

Analysis of Kimsuky's Attack on a South Korean Groupware Vendor Using a New Gomir Family Variant

 enki.co.kr/en/media-center/blog/analysis-of-kimsuky-s-attack-on-a-south-korean-groupware-vendor-using-a-new-gomir-family-variant

July 20, 2026



Executive Summary

- We tracked a campaign by the Kimsuky group targeting South Korean groupware vendors from 2025 through early 2026.
- Kimsuky compromised vendor infrastructure by exploiting vulnerabilities in internet-facing mail servers or by conducting spear-phishing against employees.
- We identified two new malware strains based on Gomir/HttpTroy, which we tracked as BirdTroy and DriveTroy.
- Even after the initial compromise, Kimsuky aggressively pursued lateral movement, including compromising customer groupware servers and stealing the vendors' infrastructure credentials.
- We identified notable similarities with past Kimsuky campaigns across malware characteristics, attack infrastructure, and tooling.

1. Overview

The ENKI WhiteHat Threat Research Team tracked a campaign by Kimsuky, a North Korea-linked threat group, that infiltrated the internal networks of South Korean groupware vendors between 2025 and early 2026. Our analysis revealed that Kimsuky gained control of internet-facing servers through vulnerability exploitation and spear-phishing, and deployed Gomir and its variants.

Kimsuky developed Gomir variants with significantly altered C2 communication methods to evade detection, including leveraging Google Drive as a C2 channel and implementing a new custom protocol. We also observed indicators of aggressive lateral movement, including compromising customer servers that used the affected vendors' groupware and tampering with groupware login pages to harvest employee credentials.

This report also provides Indicators of Compromise(IoCs) and YARA rules to detect the Gomir, BirdTroy, and DriveTroy malware used in this campaign.

2. Background

The Gomir malware family analyzed in this report was first reported in Kimsuky's earlier TrollStealer campaign. In a 2024 [report by S2W](#), Kimsuky was observed distributing TrollStealer and GoBear, both written in Go. [AhnLab reported](#) the same malware family under the name Endoor. A later [report by Symantec](#) then analyzed Gomir, a Linux variant of the Windows-based GoBear backdoor

Kimsuky has historically targeted South Korean corporate infrastructure. [AhnLab's 2024 SmallTiger report](#) documented attacks against South Korean enterprises, where malware previously attributed to Kimsuky was also observed. In the TrollStealer campaign mentioned earlier, Kimsuky also trojanized a security software installer distributed through the website of a South Korean construction industry association.

3. Initial Access

In the compromise of groupware vendor A, identified in November 2025, Gomir was installed through a remote code execution vulnerability in an externally accessible mail server. It remains unclear how Kimsuky acquired prior knowledge of this vulnerability.

 Gomir installation via vulnerability exploitation

Gomir installation via vulnerability exploitation

In the compromise of groupware vendor B, identified in December 2025, initial access to the internal infrastructure was achieved via an employee's PC. The installed malware was identified as a variant of HttpTroy, which [Gen Digital reported](#) in September 2025. For a detailed analysis of HttpTroy's behavior, refer to Gen Digital's report. Based on past HttpTroy distribution cases, we assess that Kimsuky installed the malware on the employee's PC using social engineering techniques such as spear-phishing. Key information on the HttpTroy variant used in this attack is shown in the table below.

MD5	633c3076c396f2200048d1b62da204d2
Export Function	hello
C&C Server	hxxps://auth.samecloud.o-r[.]kr/index.php
Task Name (Persistence)	AhnlabUpdate

