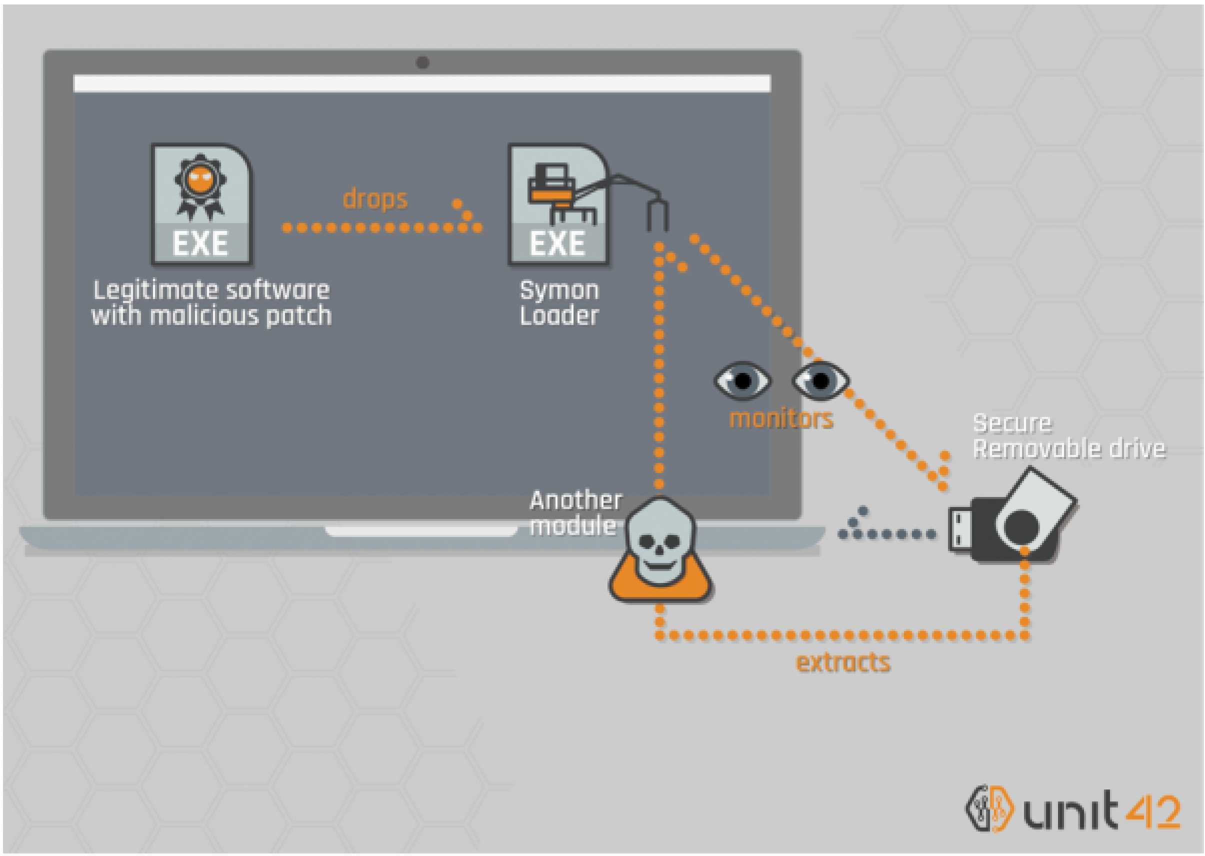
Tick Group Weaponized Secure USB Drives to Target Air-Gapped Critical Systems

unit42.paloaltonetworks.com/unit42-tick-group-weaponized-secure-usb-drives-target-air-gapped-critical-systems

By Kaoru Hayashi and Mike Harbison

June 22, 2018



Summary

Tick is a cyberespionage group primarily targeting organizations in Japan and the Republic of Korea. The group is known to conduct attack campaigns with various custom malware such as Minzen, Datper, Nioupale (aka Daserf), and HomamDownloader. Unit 42 last wrote about the Tick group in July 2017.

