Finding Malware: Detecting SOGU with Google Securi...

googlecloudcommunity.com/gc/Community-Blog/Finding-Malware-Detecting-SOGU-with-Google-Security-Operations/ba-p/758777

June 5, 2024

R

Welcome to the Finding Malware Series

Introducing "Finding Malware," a new blog series from Managed Defense to empower the Google Security Operations community to detect emerging and persistent malware threats. Our first post dives deep into the SOGU malware family and the detection opportunities available within Google SecOps. Happy hunting!

About SOGU

Also known as: PlugX, Korplug

SOGU is a backdoor that supports commands to exfiltrate files, keylogging, remote command shell, upload/download files, and is able to extend its functionality with additional plugins. The backdoor has existed since at least 2008, and is still under continuous development that new variants are constantly being discovered.

SOGU is primarily associated with Advanced Persistent Threat (APT) groups, such as TEMP.Hex, and often used for cyber-espionage.

Attack Lifecycle

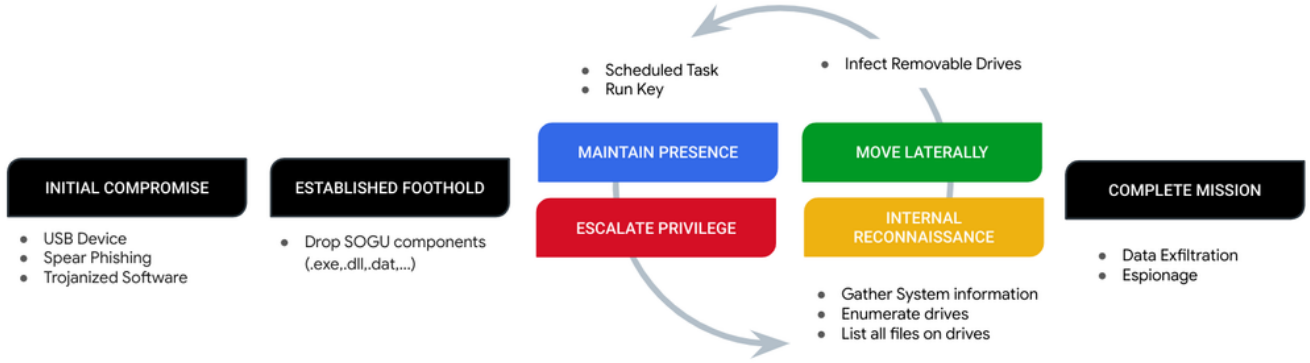


Figure 1: SOGU Attack Lifecycle

Initial Compromise

SOGU spreads through several methods: infected USB flash drives, targeted phishing emails containing malicious attachments or links, or compromised software downloads.

Establish Foothold

The infection consists of three core files: a *legitimate executable*, a *malicious DLL loader*, and an *encrypted SOGU payload*.

Figure 2: SOGU components

Upon execution, the legitimate executable loads a malicious DLL via search-order-hijacking. This loader decrypts a shellcode (often disguised as a .dat file), loads it into memory, and executes it. The shellcode is tracked as **SOGU**.

The executable and .dat files can be renamed to any filename, but the DLL filename is usually fixed and resembles a filename of a legitimate DLL, which the executable expects to load.

Internal Reconnaissance and Data Staging

In some variations of the SOGU malware, a dropped batch file performs host reconnaissance commands. The output of these commands is then saved to a file, often named "c3lzLmluZm8". This filename, when decoded from Base64, reveals the file's true name: "sys.info".

```
tasklist /v
arp -a
netstat -ano
ipconfig /all
systeminfo
```

After host reconnaissance, the malware searches the host for specific file types, including common office documents (e.g., .doc, .docx, .ppt, .pptx, .xls, .xlsx) and PDFs. Upon finding a match, the malware encrypts a copy of the file using the RC4 algorithm. The original filename is then encoded into Base64, and the encrypted file is moved to a designated directory.

- C:\Users\<user>\AppData\Roaming\Intel\<SOGU unique ID>\<filename in Base64>
- <drive>:\RECYCLER.BIN\<SOGU unique ID>\<filename in Base64>

The SOGU unique ID, which also functions as a staging directory, is typically a 16-character identifier consisting of numbers and uppercase letters.

Move Laterally

SOGU has spreading capabilities, it identifies removable drives on an infected host, and propagates to those drives.

The malware creates the following at the root of the drive:

- A new hidden folder with a single space (e.g., <drive>:\ \)
- A new hidden folder called "RECYCLER.BIN" or “RECYCLERS.BIN"
- A Windows shortcut cut file named after the drive (e.g., "My USB Key.lnk")

Maintain Persistence

SOGU malware was observed utilizing scheduled tasks or registry run keys to maintain persistence on infected systems.