Key information on the HttpTroy variant used in the attack

Kimsuky used HttpTroy to install additional tools, including DWAgent and a proxy tool. DWAgent is an open-source remote access tool for remote desktop and system management, which Kimsuky abused to remotely control the victim's Windows PC through its web UI. Kimsuky recorded their account credentials in `config.json`, compressed all DWAgent components into a zip archive, and installed and run it on the victim's server. The `config.json` used in the attack is shown below.

```
{
  "enabled": true,
  "key": "sUgmupzJnAcroInrxqX0",
  "listen_port": 7950,
  "nodes": [
    {
      "id": "ND896147",
      "port": "443",
      "server": "node896147.dwservice[.]net"
    },
    {
      "id": "ND828765",
      "port": "443",
      "server": "node828765.dwservice[.]net"
    },
    {
      "id": "ND449013",
      "port": "443",
      "server": "node449013.dwservice[.]net"
    }
  ],
  "password": "eJxLL090rQorcisqzM7ITa8qcvVzcQ50Tsr1KisoS0oHALaSC5E=",
  "url_primary": "https://www.dwservice[.]net/"
}
```

Kimsuky also installed a proxy tool in the victim environment to facilitate lateral movement into isolated network segments. The proxy tool is built on the yamux (TCP session multiplexer) library. After connecting to a relay address and accepting yamux streams, it opens a TCP connection to a target address for each stream and relays data bidirectionally.

4. Malware Analysis

4.1. Gomir

Throughout this campaign, Kimsuky installed Gomir first, immediately after gaining initial access to internal infrastructure. The analyzed Gomir sample is packed with UPX 3.96, and its Go module path is shown below.

Go module path: `local[.]github[.]com/Github-Lin/Main`

 Gomir binary showing Go module path strings

Gomir binary showing Go module path strings

On execution, it checks the command-line arguments. If the argument is `install`, it first executes the persistence routine. The persistence routine checks the effective group ID via `getegid` and registers a systemd service if running as root, or a cron job if not. When running as root, it establishes persistence as a systemd service through the following steps.

1. Copies itself to `/var/log/rsyslogd` with permission `0777`
2. Writes a unit file to `/etc/systemd/system/rsyslogd.service`
3. `systemctl daemon-reload` → `systemctl reenale rsyslogd` → `systemctl start rsyslogd`
4. Deletes the original

When not running as root, it uses `crontab` instead through the following steps.

1. Reads the existing content with `crontab -l`, appends a `/10 * * * * <own path>` entry, and writes it to `cron.txt`
2. Runs `crontab cron.txt`
3. Deletes `cron.txt` and re-executes

After establishing persistence (or immediately, if `install` is not specified), it enters the C&C communication loop and generates a bot ID. The bot ID follows the format `XX-[0-9a-f]{10}`, where the hex portion is the first 5 bytes of the MD5 hash of the username, hostname, and MAC address.

 Gomir bot ID generation routine

Gomir bot ID generation routine

Gomir spawns a TCP listener alongside the C2 loop on startup to prevent duplicate execution. Normal mode occupies TCP port 19838 and install mode TCP port 19837. If the port is already in use, it determines that another instance is running and exits.

 Gomir port-binding routine

Gomir port-binding routine

Gomir uses HTTPS POST in its main communication loop with the C&C server. The C&C server URL is as follows.

C&C server: `hxxps://commit[.]hanbiro[.]o-r[.]kr/index.php`

The POST parameters follow the format `<prefix>[a-zA-Z0-9]{9}`. The parameters for the command-retrieval request are shown in the table below.

Prefix	Value	Description
a	2	Request type
b	Bot ID + "1"	Victim Identifier
c	-	Unknown

Gomir command-retrieval request parameters

If the first byte of the server response is S (0x53), the remainder is decoded as URL-safe Base64, which uses "-" instead of the "/" of standard Base64. The first 4 bytes of the decoded result serve as the decryption key, and $(\text{ciphertext} - \text{key}[i\%4]) \bmod 255$ is applied to the remaining data, leaving 0x00 bytes unchanged.

When sending command results, a random 4-byte key is generated and the operation $(\text{plaintext} + \text{key}[i\%4]) \bmod 255$ is applied to each byte, leaving 0x00 bytes unchanged. The 4-byte encryption key is prepended to the ciphertext, and the combined result is then encoded as URL-safe Base64. The parameters for the result-submission request are shown in the table below.

