# Enhanced Attack Report

## Small Sieve

*Generated on 2025-03-07*

## Disclaimer

## Definitions

**Pre-Conditions:** Conditions that must be true to execute the attack steps in the milestone.

**Post-Conditions:** Traces that an attacker leaves behind after executing the attack steps in the milestone.

**Attack Steps:** Steps that an attacker would take to achieve the goal of the milestone.

**MITRE Technique:** Techniques from the MITRE ATT&CK; framework that are relevant to the milestone.

## STIX

**Malware Name:** Small Sieve
**Malware Description:** Small Sieve is a simple â€“ possibly disposable â€“ Python backdoor which is distributed using an NSIS installer that performs persistence. It provides basic functionality required to maintain and expand a foothold in victim infrastructure using custom string and traffic obfuscation schemes together with the Telegram Bot API to avoid detection.

## Quick Overview

**Milestone 1**
1. Command and Scripting Interpreter: Python as used by the malware (T1059.006)

**Milestone 2**
1. Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder as used by the malware (T1547.001)

**Milestone 3**
1. Obfuscated Files or Information as used by the malware (T1027)
2. Execution Guardrails as used by the malware (T1480)
3. Masquerading: Match Legitimate Name or Location as used by the malware (T1036)

**Milestone 4**
1. Application Layer Protocol: Web Protocols as used by the malware (T1071.001)
2. Data Encoding: Non-Standard Encoding as used by the malware (T1132.002)

# Milestone 1

## Pre-Conditions:

- The system has a Python script named Out1 available.
- Tools: Python, PyInstaller, Telegram Bot API, urllib module.
- The system has a Python interpreter installed.
- The system has a network connection to the Telegram Bot API.
- The system has a registry run key available for the malware to start.
- The system has a PyInstaller-packed Python script named Small Sieve available.
- The system has a network connection to the internet.
- The Telegram Token 2003026094: AAGoitvpcx3SFZ2_6YzIs4La_kyDF1PbXrY is available.

## Post-Conditions:

- The victim's IP address and domain name are collected.
- Credentials from Web Browsers are stolen.
- Python tools are used to execute the POWERSTATS payload.
- VBS files are used to execute the POWERSTATS payload.
- Credentials from Password Stores are dumped.
- LaZagne tool logs.
- Skype connections on the target machine are checked.
- InvokeScript method files.
- PowerShell backdoor logs.
- Browser64 tool logs.
- The Office vulnerability CVE-2017-0199 is exploited.
- The victim's OS version and machine name are collected.
- Python urllib module files.
- PyInstaller-packed Python script files.
- The Small Sieve script is started by a registry run key.
- Registry run key logs.
- A backdoor is established using a registry run key.
- A list of running processes is obtained.
- Small Sieve script files.

## Attack Step 1.1

==================================================
**Name:** Command and Scripting Interpreter: Python as used by the malware
**Description:** MuddyWater utilizes a PyInstaller-packed Python script as its malware. This script leverages the Python programming language to execute various malicious functions. Let me know if you have any other actions you'd like me to elaborate on!

### *MITRE Technique*

**ID:** T1059.006
**Name:** Command and scripting interpreter: python
**Description:** Adversaries may abuse Python commands and scripts for execution. Python is a very popular scripting/programming language, with capabilities to perform many functions. Python can be

executed interactively from the command-line (via the python.exe interpreter) or via scripts (.py) that can be written and distributed to different systems. Python code can also be compiled into binary executables.
**More info:** https://attack.mitre.org/techniques/T1059/006

## *Indicators*

No indicators found.

# Milestone 2

## Pre-Conditions:

- The malware has been installed on the system.
- The system has a startup folder.
- The system has a registry.
- Connectivity to the system.
- A user account with access to the registry and startup folder.
- The malware has the necessary permissions to modify the registry.
- Windows operating system.
- The system has a user account with administrative privileges.

## Post-Conditions:

- Scheduled tasks are used to establish persistence
- Credential dumping with LaZagne
- Scheduled task logs
- Browser64 logs
- Registry key HKCU\Software \Microsoft \Windows\CurrentVersion\Run\OutlookMicrosift is added
- Small Sieve is started by a registry run key
- List of running processes on the system is obtained
- index.exe is installed in the user's AppData/Roaming directory
- "Platypus" argument is present in the registry persistence key
- LaZagne logs
- Victim's OS version and machine name are collected
- Python backdoor, index.exe, is installed and added as a registry run key
- Run key in the registry is added to enable persistence after reboot
- Word Template, Normal.dotm, is used for persistence
- Credential dumping with Browser64
- Registry key HKCU\Software \Microsoft \Windows\CurrentVersion\Run\SystemTextEncoding is added
- Passwords saved in victim email are stolen

## Attack Step 2.1

==================================================
**Name:** Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder as used by the malware
**Description:** The malware establishes persistence by adding a registry run key. Here's a breakdown of how it works based on the provided context: Registry Run Key Creation: The malware creates a new entry in the Windows Registry under a specific path, typically HKCU\Software\Microsoft\Windows\CurrentVersion\Run. Key Name: The malware assigns a name to this registry key, often disguised as a legitimate Windows component or service. This helps it blend in and avoid detection. Value: The value associated with this registry key points to the location of the malware's executable file. Autostart Execution: When the victim starts their computer or logs in, Windows automatically reads the registry keys under HKCU\Software\Microsoft\Windows\CurrentVersion\Run. Because the malware's executable is linked to this key, it gets executed alongside legitimate system processes. Let me know if you'd like more details on any specific aspect of this process!