It creates a copy of itself masquerading as a legitimate program and sets the directory's attribute to hidden. It then copies its main components into this directory, with the following commonly used file paths:

- C:\ProgramData\<folder name>
- C:\Users\Public\<folder name>
- %APPDATA%\<folder name>

Mission Complete

At the last stage of the attack lifecycle, the malware will exfiltrate any data that has been staged. It is observed using various communication channels to connect with its command-and-control (C2) server. This communication can occur over HTTP, HTTPS, a custom binary protocol using TCP or UDP, or ICMP.

Upon establishing a connection with the C2 server, SOGU enables an attacker to remotely control the infected system. The range of capabilities at the attacker's disposal includes: file transfer, file execution, remote desktop, screenshot capture, reverse shell, and keylogging.

Detection Through Google Security Operations

Enterprise and Enterprise Plus customers will benefit from these detections being applied automatically through curated detections. Standard customers can create single or multi-event rules to detect the malware.

This rule detects the execution of SOGU malware within a known directory.

```
rule sogu_recycler_bin
{
    meta:
        author = "Mandiant"
        description = "This rule matches the process launch event for a binary from
the directory RECYCLER.BIN or RECYCLERS.BIN with numerical arguments."
        mitre_attack_tactic = "User Execution"
        mitre_attack_technique = "User Execution: Malicious File"
        mitre_attack_url = "https://attack.mitre.org/techniques/T1204/002/"
        mitre_attack_version = "v14.1"
        severity = "High"
        priority = "High"
        platform = "Windows"
        type = "hunt"

    events:
            $e.metadata.event_type = "PROCESS_LAUNCH" and
        (
            re.regex($e.target.process.command_line, `(RECYCLER|RECYCLERS)\.BIN[a-zA-
Z0-9\\]{0,30}\.exe [0-9]{3} [0-9]{2}`) nocase or
            re.regex($e.principal.process.command_line, `(RECYCLER|RECYCLERS)\.BIN[a-
zA-Z0-9\\]{0,30}\.exe [0-9]{3} [0-9]{2}`) nocase
        )

    condition:
        $e
}
```

This rule identifies the preparation of the host information for exfiltration by the SOGU malware.

```
rule sogu_sys_info
{
    meta:
        author = "Mandiant"
        description = "This rule matches on a file event for a file with a name
c3lzLmluZm8 that base64-decodes to SYS.INFO. This has been observed in SOGU
compromises, where the file has been observed to contain host-based reconnaissance
data staged for exfiltration."
        mitre_attack_tactic = "Collection"
        mitre_attack_technique = "Data Staged: Local Data Staging"
        mitre_attack_url = "https://attack.mitre.org/techniques/T1074/001/"
        mitre_attack_version = "v14.1"
        severity = "High"
        priority = "High"
        platform = "Windows"
        type = "hunt"

    events:
        (
            $e.metadata.event_type = "FILE_CREATION" or
            $e.metadata.event_type = "FILE_MODIFICATION"
        ) and
        (
            re.regex($e.target.file.names, `c3lzLmluZm8`) nocase or
            re.regex($e.target.file.full_path, `c3lzLmluZm8`) nocase
        )

    condition:
        $e
    }
```

This rule identifies the preparation of the stolen data for exfiltration by the SOGU malware.