Prefix	Value	Description	 	
x	1	Request type	 	
y	Bot ID + "2"	Victim Identifier	 	
z	URL-safe Base64(key 	ciphertext)	Encrypted execution result	

Gomir command-result submission request parameters

The handler functions Gomir invokes for each command code received from the C&C server, and the operations they perform, are shown in the table below. Note that the Upload and Download handlers are named from the C&C server's perspective.

Command code	Handler	Function
--------------	---------	----------

01	Process_Sleep	Wait for a specified duration
02	Process_Cmd	Execute shell command
03	Process_Pwd	Query current directory
04	Process_Cd	Change directory
05	Process_Conn	TCP connection test
06	Process_Exit	Terminate
07	Process_Where	Query execution path
08	Process_Dirsize	Query directory size
09	Process_GetInfo	Query system information
10	-	Change shell path
11	-	Set code page
12	Process_Hibernate	Hibernate until a specified date/time (Asia/Seoul time zone)
13	"Not implemented on Linux!" 전송	Vestige of the original Windows variant (screenshot capture)
14	Process_PForward_Add	Add proxy
15	Process_PForward_List	List proxies
30	Process_Upload	Transfer a file from the C&C server to the victim system
31	Process_Download	Transfer a file from the victim system to the C&C server
Others	Sends "Unknown Command!\r "	

Gomir command code list

The proxy functions of 14 and 15 perform essentially the same role as the yamux-based proxy tool described earlier. The authentication key bf215181b5140522137b3d4f6b73544a is the MD5 hash of

github. Combined with the Go module path (local.github.com), this suggests that github is a recurring indicator associated with this actor.

 Gomir proxy authentication key

Gomir proxy authentication key

A notable trait of the file transfer functions (30 and 31) is that the handler names Upload and Download are oriented to the C&C server. After receiving the command, Gomir generates a session token in the format [a-zA-Z0-9]{10} and sends it to the C&C server. Until the transfer completes, it communicates with the C&C server using the session token instead of the bot ID. The command codes used for the transfer are shown in the table below.

Command Code	Description
20	Data Chunk
21	Transmission Complete (EOF)
22	Error
OK	Acknowledgment (ACK)

Gomir file transfer command code list

4.2. BirdTroy

BirdTroy is a previously undocumented RAT that exhibits characteristics of both Gomir and HttpTroy. Its build module path is Bird/module, suggesting the threat actor used Bird as an internal project name. Since it shares command codes and functionality with HttpTroy, we named it BirdTroy.

 BirdTroy Go module path

BirdTroy Go module path

On execution, it parses command-line arguments and uses them to control the malware's behavior. Each argument and its function are described in the table below.

Command-line argument	Description
-url	C&C server URL (default: hxxps://69[.]10[.]50[.]165:8443)
-method	Transport protocol selection (http or http3, default: http)
-m	Multi mode: adds the multi_ prefix to the bot ID

-i Install mode: runs the persistence routine before entering the C&C communication loop

BirdTroy command-line argument list

For duplicate-execution prevention, it uses a file created by appending the `.tmp` extension to the executable's absolute path. It attempts an exclusive lock (`LOCK_EX|LOCK_NB`) on this file via `syscall.Flock` and exits if the lock cannot be acquired. The same file also serves as bot ID storage. If the file does not exist or is empty, it generates a bot ID in the format `<[a-zA-Z0-9]{10}>` and saves it to the file at that path.

BirdTroy duplicate-execution prevention routine

BirdTroy duplicate-execution prevention routine

When the `-i` flag is specified, it runs a persistence routine identical to Gomir's, which confirming that BirdTroy is a Gomir variant. After establishing persistence (or immediately, if `-i` is not specified), it enters the C&C communication loop.

