Buzzing in the Background: BumbleBee, a New Modular Backdoor Evolved From BookWorm

9/2/2022

Updated on Sept. 2, 2022, at 9:55 p.m. ET to clarify the difference between this Bumblebee malware and the Bumblebee ransomware loader.

Introduction

In March 2021, we investigated a backdoor with a unique modular architecture and called it BumbleBee due to a string embedded in the malware. Its type of modular framework has made our static analysis more challenging because it required us to first rebuild its structure or use dynamic analysis to understand its functionality and behavior.

Our analysis found that BumbleBee only had little malicious code in its payload, and what it does on the surface is track keys and clipboard content. However, further investigation revealed a controller application that expands the malware's capabilities.

This type of backdoor is similar to another of its kind called BookWorm, in which it can be inferred that BumbleBee is a refactored version of BookWorm. At the time of writing, BumbleBee has only been deployed in Taiwan; together with its use of Simplified Chinese as the language for its user interface, this malware can be suspected to be deployed by malicious Chinese actors. This blog will tackle BumbleBee's capabilities and our analysis of this backdoor. It's important to note that this Bumblebee malware family is different from the Bumblebee loader, a loader malware that is used by ransomware groups to drop backdoors to gain access to corporate networks.

BumbleBee – a refactored modular backdoor

BumbleBee is a modular backdoor that comprises two applications, a server and a client application (a master and slaver application, respectively in the malware's jargon). Once the client application is deployed on the target computer (these are commonly local government devices), threat actors can control the machine using the server module. Let us take a deeper look into this backdoor.

Layered deployment – client application

We have encountered the client application in a security breach incident. Its unique "layer-in-layer" architecture caught our attention. The module has a self-extracted file that contains three main parts: a legitimate executable (*XcrSvr.exe*), side-loaded DLL (*XecureIO_v20.dll*) and the shellcode binary file (ore) in the file system to execute the legitimate executable.