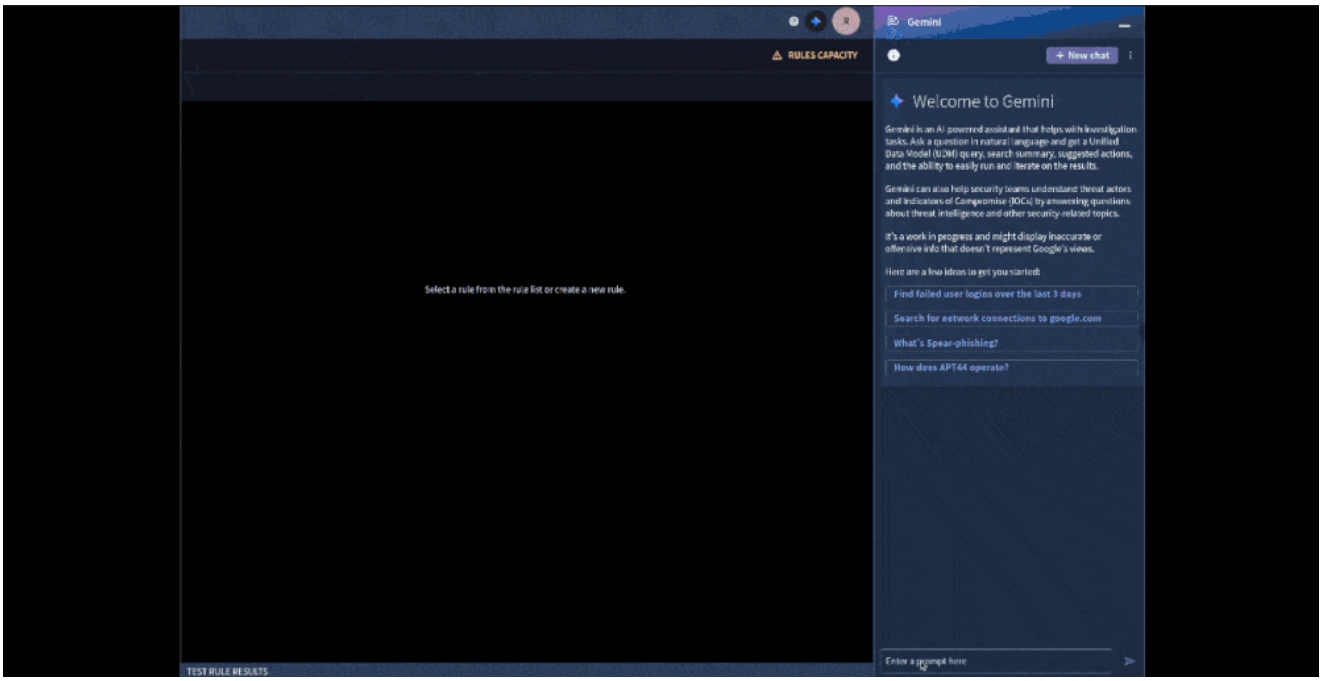
```
rule sogu_data_staging
{
    meta:
        author = "Mandiant"
        description = "This rule matches on directory and filename patterns observed
in data staging for exfiltration, as part of a SOGU compromise. The data staging
directory is typically 16 characters composed of numbers and capital letters, while
the filenames are base64-encoded legitimate filenames from the affected system."
        mitre_attack_tactic = "Collection"
        mitre_attack_technique = "Data Staged: Local Data Staging"
        mitre_attack_url = "https://attack.mitre.org/techniques/T1074/001/"
        mitre_attack_version = "v14.1"
        severity = "High"
        priority = "High"
        platform = "Windows"
        type = "hunt"

    events:
        (
            $e.metadata.event_type = "FILE_CREATION" or
            $e.metadata.event_type = "FILE_MODIFICATION"
        ) and
        (
            $e.target.file.size > 0 and
            re.regex($e.target.file.full_path, `.`) nocase and
            re.regex($e.target.file.full_path, `\\[A-Z0-9]{16}\\[a-zA-Z0-9]{5,}=
{1,3}$`)
        )

    condition:
        $e
}
```

Beyond the Blog: Empower Your SecOps with Gemini

In the ever-changing threat landscape, the ability to respond quickly is key to an effective SecOps. This blog provides insights on the SOGU malware, but it's important to remember that malware is constantly evolving and finding new ways to bypass defenses.

Gemini in Google Security Operations can accelerate our responses by empowering teams to quickly develop new detection rules to counter emerging attack techniques. This helps defenses to be always ready to stop attacks and promptly resolve incidents.

Here's a quick demonstration of how you can easily create a YARA-L rule to detect potential malware compromise using Gemini within Google Security Operations. It took just a few seconds to generate this rule, and now we simply need to review it for accuracy and effectiveness!




```
rule
process_launch_with_command_line_containing_recycler_bin_or_recyclers_bin_and_exe_wit
h_numerical_argument {
  meta:
    author = "Google SecOps Gemini"
    description = "Process launch with command line containing RECYCLER.BIN or
RECYCLERS.BIN and .exe with numerical argument"
    severity = "LOW"
  events:
    $e.metadata.event_type = "PROCESS_LAUNCH"
    re.regex($e.target.process.command_line, `RECYCLER\.BIN|RECYCLERS\.BIN`) nocase
    re.regex($e.target.process.command_line, `\.exe\s+\d+`) nocase
  outcome:
    $process_name = $e.target.process.file.full_path
    $command_line = $e.target.process.command_line
    $user = $e.principal.user.userid
  condition:
    $e
}
```

The rule created by Gemini.

More information, please check out the documentation on Gemini in Google SecOps!