BirdTroy communicates with the C&C server using custom packets. It encodes the entire packet as Base64 and sends it via HTTP POST with a `Content-Type: application/json` header. If `http3` is specified as the transport protocol, it uses HTTP/3 (QUIC) transport based on the github.com/quic-go/quic-go library. HTTP/3 (QUIC) is a UDP-based protocol, so it can bypass inspection by network security appliances designed for TCP-based traffic. The structure of the packets used to communicate with the C&C server is shown below.

```
type PACKET struct {
    Type      uint8    // Packet type
    ChatID    uint8    // Session identifier
    Num       uint8    // Sequence number
    Checksum  uint32   // First 4 bytes of MD5 (integrity check)
    Data      []byte   // Payload
}
```

The checksum for the packet's Data field is the first 4 bytes of its MD5 hash, interpreted as a big-endian uint32. The packet type stored in the first byte of the packet is shown in the table below; type 3 is not defined.

Type	Description
1	Session initialization (sends bot ID)
2	Server response (assigns ChatID)
4	Keepalive

5	Command / result
6	Start file reception
7	EOF (transfer complete)
8	Connection close notification / retransmission request
9	Data chunk
10	ACK
11	Start file transfer

BirdTroy packet type information

For an overview of the entire communication flow, refer to the figure below.

 BirdTroy communication flow overview

BirdTroy communication flow overview

A command string received in a type 5 packet is split on whitespace, and the first token is used as the command code. As in Gomir, the command codes and handler function names for the file transfer functions are written from the C&C server's perspective. The handler functions invoked for each command code, and the operations they perform, are shown in the table below.

Command code	Handler	Description
up	Bird_module_file.UploadFile	Transfer a file from the C&C server to the victim system
down	Bird_module_file.DownFile	Transfer a file from the victim system to the C&C server
sleep	time.Sleep	Wait for N hours
delete	Bird_module_shell.Selfdelete	Self-delete
screen	Bird_module_file.ScreenDown	Capture and upload a screenshot
Others	Bird_module_shell.RunShell	Execute shell command

BirdTroy command code list

4.3. DriveTroy

DriveTroy is a backdoor that abuses Google Drive as its C&C server. It shares numerous similarities with Gomir, including the persistence mechanism and command codes. We therefore assess that it is a newly developed Gomir variant. Its build path is embedded as
C:/Users/jira/go/src/jira/payload/.

 DriveTroy Go module path

DriveTroy Go module path

DriveTroy extracts the configuration data used for Google Drive communication from the end of its own file. It reads the last 4 bytes of the file as an ASCII string to obtain the configuration size (in bytes), then moves back by that amount to read the encrypted configuration data. The configuration data is RC4-encrypted and then encoded with the URL-safe Base64 ("-_" instead of "+/") used in Gomir. The RC4 key and the decrypted configuration are shown below.

RC4 key: @st34GMEgfesy4R#r21f

Item	Value
uid	E8C46207
client_id	368946970262-i4nsp00cnj1olit05i14v6u2Ingpvugv.apps.googleusercontent.com
client_secret	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
refresh_token	1//0eAl3kWDUNkxHCgYIARAAGA4SNwF-L9lrQd8BchULcAyKaEsz-Qk7URxr2QtZj6v5_eJXh1fo-UY3H2Y8Qkrq3ltOWFul1PuwTo

DriveTroy Google account authentication information

The threat actor's Google account information, identified through these credentials, is shown in the table below.

Item	Details
email	vcenter[.]secmail@gmail[.]com
username	mo mo

DriveTroy Google account information

After initializing the Engine, it sends an HTTP GET request to `hxxps://api[.]ipify[.]org` and looks up the victim system's external IP address. It encrypts the retrieved IP with RC4, encodes it as

URL-safe Base64, and uploads it to Google Drive under the filename E8C46207.ses. The RC4 key used to encrypt communication data is the same as the one used to decrypt the configuration data.

세션 파일 업로드 루틴

Session file upload routine

In the subsequent communication loop, it polls commands by downloading the E8C46207.cmd file from Google Drive every second. The received data is decoded from URL-safe Base64 and then decrypted with RC4. The first 2 bytes are the command code in ASCII, and the remainder is the command data. In the same way, it encrypts the execution result with RC4, encodes it as URL-safe Base64, and uploads it to Google Drive under the filename E8C46207.res.