Recently, Palo Alto Networks Unit 42 discovered the Tick group targeted a specific type of secure USB drive created by a South Korean defense company. The USB drive and its management system have various features to follow security guidelines in South Korea.

The weaponization of a secure USB drive is an uncommon attack technique and likely done in an effort to spread to air-gapped systems, which are systems that do not connect to the public internet. In addition, our research shows that the malware used in these attacks will only try to infect systems running Microsoft Windows XP or Windows Server 2003. This is despite the fact that the malware appears to have been created when newer versions of Windows software were available. This would seem to indicate an intentional targeting of older, out-of-support versions of

Microsoft Windows installed on systems with no internet connectivity. Air-gapped systems are common practice in many countries for government, military, and defense contractors, as well as other industry verticals.

We have not identified any public reporting on this attack, and we suspect the Tick group used the malware described in this report in attacks multiple years ago. Based on the data collected, we do not believe this malware is part of any active threat campaign.

Our picture of this past attack is incomplete at this time. Based on our research thus far, we are able to sketch out the following hypothesized attack scenario:

1. The Tick Group somehow compromised a secure type of USB drive and loaded a malicious file onto an unknown number of them. These USB drives are supposed to be certified as secure by the South Korean ITSCC (English).
2. The Tick Group created a specific malware we are calling SymonLoader that somehow gets on older Windows systems and continuously looks for these specific USB drives.
3. SymonLoader specifically targets Windows XP and Windows Server 2003 systems ONLY.
4. If SymonLoader detects the presence of a specific type of secure USB drive, it will attempt to load the unknown malicious file using APIs that directly access the file system.

In the research below, we outline our findings around SymonLoader. We do not currently have either a compromised USB drive nor the unknown malicious file we believe is implanted on these devices. Because of this we are unable to describe the full attack sequence.

Because we do not have either a compromised USB drive or the unknown malicious file, we are also unable to determine how these USB drives have been compromised. Specifically, we do not know if there has been a successful compromise in the supply-chain making these devices, or if these have been compromised post-manufacturing and distributed using other means such as social engineering.

Tick and Trojanized Legitimate Software

Unit 42 hasn't identified the initial delivery method; the overview of the infection process is shown below in Figure 1.