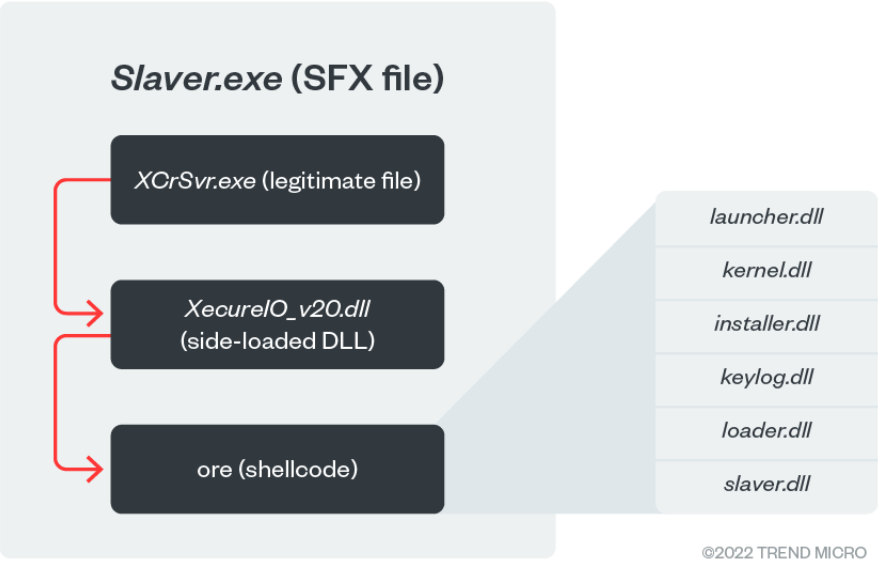


Figure 1. Architecture of BumbleBee

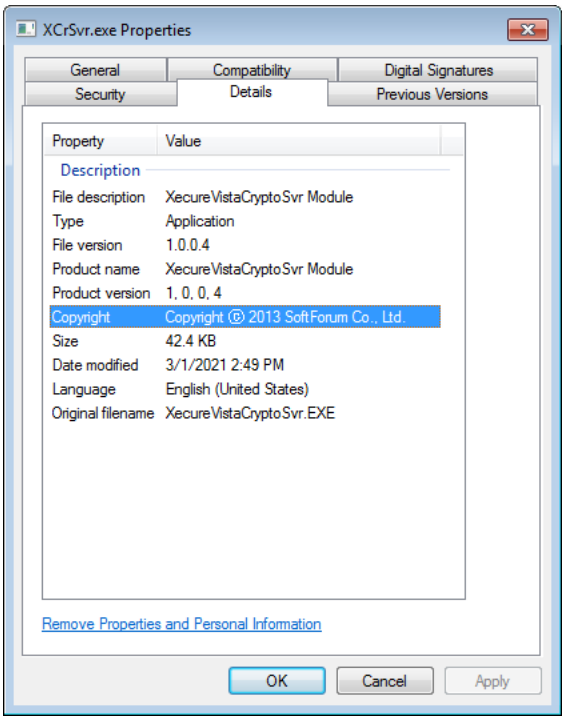


Figure 2. Metadata of XcrSvr.exe

XCrSvr.exe is the executable in the XecureVistaCryptoSvr module developed by SoftForum. This file is exploited to launch the side-loaded DLL, *XecureIO_v20.dll*, which will work as the next-stage loader that executes the shellcode "ore," which is the main component in this backdoor. This shellcode contains multiple modules of its own (shown in Table 1). Each module has corresponding 32-bit and 64-bit versions of binaries in the shellcode except for *launcher.dll*.

Name	Description
launcher.dll	The first-stage launcher that loads all the subsequent modules. It decrypts a list of modules in memory and executes each in order.
kernel.dll	The utility component that controls all the other modules.
installer.dll	The module used to install components in the compromised machine.
keylog.dll	The keylog component monitors the keystrokes and clipboard content of the victim, and records actions from the victim such as running a process, entering a password, and getting the text of a window. The stolen data will then be run through a XOR logic gate with a two-byte key 0xF29D and saved under %temp%\kb\[UserName]\. The timestamp will be used as the file name.
loader.dll	The module that reads the shellcode.
slaver.dll	The main module that interacts with the other methods once the backdoor is launched.

Table 1. BumbleBee's modules

If a victim is compromised for the first time, *launcher.dll* loads and launches all the other modules. The installer modules will be responsible for the installation and establishing persistence on the compromised machine via the following steps:

1. Drop a copy of the XecureIO_v20.dll in %APPDATA%\LOCAL\TEMP folder.
2. Encrypt original shellcode file (to be a "bin" file) and path information (to be a "path" file) by using RC4 algorithm (key is the value of "ProductID" from "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Registration")
3. Drop bpu.dll (used to bypass UAC) and launched by rundll32.exe.
4. Establish persistence on compromised machine.
5. Delete the original SFX file.

Notably, as *XecureIO_v20.dll* is loaded by *XcrSvr.exe*, it will check if the parent process is "*XcrSvr.exe*." If so, it will patch the entry point of *XcrSvr.exe* with a long jump instruction to direct execution flow to the malicious code.

```
result = GetModuleHandleW(0);
base = (PIMAGE_DOS_HEADER)result;
if ( *(_WORD *)result != 'ZM' )
  return result;
v5 = base->e_lfanew;
nt_headers_1 = *(PIMAGE_NT_HEADERS *)((char *)&base->e_magic + v5);
result = (HMODULE)((char *)base + v5);
nt_headers = (PIMAGE_NT_HEADERS)((char *)base + v5);
if ( nt_headers_1 != (PIMAGE_NT_HEADERS)'EP' )
  return result;
entrypoint = (char *)base + nt_headers->OptionalHeader.AddressOfEntryPoint;// get entrypoint VA
result = (HMODULE)VirtualProtect(
                    (char *)base + nt_headers->OptionalHeader.AddressOfEntryPoint,
                    0x10u,
                    0x40u,
                    &flOldProtect);
base = (PIMAGE_DOS_HEADER)result;
if ( !result )
  return result;
v3 = (char *)load_malicious_shellcode - (char *)entrypoint - 5;
entrypoint[1] = (char *)load_malicious_shellcode - (char *)entrypoint - 5;// patch entrypoint
entrypoint[4] = HIBYTE(v3);
*entrypoint = 0xE9;                             // call
entrypoint[2] = (unsigned __int16)((char *)load_malicious_shellcode - (char *)entrypoint - 5) >> 8;
entrypoint[3] = (unsigned int)((char *)load_malicious_shellcode - (char *)entrypoint - 5) >> 16;
return (HMODULE)VirtualProtect(entrypoint, 0x10u, flOldProtect, &flOldProtect);
```