Google Drive polling routine

Google Drive polling routine

The command code structure is similar to Gomir's, but only 8 command are implemented in the analyzed sample. Unlike Gomir and BirdTroy, the Recv and Upload handlers are named from the victim system's perspective. The operations performed for each command code are shown in the table below.

Command code	Handler	Function
00	GetBasicInfo	Collect system information (user, os, path)
01	ShellRun	Execute shell command
02	RecvFile	Download a file from Google Drive to the victim system
03	UploadFile	Upload a file from the victim's PC to Google Drive
04	Sleep	Wait for a specified duration
05	AutoRun	Establish persistence (systemd/crontab)
06	Die	Self-delete and terminate
07	ConnCheck	TCP connection test

DriveTroy command code list

Upon receiving command 05, it performs the persistence securing operation. There are differences between the persistence securing routine of Gomir and BirdTroy and that of DriveTroy as shown in the table below.

Item	Gomir / BirdTroy	DriveTroy
------	------------------	-----------

Root privilege check	getegid	Reads /etc/shadow
Service name	rsyslogd	Random choice among syslogd, logd, cachelogd, cachemond, monlogd
ExecStart	/bin/sh -c "/var/log/rsyslogd"	<self path> &
Restart	always	on-failure
Temporary file	cron.txt (current directory)	/tmp/[a-zA-Z0-9]{6}

Persistence routine comparison

5. Post-Compromise Activity

5.1. Customer Compromise Cases

During our analysis, we identified multiple cases in which the threat actor exfiltrated internal information from a compromised groupware vendor and used it to compromise the infrastructure of customers using that vendor's products. These cases reveal that the threat actor made groupware vendors a primary target, obtaining internal configuration information or source code and then pivot to downstream organizations.

5.1.1. Case A

Groupware vendor A was the vendor whose mail server had been compromised via the vulnerability described in Section 3. We identified indications that Gomir had also been installed on the groupware server of one of vendor A's SaaS customers. Although we could not determine the exact intrusion path, we assess that the threat actor enumerated the vendor's infrastructure, obtained configuration information for the SaaS groupware server, and then carried out further intrusion. The timeline for this customer compromise case is shown below.

 Case A timeline
Case A Timeline

5.1.2. Case B

While monitoring the threat actor's Google Drive, where DriveTroy uploads its execution logs, we found indications that a customer of the compromised groupware vendor B had also been breached. The day after we confirmed DriveTroy installation, we observed the output of an `id` command executed on another victim server.

 Customer server information on the threat actor's Google Drive

Customer server information on the threat actor's Google Drive

This username matched a directory name found in a VOC(Voice of Customer) server listing uploaded to the same Google Drive the previous day. The newly infected server was an on-premises customer running vendor B's products. We assess that the threat actors used customer server information or authentication keys from the VOC server to intrude into the customer's infrastructure, then installed DriveTroy and executed remote commands.

 Vendor VOC server directory information on the threat actor's Google Drive

Vendor VOC server directory information on the threat actor's Google Drive

A few days later, the threat actors leveraged a compromised groupware account to access customer Z's email, which uses the same groupware product. The timeline for this customer compromise case is shown below.

 Case B timeline

Case B Timeline

5.2. Groupware Login Page Tampering

During our analysis, we found code in the HTML of the internal groupware login page of one of the compromised groupware vendors that harvests user-submitted credentials to the C&C server via HTTPS POST.

 Original groupware login page

Original groupware login page

 Tampered groupware login page

Tampered groupware login page

Even with control of the groupware server, the threat actors could not recover plaintext passwords from the database alone, since employees' passwords are stored on the server in hashed form. They therefore tampered with the login page to exfiltrate employees' groupware login credentials, allowing them to log in to the groupware directly and continue their reconnaissance. Because the victim server's login process had no secondary authentication such as OTP at the time of the compromise, we assess that the stolen credentials would have provided the threat actor with broad access to internal systems.