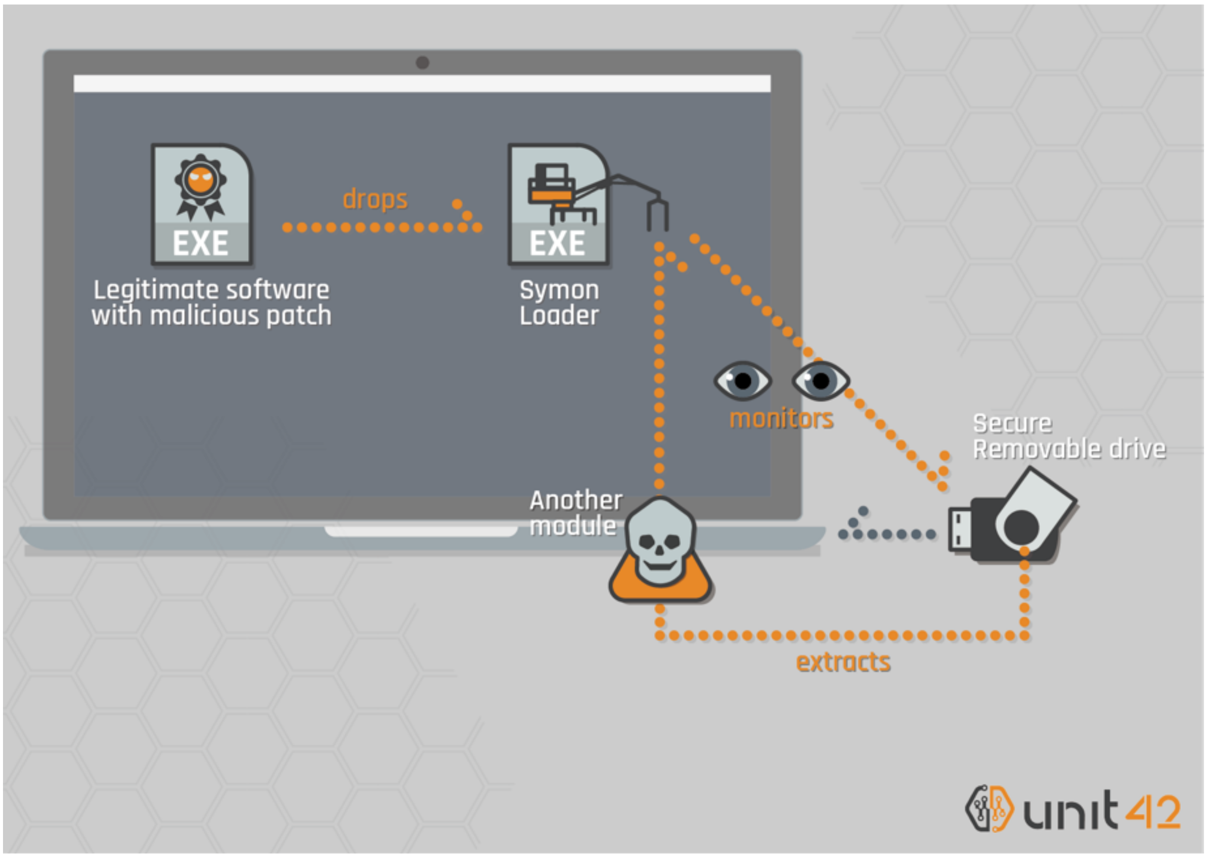


Figure 1 Infection process

First, the attacker tricks users with a Trojanized version of legitimate software to install the loader program, which is a new tool to Tick we've named "SymonLoader".

When executed, the loader starts monitoring storage device changes on a compromised machine. If SymonLoader detects the targeted type of secure USB drive, it attempts to access the storage through the device driver corresponding to the secure USB and checks for strings specific to one type of secure USB in the drive information fields. Then, it accesses a predefined location of the storage on the USB and extracts an unknown PE file.

As we described in a previous blog last July, the Tick group Trojanized a legitimate program and embedded malware called HomamDownloader. The attackers then sent the Trojanized legitimate applications as attachments to spear phishing email targets. When executed, the Trojanized legitimate application drops HomamDownloader and installs the legitimate program. Recipients may not be aware of the malware as the legitimate application works as expected.

In our research into these latest attacks, we found additional legitimate Trojanized Korean language software since publishing out blog last year (Table 1). Similar to the previous samples we examined in July 2017, these newly Trojanized legitimate programs also drop HomamDownloader. Also like we saw in our July 2017 research, HomamDownloader can install other malicious files from the remote C2 server; in this case, pre.englandprevail[.]com.

Trojanized Legitimate Software	SHA256
Movie Player installer	b1bb1d5f178b064eb1d7c9cc7cad-cf8b3959a940c14cee457ce3aba5795660aa
Industrial battery monitoring software	3227d1e39fc3bc842245ccdb16eeaadad3bcd298e811573b2e68e-f2a7077f6f6

Storage encryption software	92e0d0346774127024c672cc7239dd269824a79e85b84c532128fd9663a0ce78
File encryption software	33665d93ab2a0262551c61ec9a3adca2c2b8dfea34e6f3f723274d88890f6ceb