## *MITRE Technique*

**ID:** T1547.001
**Name:** Boot or logon autostart execution: registry run keys / startup folder
**Description:** Adversaries may achieve persistence by adding a program to a startup folder or referencing it with a Registry run key. Adding an entry to the "run keys" in the Registry or startup folder will cause the program referenced to be executed when a user logs in. These programs will be executed under the context of the user and will have the account's associated permissions level.
**More info:** https://attack.mitre.org/techniques/T1547/001

## *Indicators*

- Registry value name: HKCU\Software\Microsoft\Windows\CurrentVersion\Run\OutlookMicrosift:
- Path: %AppData%\OutlookMicrosift\index.exe:

# Milestone 3

## Pre-Conditions:

- Resources: A system with a processor that can perform byte swapping and Base64 encoding, and a memory that can store obfuscated data.
- Connectivity: A network connection to the internet.
- Tools: A custom hex byte swapping encoding scheme, an obfuscated Base64 function, and a Telegram API.
- The malware has access to Hypertext Transfer Protocol Secure (HTTPS).
- Operating System: A system with an operating system that supports HTTPS.
- Access to HTTPS: Access to a system with a network connection that supports HTTPS.
- Access to Telegram API: Access to a Telegram API to send and receive data.
- The malware has access to a Telegram API.
- The malware has access to a custom hex byte swapping encoding scheme.
- The malware has access to a system with a memory that can store obfuscated data.
- Secure Storage: A system with a secure storage to store obfuscated data.
- Environment: A system with a network connection and a secure storage.
- The system has a network connection.
- The system's local proxy settings are enabled.
- The word "Platypus" is present on the command line.
- The system has the urllib module available.
- Availability of the malware (Small Sieve payload).
- The system has a command line interface.
- The system has a Python interpreter.
- Tools: Python interpreter, urllib module.
- Resources: System memory and storage to run the malware.
- Availability of the Telegram Bot API.
- The malware is executed correctly.
- The malware has the ability to disguise its malicious executables.
- The malware has the ability to create and modify files with legitimate names.
- A reliable method to update the malware's configuration and updates.
- A Windows-based system with access to the system's file system and registry.
- A mechanism to evade detection by security software and human analysts.
- A list of legitimate file names and registry key names.
- A mechanism to modify the system's Windows Defender configuration.

## Post-Conditions:

- Credentials from Web Browsers are stolen.
- Process Discovery is performed.
- Credentials from Password Stores are dumped.
- System Network Configuration Discovery is performed.
- LaZagne tool logs.
- PowerShell backdoor logs.
- Obfuscated Files or Information are used.
- Encoded MicrosoftWindowsOutlookDataPlus.txt file.
- PowerShell scripts are obfuscated.
- Browser64 tool logs.
- System Owner/User Discovery is performed.

- Malware logs.
- Skype connections are checked.
- Hex byte swapping encoding scheme logs.
- Telegram API logs.
- Obfuscated Base64 function logs.
- cmd.exe logs.
- Program strings and Telegram credentials are protected.
- Obfuscated JavaScript code is stored in an image file.
- HTTP S logs.
- A connection to Telegram is terminated.
- Custom hex byte swapping encoding scheme used to obfuscate tasking traffic.
- Telegram Bot API logs.
- Execution Guardrails used to ensure correct payload execution.
- Network connections to URLs used for downloading files.
- Malware executable files.
- Logs of commands issued using the /com[BotID] format.
- Encoded tasking traffic logs.
- Logs of Python urllib module used to download files.
- Malware used to collect the victim's OS version and machine name.
- Modified system configuration files to disable local proxy settings.
- Malware used to obtain a list of running processes.
- Credential dumping with LaZagne.
- Credential dumping with Browser64.
- Downloaded files from URLs using the download url””filename command.
- Small Sieve payload used to steal sensitive information.
- Collecting user data outside the cookie realm.
- Collecting the victim's OS version and machine name.
- Displaying fake CS2 skin and cryptocurrency giveaways on YouTube.
- Using Telegram API over HTTPS for command and control.
- Credential dumping with other tools.
- YouTube account hijacking logs.
- Hijacking YouTube accounts to impersonate professional players.
- Installing malware on compromised systems.
- Using variations of Microsoft and Outlook in filenames to avoid detection.
- Obtaining a list of running processes on the system.
- Stealing passwords saved in victim email.
- Dodging detection through traffic obfuscation and custom string schemes.
- Modified system files with malware persistence.
- Credential dumping with LaZagne.
- Logs of stealing passwords saved in victim web browsers.
- Granting attackers backdoor capabilities on compromised systems.
- Compromising user devices through fingerprinting.
- Bypassing user account control.
- Disguising malicious executables and using filenames and Registry key names associated with Windows Defender.
- Performing malicious livestreams on YouTube.