6. Attribution

6.1. Malware Similarities

As mentioned in the Background section, the Gomir malware was first reported in Kimsuky's earlier TrollStealer campaign. BirdTroy and DriveTroy differ in their C&C infrastructure and detailed behavior, but they share several functional similarities with Gomir. The table below compares the similarities among the three malware families.

Item	Gomir	BirdTroy	DriveTroy
Language	Go	Go	Go
Target OS	Linux	Linux	Linux
Install mode	install argument	-i flag	Command code 05
Root verification logic	getegid	getegid	Reads /etc/shadow
Persistence	systemd or crontab	systemd or crontab	systemd or crontab
systemd service name	rsyslogd	rsyslogd	Random selection among syslogd, logd, cachelogd, cachemond, monlogd
Command code format	2-byte ASCII digits	Words	2-byte ASCII digits
File transfer handler naming	C&C server's perspective	C&C server's perspective	Victim system's perspective

Malware characteristics comparison

6.2. Attack Infrastructure Links

All C&C servers identified in this report used the same HTTPS certificate. This is the default XAMPP certificate, which has been repeatedly observed in recent Kimsuky campaigns. Details of this HTTPS certificate are shown in the table below.

Fingerprint	Subject DN	Names
-------------	------------	-------

9de541b039cfdb96c7810df49efd958b28cc2df73e314f67c1a91469a2b19796	C=DE, ST=Berlin, L=Berlin, O=Apache Friends, CN=localhost	localhost
--	--	-----------

HTTPS certificate information

The C&C server IPs identified in this report all belong to ASN 19318 and ASN 26666, both of which have been historically favored by Kimsuky. Neither the default XAMPP certificate nor these ASNs are unique to Kimsuky in isolation, but their co-occurrence strengthens attribution to Kimsuky.

In addition, the Windows username `jira` found in DriveTroy's build path also appeared in the "Last saved by" field of the decoy document and in the PDB path of `loadDll.dll`, a malware from Kimsuky's HttpSpy distribution case [reported by ENKI WhiteHat in 2026](#). This indicates either that the same operator is behind both campaigns, or that the two campaigns share a common development environment.

 Decoy document information from the HttpSpy distribution case
Decoy document information from the HttpSpy distribution case

6.3. Attack Tools

DWAgent is a legitimate remote-control tool that North Korea-linked threat groups have long abused for easy remote access to compromised Windows environments. [Kaspersky's analysis of the PebbleDash cluster](#) also identified Kimsuky using this tool.

In addition, among the proxy tools obtained during our analysis, we identified a version that used `dkanehahffk`, the string produced by typing the Korean phrase '아무도몰라' ('nobody knows') on a QWERTY keyboard. This indicates that the tool's developer is familiar with Korean.

 Proxy tool Korean-conversion authentication key
Proxy tool Korean-conversion authentication key

7. Course of Action

In this case, some of the initial access occurred through a remote code execution vulnerability in an externally accessible mail server. To reduce this attack surface, organizations should use regular penetration testing to examine their external attack surface and lateral movement paths.

The threat actor also tampered with the login page to steal credentials and then used them for further intrusion; because no secondary authentication was in place at the time of the compromise,

the stolen credentials could be used to log in immediately. Introducing secondary authentication such as TOTP can block this kind of internal intrusion even when credentials are leaked.

Finally, the threat actors used customer directory information stored on the VOC server as a foothold to pivot to customer infrastructure. Vendors that develop enterprise software should take the following measures to protect their customers' infrastructure:

- Encrypt stored customer authentication keys and access credentials
- Separate access between internal management systems and production servers
- Implement comprehensive audit logging for access to customer servers
- Establish procedures to notify customers when a breach occurs

Detection rules for the analyzed Gomir, BirdTroy, and DriveTroy are attached in Appendix D.