Table 1 Trojanized Korean programs

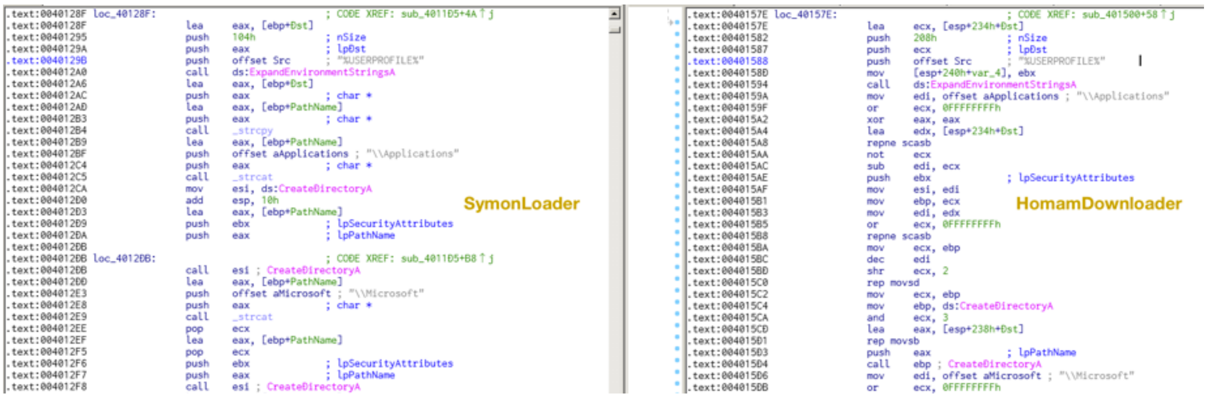
During our investigation, we found an interesting sample on January 21, 2018 (Table 2). Similar to the samples listed above, this sample is a Trojanized version of a legitimate program and drops malware. In this case, the Trojanized application is a Japanese language GO game. Instead of installing HomamDownloader like we observed in July 2017, this Trojanized program installs a new loader we've named SymonLoader.

SymonLoader extracts a hidden executable file from a specific type of secure USB drive and executes it on the compromised system. Unfortunately, we do not have a copy of this file.

Trojanized Legitimate Software	SHA256
GO Game	8549dcbdfc6885e0e7a1521da61352ef4f084d969dd30719166b47fdb204828a

Table 2 Trojanized Japanese language program

Despite the differences from previous samples, we believe this sample is related to the Tick group because the shellcode in the Trojanized Japanese game is exactly the same as that found in the Trojanized Korean programs described earlier. Also, SymonLoader shares code with HomamDownloader (Figure 2). The Tick group is known to develop and consistently update custom tools. As such, code reuse like this is consistent with their development practices.



```
.text:0040128F loc_40128F:                             ; CODE XREF: sub_4011D5+4A↑j
.text:0040128F                 lea     eax, [ebp+Dst]
.text:00401295                 push    104h            ; nSize
.text:0040129A                 push    eax             ; lpDst
.text:0040129B                 push    offset Src      ; "%USERPROFILE%"
.text:004012A0                 call    ds:ExpandEnvironmentStringsA
.text:004012A6                 lea     eax, [ebp+Dst]
.text:004012AC                 push    eax             ; char *
.text:004012AD                 lea     eax, [ebp+PathName]
.text:004012B3                 push    eax             ; char *
.text:004012B4                 call    _strcpy
.text:004012B9                 lea     eax, [ebp+PathName]
.text:004012BF                 push    offset aApplications ; "\\Applications"
.text:004012C4                 push    eax             ; char *
.text:004012C5                 call    _strcat
.text:004012CA                 mov     esi, ds:CreateDirectoryA
.text:004012D0                 add     esp, 10h
.text:004012D3                 lea     eax, [ebp+PathName]
.text:004012D9                 push    ebx             ; lpSecurityAttributes
.text:004012DA                 push    eax             ; lpPathName
.text:004012DB
.text:004012DB loc_4012DB:                             ; CODE XREF: sub_4011D5+B8↑j
.text:004012DB                 call    esi ; CreateDirectoryA
.text:004012DD                 lea     eax, [ebp+PathName]
.text:004012E3                 push    offset aMicrosoft ; "\\Microsoft"
.text:004012E8                 push    eax             ; char *
.text:004012E9                 call    _strcat
.text:004012EE                 pop     ecx
.text:004012EF                 lea     eax, [ebp+PathName]
.text:004012F5                 pop     ecx
.text:004012F6                 push    ebx             ; lpSecurityAttributes
.text:004012F7                 push    eax             ; lpPathName
.text:004012F8                 call    esi ; CreateDirectoryA
```

```
.text:0040157E loc_40157E:                             ; CODE XREF: sub_401500+58↑j
.text:0040157E                 lea     ecx, [esp+234h+Dst]
.text:00401582                 push    208h            ; nSize
.text:00401587                 push    ecx             ; lpDst
.text:00401588                 push    offset Src      ; "%USERPROFILE%"
.text:0040158D                 mov     [esp+240h+var_4], ebx
.text:00401594                 call    ds:ExpandEnvironmentStringsA
.text:0040159A                 mov     edi, offset aApplications ; "\\Applications"
.text:0040159F                 or      ecx, 0FFFFFFFFh
.text:004015A2                 xor     eax, eax
.text:004015A4                 lea     edx, [esp+234h+Dst]
.text:004015A8                 repne scasb
.text:004015AA                 not     ecx
.text:004015AC                 sub     edi, ecx
.text:004015AE                 push    ebx             ; lpSecurityAttributes
.text:004015AF                 mov     esi, edi
.text:004015B1                 mov     ebp, ecx
.text:004015B3                 mov     edi, edx
.text:004015B5                 or      ecx, 0FFFFFFFFh
.text:004015B8                 repne scasb
.text:004015BA                 mov     ecx, ebp
.text:004015BC                 dec     edi
.text:004015BD                 shr     ecx, 2
.text:004015C0                 rep movsd
.text:004015C2                 mov     ecx, ebp
.text:004015C4                 mov     ebp, ds:CreateDirectoryA
.text:004015CA                 and     ecx, 3
.text:004015CD                 lea     eax, [esp+238h+Dst]
.text:004015D1                 rep movsb
.text:004015D3                 push    eax             ; lpPathName
.text:004015D4                 call    ebp ; CreateDirectoryA
.text:004015D6                 mov     edi, offset aMicrosoft ; "\\Microsoft"
.text:004015DB                 or      ecx, 0FFFFFFFFh
```