Figure 3. XecureIO_v20.dll hooks its parent process' entry point

```
00401000    55              push ebp                                    EntryPoint
00401001    8BEC            mov ebp,esp
00401003    83EC 44         sub esp,44
00401006    56              push esi
00401007    FF15 DC504000   call dword ptr ds:[<&GetCommandLineA>]
0040100D    8BF0            mov esi,eax                                 eax:&L"XCrSvr.exe"
```

Figure 4. The original entry point

```
00401000    E9 CB0CC00F     jmp xecureio_v20.10001CD0                   EntryPoint
00401005    44              inc esp
00401006    56              push esi
00401007    FF15 DC504000   call dword ptr ds:[<&GetCommandLineA>]
0040100D    8BF0            mov esi,eax
```

Figure 5. The patched entry point

Based on our analysis, we think the reason is that the malicious code embedded in XecureIO_v20 will not run if it followed the normal execution flow of *XCrSvr.exe*. Hence, once *XecureIO_v20.dll* is loaded by *XCrSvr.exe*, it will patch the entry point of *XCrSvr.exe* and jump to the address of the malicious code to make sure the code can be executed properly.

After the client is installed and the persistence is established, the loader, *XecureIO_v20.dll*, will retrieve the value of "ProductID" from the registry key "*HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Registration*" and use it as the key to decrypt the encrypted payload (the file "bin") dropped in the first installation. Using the information on the compromised machine as a key to encrypt the payload makes it much more difficult for analysts to decrypt and debug the malware in the analysis environment.

File name	Description
path	An RC4-encrypted path string used to find the location of next-stage shellcode. It could be a file path or a registry path starting with HKLM or HKCU.
bin	The next-stage RC4-encrypted shellcode payload.

Table 2. Payload file names

Expanded control – server application

Due to BumbleBee's complex client application, it took some time for us to fully analyze its functionality. While doing so, we ran across the server application of the malware that acts as a controller. This provided us with further understanding on how BumbleBee works.

As the client application is running on the infected device, it will communicate with the server application and show the information of the machine it is in. Details, such as computer name, external IP address, geographic location, OS, CPU, and memory, are collected by the client application.

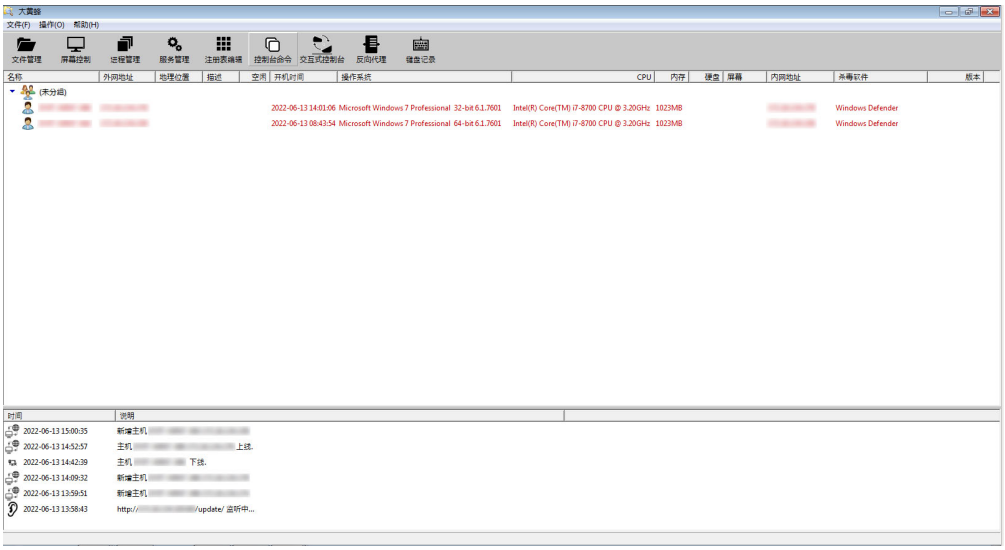


Figure 6. Connection established

Figure 7. Built-in options in server application

Based on the options in the server application shown in Figure 7, we can determine that it supports the following functions for controlling the compromised machine:

Functions	Description
文件管理 (File management)	Upload/download/delete/list files from the victim's environment
屏幕控制 (Remote desktop control)	Control the victim's desktop remotely
进程管理 (Process management)	List and manage running processes with the image names, current folder, process id and parent process id
服务管理 (Service management)	List and manage current services status
注册表编辑 (Registry editor)	List and manage the victim's registry key
控制台命令 (Command shell)	Execute the command shell
交互式控制台 (Interactive console)	Execute the command shell
反向代理 (Reverse proxy)	Reverse proxy to help expose a local server behind a NAT or firewall to the internet
键盘记录 (Keylogger)	Log keystrokes and clipboard contents