8. Conclusion

From 2025 through early 2026, a North Korea-linked threat actor breached South Korean corporate infrastructure and carried out lateral movement and downstream compromise. After gaining initial access through spear-phishing and vulnerability exploitation, the threat actors expanded into Linux server environments using Gomir and other malware. Most notably, they stole customer server information from a groupware vendor's server to reach downstream customers, and tampered with groupware login pages to harvest employee credentials.

The analyzed variants demonstrate a deliberate shift toward detection evasion, using Google Drive as a C&C channel and custom protocols. BirdTroy and DriveTroy share command structures and persistence routines with Gomir and HttpTroy, indicating that the same actor continues to develop its tooling.

9. Appendix

Appendix A. MITRE ATT&CK

Tactics	Techniques
Initial Access	T1566: Phishing T1190: Exploit Public-Facing Application T1199: Trusted Relationship
Execution	T1059.004: Command and Scripting Interpreter: Unix Shell T1204.002: User Execution: Malicious File

Persistence	T1543.002: Create or Modify System Process: Systemd Service T1053.003: Scheduled Task/Job: Cron
Stealth	T1027.002: Obfuscated Files or Information: Software Packing T1027.013: Obfuscated Files or Information: Encrypted/Encoded File T1036.004: Masquerading: Masquerade Task or Service T1140: Deobfuscate/Decode Files or Information T1070.004: Indicator Removal: File Deletion
Credential Access	T1056.003: Input Capture: Web Portal Capture
Discovery	T1082: System Information Discovery T1083: File and Directory Discovery T1016: System Network Configuration Discovery T1049: System Network Connections Discovery
Lateral Movement	T1021: Remote Services T1078: Valid Accounts T1570: Lateral Tool Transfer
Collection	T1005: Data from Local System T1113: Screen Capture
Command and Control	T1071.001: Application Layer Protocol: Web Protocols T1132.001: Data Encoding: Standard Encoding T1573.001: Encrypted Channel: Symmetric Cryptography T1090: Proxy T1102.002: Web Service: Bidirectional Communication
Exfiltration	T1041: Exfiltration Over C2 Channel T1567.002: Exfiltration to Cloud Storage

MITRE ATT&CK

Appendix B. IOCs

MD5

MD5	Information
e6c6fa32da47d9341b778bfa424abb4c	Gomir (UPX packed)
b1c72139f2cdd9419562369fc6ced4fc	Gomir

84e9b066bebd49036b7fc71b5f5f8d83	Gomir
e911f8f7c49476806ada37f3ebb7a28a	Gomir
aa61e76255a6e13313439655bc02bdf5	Gomir
ca98a51cebdc802d255030b4baa44ca0	BirdTroy
c2e37232556357944a04edf1dec3934b	BirdTroy
a452a860f973c7a43ea804c17e9427d2	DriveTroy
633c3076c396f2200048d1b62da204d2	HttpTroy
dff787bce68c7653495f153c0534cb96	DWAgent.zip

MD5

IP

- 69[.]10[.]50[.]165
- 208[.]73[.]202[.]29
- 163[.]245[.]195[.]172

URLs

- hxxps://auth[.]samecloud[.]o-r[.]kr/index.php
- hxxps://commit[.]hanbiro[.]o-r[.]kr/index.php
- hxxps://oobe[.]webjine[.]o-r[.]kr/index.php
- hxxps://global[.]webjine[.]o-r[.]kr/index.php
- hxxp://www[.]ilskdeid[.]o-r[.]kr:8000/
- hxxps://oauth[.]shacloud[.]o-r[.]kr:8443
- hxxps://69[.]10[.]50[.]165:8443

Appendix C. Metadata

Threat Actor's Google Account Tokens and Information

Item	Info
------	------

client_id	368946970262-i4nsp00cnj1olit05i14v6u2Ingpvugv.apps.googleusercontent.com
client_secret	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
refresh_token	1//0eAl3kWDUNkxHCgYIARAAGA4SNwF-L9lrQd8BchULcAyKaEsz-Qk7URxr2QtcZj6v5_eJXh1fo-UY3H2Y8Qkrq3ltOWFul1PuwTo
email	vcenter[.]secmail@gmail[.]com
username	mo mo