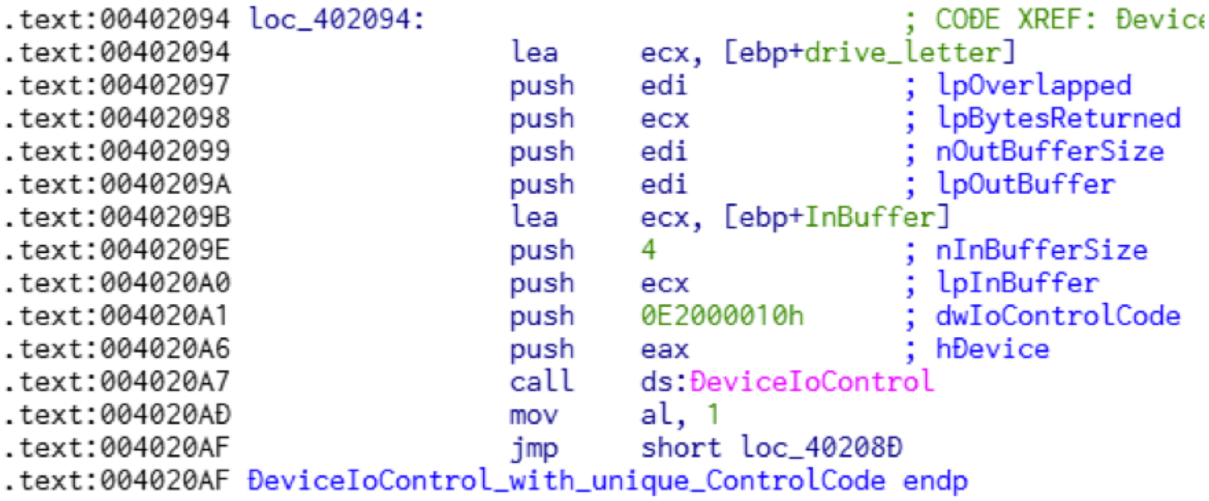
Figure 2 Sharing code between SymonLoader and HomamDownloader

Secure USB

SymonLoader first checks the operating system version of the target host and if it is newer than Windows XP or Windows Server 2003, it stops working. According to the timedate stamp in the PE header, the malware was created on 26 September 2012. Both Windows 7 and Windows Server 2008 were already released by then if the timedate value is not modified.

