Leq/15
Qakbot	fce88b20bceebd0bfed68131820efab6	-