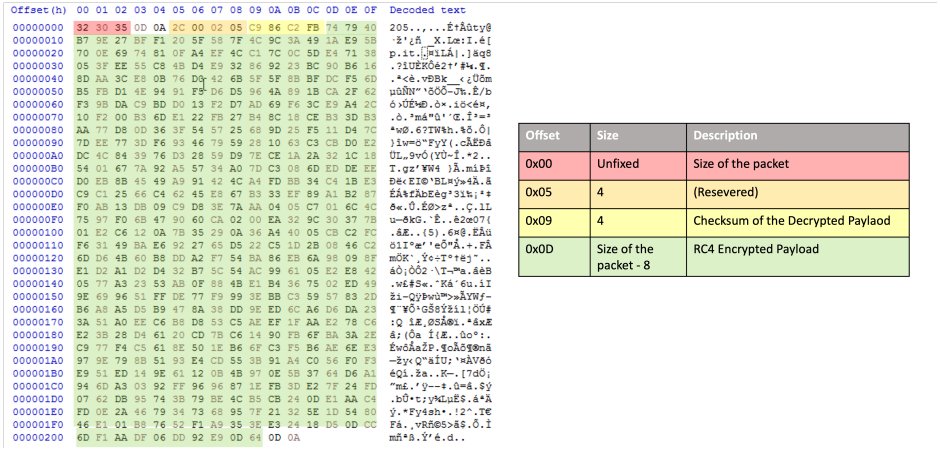
Table 3. Supported functions

BumbleBee's modular framework allowed it to embed a small amount of malicious code that involves stealing keystrokes and clipboard content in the client's shellcode. However, it could expand its capabilities through its server application by loading additional modules. This design proves that BumbleBee is flexible, allowing its developers to focus on the development of additional modules instead of having to rebuild the malware itself. Its structure could also reduce the risk of exposing itself to analysts and their own modules for comparison.

Network communication

BumbleBee communicates over the HTTP protocol. It first creates an HTTP request that acts as a network beacon to notify the command and control (C&C) server. The POST request with the following URL, *http://<C&C server>/update/*, is the initial network beacon. The client application will send information of the compromised machine, which is encrypted by RC4 (see Figure 8 and Figure 9) once the first connection is established successfully. All other communication traffic, except for the victim information, are encrypted between server and client applications using the RC4 and compressed by LZO (Lempel–Ziv–Oberhumer) algorithm.

To make sure the received payload is correct, BumbleBee adopts a CRC32 checksum with reversed-presentation mode to verify the received data. For the CRC32 calculation, a self-defined value, "20200105" is used as the initial value (typically, the value is 0xffffffff) for checksum calculation.



Offset	Size	Description
0x00	Unfixed	Size of the packet
0x05	4	(Resevered)
0x09	4	Checksum of the Decrypted Paylaod
0x0D	Size of the packet - 8	RC4 Encrypted Payload

Figure 8. Encrypted information of the compromised machine

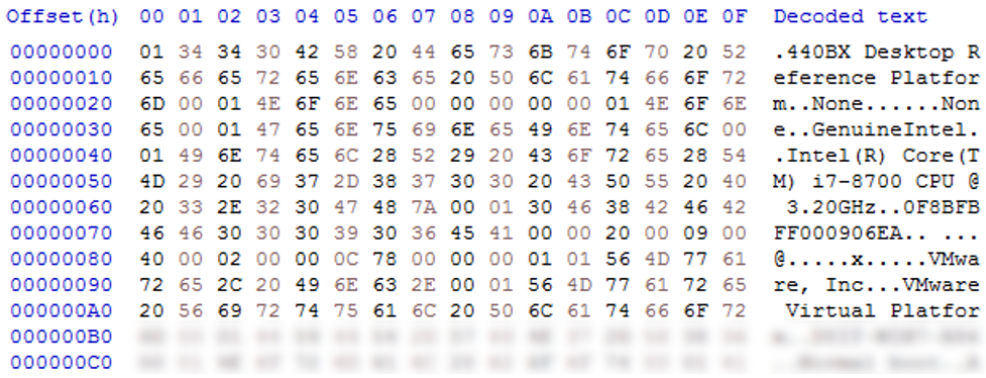


Figure 9. Decrypted information (by RC4)