After checking the OS version, SymonLoader creates a hidden window named “device monitor” that starts monitoring storage device changes on the compromised system. When a removable drive is connected the malware checks the drive letter and drive type. If the drive letter is not A or B, and the drive type is not a CDROM the malware calls CreateFile API and gets a handle to the storage device. By excluding drives A and B (typically used for floppy drives) and CDROM drive types, it appears likely that the malware is targeting removable USB drives.

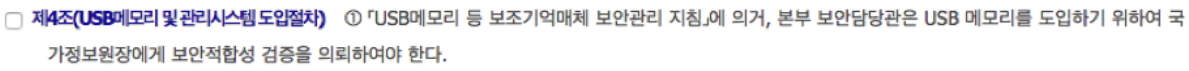
Next, the malware calls the DeviceIoControl() function with an undocumented custom control code, 0xE2000010 (Figure 3). The control code consists of four different types of values; DeviceType, Function, Access, and Method. In this case, the DeviceType value is 0xE200 computed as: (0x0E2000010 & 0xffff0000)>>0x10.   According to Microsoft, this specific DeviceType value should be in the range for third-party vendors. To function properly, the third-party driver needs to be present on the compromised system before the malware calls DeviceIoControl() with the custom control code 0xE2000010. But which third-party driver? There is a clue in the next function.



```
.text:00402094 loc_402094:                             ; CODE XREF: Devic
.text:00402094                 lea     ecx, [ebp+drive_letter]
.text:00402097                 push    edi             ; lpOverlapped
.text:00402098                 push    ecx             ; lpBytesReturned
.text:00402099                 push    edi             ; nOutBufferSize
.text:0040209A                 push    edi             ; lpOutBuffer
.text:0040209B                 lea     ecx, [ebp+InBuffer]
.text:0040209E                 push    4               ; nInBufferSize
.text:004020A0                 push    ecx             ; lpInBuffer
.text:004020A1                 push    0E2000010h      ; dwIoControlCode
.text:004020A6                 push    eax             ; hDevice
.text:004020A7                 call    ds:DeviceIoControl
.text:004020AD                 mov     al, 1
.text:004020AF                 jmp     short loc_40208D
.text:004020AF DeviceIoControl_with_unique_ControlCode endp
```

Figure 3 DeviceIoControl with custom IoControlCode

SymonLoader gets device information by SCSI INQUIRY command using the IOCTL_SCSI_PASS_THROUGH parameter and determines if it is the targeted drive by searching for a specific string in the Vendor or Product identification on INQUIRY data. Our research into the string used in these searches showed a company in the South Korea Defense Industry whose name matched the string. This company develops information and communication security equipment used by military, police, government agencies and public institutions. In a press, this company announced that they make secure USB storage devices that are certified to meet security requirements set out by South Korea’s IT Security Certification Center (ITSCC (English)). In South Korea, certain organizations are required to follow the “USB storage medium guideline” and using only the devices passed the audit by the government agency. For example, the guideline of the Ministry of Unification of South Korea defines the USB memory and management system introduction procedure at section 4, item 1 (Figure 4).



□ 제4조(USB메모리 및 관리시스템 도입절차) ① 「USB메모리 등 보조기억매체 보안관리 지침」에 의거, 본부 보안담당관은 USB 메모리를 도입하기 위하여 국가정보원장에게 보안적합성 검증을 의뢰하여야 한다.

Figure 4 USB memory introduction procedure

Google translation from Korean to English follows.

“According to ‘Guidelines for Security Management of Auxiliary Storage Media such as USB Memory’, the headquarters security officer shall request the National Intelligence Service to verify the security compliance to introduce USB memory.”

We found the third-party device driver in question in the installer of the secure USB drive in a public sample repository, and confirmed it supports the custom control code, 0xE2000010. The driver provides some functions to applications, including access to the corresponding secure USB volumes. We feel this evidence shows that the malware attempts to work only on the secure USB product made by this particular company.