Threat Actor's Google account tokens and information

RC4 Key

@st34GMEgfsy4R#r21f

Appendix D. YARA Rule

```
rule BirdTroy_Linux {
  meta:
    description = "BirdTroy - Go-based Linux backdoor (Bird/module namespace)"
    date = "2026-05-28"
    hash1 = "ca98a51cebdc802d255030b4baa44ca0"
    hash2 = "c2e37232556357944a04edf1dec3934b"
  strings:
    $ns1 = "Bird/module/packet."
    $ns2 = "Bird/module/autorun."
    $ns3 = "Bird/module/shell."
    $ns4 = "Bird/module/protocol."
    $resp1 = "success upload file"
    $resp2 = "usege: sleep [hour]"
    $resp3 = "delete success"
    $code_bswap = { 8B 00 0F C8 48 81 C4 B8 00 00 00 5D C3 }
  condition:
    uint32(0) == 0x464C457F and (
      3 of ($ns*) or
      (2 of ($resp*) and 1 of ($ns*)) or
      ($code_bswap and 1 of ($ns*))
    )
}
```

```
rule DriveTroy_Linux {
  meta:
    description = "DriveTroy - Go-based Linux backdoor abusing Google Drive"
    date = "2026-05-28"
    hash = "a452a860f973c7a43ea804c17e9427d2"
  strings:
    $ns1 = "jira/payload/gdrive."
```

```

$ns2 = "jira/payload/common."
$ns3 = "jira/payload/common/transfer."
$svc1 = "cachelogd"
$svc2 = "cachemond"
$svc3 = "monlogd"
$build = "C:/Users/jira/go/src/jira/payload/"
$code_tail = { 48 C7 C3 FC FF FF FF B9 02 00 00 00 0F 1F 00 E8 }
$code_shadow = { 48 8D 05 ?? ?? ?? ?? BB 0B 00 00 00 E8 ?? ?? ?? ?? 48 85 FF 74 07 }
condition:
    uint32(0) == 0x464C457F and (
        2 of ($ns*) or
        $build or
        2 of ($svc*) or
        ($code_tail and $code_shadow)
    )
}

```

```

rule Gomir_Linux {
    meta:
        description = "Gomir - Go-based Linux backdoor (Github-Lin/Main/Kernel)"
        date = "2026-06-02"
        hash1 = "b1c72139f2cdd9419562369fc6ced4fc"
        hash2 = "84e9b066bebd49036b7fc71b5f5f8d83"
        hash3 = "e911f8f7c49476806ada37f3ebb7a28a"
        hash4 = "aa61e76255a6e13313439655bc02bdf5"
    strings:
        $ns1 = "local.github.com/Github-Lin/Main/Kernel."
        $ns2 = "local.github.com/Github-Lin/Main/exe"
        $cmd1 = "Not implemented on Linux!"
        $cmd2 = "Unknown Command!\r\n"
        $cmd3 = "Invalid Param!\r\n"
        $persist1 = "/etc/systemd/system/rsyslogd.service"
        $persist2 = "systemctl reenale rsyslogd"
        $code_mod255 = { 48 B8 81 80 80 80 80 80 80 80 49 89 D3 49 F7 E8 }
        $code_sleep = { 48 B8 00 10 A5 D4 E8 00 00 00 E8 ?? ?? ?? ?? EB EF }
        $code_uid = { 48 83 F9 05 7D 3B 0F B6 74 0C 46 89 F7 40 C0 EE 04 }
    condition:
        uint32(0) == 0x464C457F and (
            1 of ($ns*) or
            (2 of ($cmd*) and 1 of ($persist*)) or
            $code_mod255 or
            ($code_sleep and 1 of ($ns*, $persist*)) or
            ($code_uid and 1 of ($ns*, $persist

```



ENKI Whitehat

ENKI Whitehat

Offensive security experts delivering deeper security through an attacker's perspective.