Persistence

During the investigation, we found that BumbleBee adopted several techniques for persistence. It will use different techniques depending on the configuration. Here are the techniques adopted by the BumbleBee sample we found:

- Abuse registry run key to repeatedly execute the malware once system boot
- Create Windows services to repeatedly execute malicious payloads
- Use Windows logon scripts automatically executed at logon initialization to establish persistence via adding a Registry key *HKEY_CURRENT_USER\Environment* "*UserInitMprLogonScript*"

Attribution

Due to the unique modular structure and installation procedures, we started to work on a literature review to clarify if it is an exclusive tool used by a certain threat actor. We found a similar backdoor, “BookWorm,” revealed by Palo Alto in 2015. They share the following features:

1. Both are self-extracted files and abuse legitimate executables to load self-made malware.
2. Both use the same registry value as RC4 encryption key to encrypt their payload.
3. Both use modular architecture in the conception of the backdoor.
4. Both appeared in Southeast Asia, targeting local government-related organizations (similar victimology).
5. Both use RC4 and LZO algorithms in C&C communications (similar network protocol).

We think BumbleBee is likely to be the refactored BookWorm backdoor. They have similar tactics, techniques, and procedures (TTPs), unique encryption approach, and similar target sectors. According to the language (Simplified Chinese) shown in server application, we suspect that the origins and developers of BumbleBee may be in China and of Chinese descent.

Conclusion

Since BumbleBee and Bookworm share the same features, BumbleBee is likely a refactored form of the latter. Focusing on Asian local government targets, all signs point to a suspect linked to a Chinese hacker group.

BumbleBee, being a modular framework, is not only flexible but sophisticated as it will require analysts to investigate its structure and behavior. Another aspect of having a modular framework is that they can just keep developing additional modules since it can easily be integrated with the current version of said malware.

With its modular capabilities, the threat may deploy additional modules that may prove dangerous. Thus, an advanced layer of protection and quick detection is needed to prevent the backdoor from taking root in the system. Trend Micro Vision One™ offers both within different entry points of a backdoor.

IOCs

Trojan.Win32.MULTICOM.ZTIC

f8809c6c56d2a0f8a08fe181614e6d9488eeb6983f044f2e6a8fa6a617ef2475	*slaver.exe*

Trojan.Win32.REGLOAD.ZTI

ea5db8d658f42acad38106cbc46eea5944607eb709fb00f8adb501d4779fbea0	*XecureIO_v20.dll*
3fc6c5df4a04d555d5cbf2ca53bed7769b5595fc6143a2599097cb6193ef8810	*XecureIO_v20.dll*

Backdoor.Win32.BUMBLEB.ZTIC

eeca34fba68754e05e7307de61708e4ce74441754fcc6ae762148edf9e8e2ca0	ore
6690b7ace461b60b7a72613c202d70f4684c8cdc5afbb4267c67b5fe5dbf828e	bin
4ecde81a476f1e4622d192fe2f120f7c5c3ec58bf118b791d5532f3ff61c09ee	bin
8ab8bb836b074e170c129b7f0523d256930fd1f8cf126ca1875b450fdb6c4c05	bin
515cb31b2c89df83ea6d54d5c0c3e4fe9a024319d9bd8fd76ad351860bd67ea3	ore
8e340746339614ca105a1873dad471188b24421648d080e37d52b87f4ced5e6d	bin

C&C:

- *http[:]//www[.]synolo[.]ns01[.]biz:80/update*
- *http[:]//118[.]163[.]105[.]130:80/update*

MITRE

Tactics	Techniques
Defense Evasion	T1574.002 - Hijack Execution Flow: DLL Side-Loading
	T1070.004 - Indicator Removal on Host: File Deletion
	T1055 - Process Injection
	T1480.001 - Execution Guardrails: Environmental Keying
Persistence	T1547.001 - Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
	T1037.001 - Boot or Logon Initialization Scripts: Logon Script (Windows)
	T1548.003 - Create or Modify System Process: Windows Service
Privilege Escalation	T1548.002 - Abuse Elevation Control Mechanism: Bypass User Account Control
Collection	T1056.001 - Input Capture: Keylogging
Reconnaissance	T1592 - Gather Victim Host Information
Command and Control	T1071.001 - Application Layer Protocol: Web Protocols
	T1090 - Proxy
	T1573.001 - Encrypted Channel: Symmetric Cryptography
	T1132.001 - Data Encoding: Standard Encoding
Resource Development	T1587.001 - Develop Capabilities: Malware