Loading Hidden Module

If SymonLoader finds it is on a Windows XP or Windows Server 2003 system and finds that a newly attached device is a USB drive made by this particular company, then it will extract an unknown executable file from the USB. While we do not have this file, we can glean information about it by analyzing SymonLoader and the third-party driver. The attacker encrypted the unknown executable file and concealed it at the ending part of the secure USB storage in advance. The hidden data is not accessible through logical file operation APIs, such as ReadFile(). Instead, SymonLoader uses Logical Block Addressing(LBA) and SCSI commands to read the data physically from the particular expected location on the removable drive.

LBA is a simple linear addressing scheme. Storage is divided into blocks by fixed size, and each block has a number starting from zero to N-1, depends on the volume size. Applications can specify the block number and access the data by SCSI commands.

Finally, SymonLoader saves the extracted file in the temporary directory on the local disk and executes it. The procedure is as follows:

1. Obtains final Logical Block Address(LBA) of the storage “N-1” by using the READ CAPACITY (10) command.
2. Read the third last block “N-3” by READ (10) command and decrypts it.
3. From the decrypted data, gets the LBA “X” where the main module locates.
4. Loads data from LBA “X” to “N-4” by READ (10) command and decrypts it.
5. Saves the decrypted file as %Temp%\[random characters].tmp and execute it.
6. Writes hostname and local time of the compromised system at LBA “N-2” by SAVE (10) command.

Figure 5 shows the data layout of the malicious storage from the perspective of Logical Block Addressing.

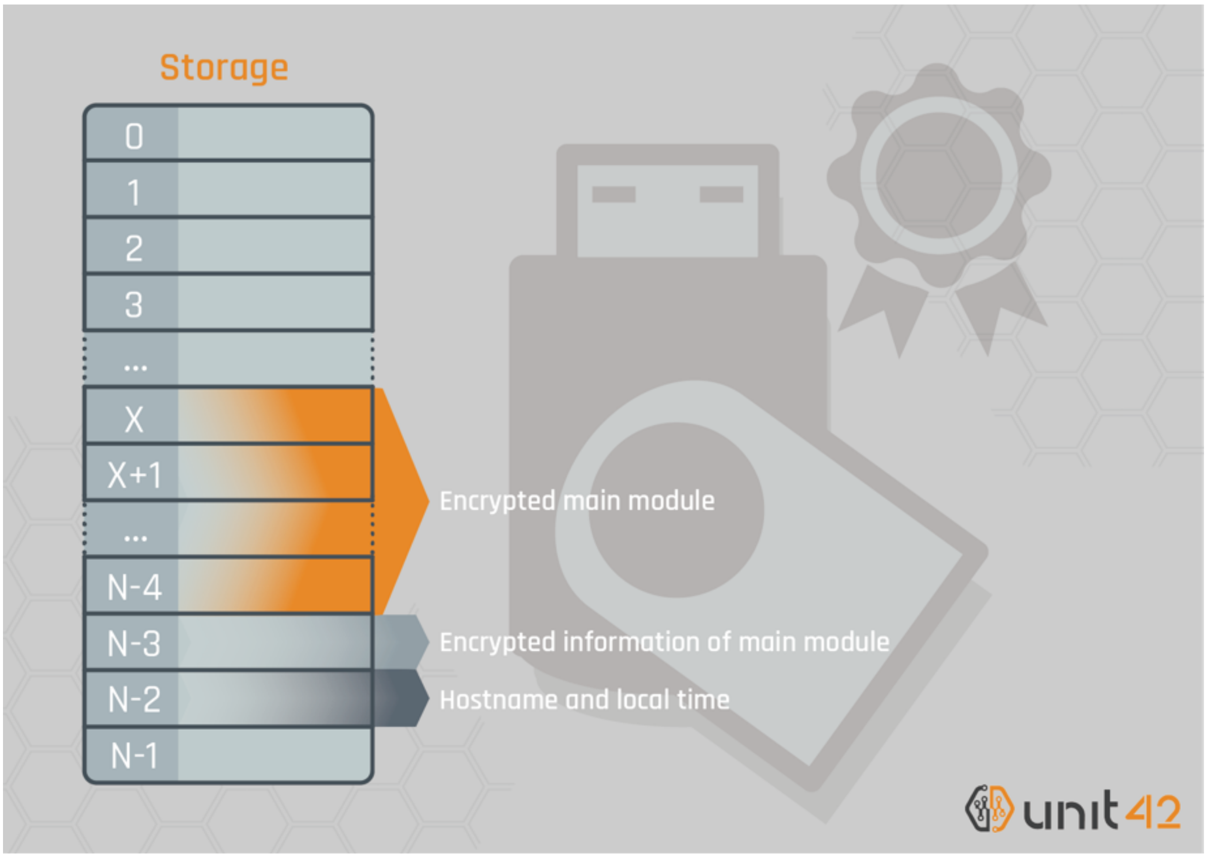


Figure 5 Data layout on the malicious storage

Conclusion

The Tick group uses Trojanized legitimate applications to trick victims into installing first stage malware, mostly HomamDownloader. In this research, we identified a previously unknown loader malware being dropped instead of HomamDownloader, which was most likely used in attacks multiple years ago. In contrast to HomamLoader, which requires an Internet connection to reach its C2 server to download additional payloads, SymonLoader attempts to extract and install an unknown hidden payload from a specific type of secure USB drive when it's plugged into a compromised system. This technique is uncommon and hardly reported among other attacks in the wild.

While we do not have a copy of the file hidden on the secure USB, we have more than enough information to determine it is more than likely malicious. Weaponizing a secure USB drive is an uncommon technique and likely done in an effort to compromise air-gapped systems, which are systems that do not connect to the public internet. Some industries or organizations are known for introducing air gapping for security reasons. In addition, outdated versions Operating Systems are often used in those environments because of no easy-update solutions without internet connectivity. When users are not able to connect to external servers, they tend to rely on physical storage devices, particularly USB drives, for data exchange. The SymonLoader and secure USB drive discussed in this blog may fit for this circumstance.

Palo Alto Networks customers are protected from these threats in the following ways:

1. All samples discussed are classified as malicious by the WildFire sandbox platform.
2. All identified domains have been classified as malicious.
3. AutoFocus users can track the malware described in this report using Tick campaign tag, SymonLoader and HomamDownloader malware tags.

4. Customers running Traps are protected from the discussed threats.

IoCs

SymonLoader

Malformed Legitimate software SHA256
8549dcbdfc6885e0e7a1521da61352ef4f084d969dd30719166b47fdb204828a

SysmonLoader SHA256
31aea8630d5d2fcbb37a8e72fe4e096d0f2d8f05e03234645c69d7e8b59bb0e8

Mutex
SysMonitor_3A2DCB47

File Path
%ProgramFiles%\Windows NT\Accessories\Microsoft\msxml.exe
%UserProfile%\Applications\Microsoft\msxml.exe

Registry Entry
HKLM\Software\Microsof\Windows\CurrentVersion\run\”xml” = %ProgramFiles%\Windows NT\Accessories\Microsoft\msxml.exe

HKCU\Software\Microsof\Windows\CurrentVersion\run\”xml” = %UserProfile%\Applications\Microsoft\msxml.exe

HomamDownloader

Trojanized Legitimate Software SHA256
b1bb1d5f178b064eb1d7c9cc7cadcf8b3959a940c14cee457ce3aba5795660aa

3227d1e39fc3bc842245ccdb16eeaadad3bcd298e811573b2e68ef2a7077f6f6

92e0d0346774127024c672cc7239dd269824a79e85b84c532128fd9663a0ce78

33665d93ab2a0262551c61ec9a3adca2c2b8dfea34e6f3f723274d88890f6ceb

HomamDownloader SHA256
019874898284935719dc74a6699fb822e20cdb8e3a96a7dc8ec4f625e3f1116e

ee8d025c6fea5d9177e161dbcedb98e871baceae33b7a4a12e9f73ab62bb0e38

f817c9826089b49d251b8a09a0e9bf9b4b468c6e2586af60e50afe48602f0bec

C2 of HomamDownloader
pre.englandprevail[.]com

Get updates from Palo Alto Networks!

Sign up to receive the latest news, cyber threat intelligence and research from us

By submitting this form, you agree to our Terms of Use and acknowledge our Privacy Statement.