## Attack Step 3.1

================================================
**Name:** Obfuscated Files or Information as used by the malware

**Description:** MuddyWater uses a custom hex byte swapping encoding scheme combined with an obfuscated base64 function to protect program strings and updated Telegram credentials. This technique, known as obfuscation, makes the malware's code and data harder to understand and analyze by security researchers and antivirus software. Let me know if you have any other actions you'd like me to explain!

### *MITRE Technique*

**ID:** T1027
**Name:** Obfuscated files or information
**Description:** Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit. This is common behavior that can be used across different platforms and the network to evade defenses.
**More info:** https://attack.mitre.org/techniques/T1027/

### *Indicators*

- Small Sieve sample (Filename: gram_app.exe)
- Small Sieve sample (Filename: index.exe)
- Path: %LocalAppData%\MicrosoftWindowsOutlookDataPlus.txt
- Registry value name: HKCU\Software\Microsoft\Windows\CurrentVersion\Run\OutlookMicrosift
- Path: %AppData%\OutlookMicrosift\index.exe

## Attack Step 3.2

==================================================
**Name:** Execution Guardrails as used by the malware
**Description:** The malware, Small Sieve, implements an execution guardrail by requiring the word "Platypus" to be passed as a command-line argument for it to execute correctly.

### *MITRE Technique*

**ID:** T1480
**Name:** Execution guardrails
**Description:** Adversaries may use execution guardrails to constrain execution or actions based on adversary supplied and environment specific conditions that are expected to be present on the target. Guardrails ensure that a payload only executes against an intended target and reduces collateral damage from an adversary’s campaign. Values an adversary can provide about a target system or environment to use as guardrails may include specific network share names, attached physical devices, files, joined Active Directory (AD) domains, and local/external IP addresses.
**More info:** https://attack.mitre.org/techniques/T1480/

### *Indicators*

- Small Sieve sample (Filename: gram_app.exe)
- Small Sieve sample (Filename: index.exe)
- Path: %LocalAppData%\MicrosoftWindowsOutlookDataPlus.txt
- Path: %AppData%\OutlookMicrosift\index.exe

## Attack Step 3.3

==================================================

**Name:** Masquerading: Match Legitimate Name or Location as used by the malware
**Description:** MuddyWater malware attempts to evade detection by using filenames and Registry key names that closely resemble legitimate Microsoft (Microsift) and Outlook components. For example, it might name a malicious executable "Microsoft.exe" or create a Registry key named "OutlookData" to blend in with existing system files and processes. This tactic relies on attackers hoping that security analysts or users will quickly overlook these subtle variations during a cursory inspection.

## *MITRE Technique*

**ID:** T1036
**Name:** Masquerading
**Description:** Adversaries may attempt to manipulate features of their artifacts to make them appear legitimate or benign to users and/or security tools. Masquerading occurs when the name or location of an object, legitimate or malicious, is manipulated or abused for the sake of evading defenses and observation. This may include manipulating file metadata, tricking users into misidentifying the file type, and giving legitimate task or service names.
**More info:** https://attack.mitre.org/techniques/T1036/

## *Indicators*

- Small Sieve sample (Filename: gram_app.exe)
- Small Sieve sample (Filename: index.exe)
- Path: %LocalAppData%\MicrosoftWindowsOutlookDataPlus.txt
- Registry value name: HKCU\Software\Microsoft\Windows\CurrentVersion\Run\OutlookMicrosift
- Path: %AppData%\OutlookMicrosift\index.exe

# Milestone 4

## Pre-Conditions:

- The malware has access to the internet.
- The malware has access to the Telegram Bot API.
- The malware has the capability to prefix commands with /com[Bot ID].
- Environment: A network environment with internet connectivity.
- The malware has been deployed on the victim's machine.
- The Small Sieve tool is configured to use a custom hex byte swapping encoding scheme.
- Environment: A Windows-based system is required.
- The Small Sieve tool is available in the environment.
- The Small Sieve tool is not blocked by any security measures.
- Resources: The malware requires sufficient resources to execute the Small Sieve tool.

## Post-Conditions:

- Files created by Divine Mathematician Password Cracking Platform.
- HTTP is used for C2 communications.
- LaZagne tool logs.
- Skype connections on the target machine are checked.
- Victim's OS version and machine name are collected.
- Files created by LaZagne (e.g., MicrosoftWindowsOutlookDataPlus.txt).
- Small Sieve beacons and tasking are performed using the Telegram API over HTTPS.
- Telegram Bot API logs.
- Logs of spear phishing attacks on Twitter.
- PowerStats is controlled from behind a proxy network to obfuscate the C2 location.
- Credentials and session values are stored in a table.
- Encoded files containing updated tokens.
- Files created by Automated Penetration Testing Platform.
- PowerStats logs.
- Process discovery logs.
- Passwords saved in victim email are stolen.
- Small Sieve uses variations of Microsoft (Microsift) and Outlook in its filenames to attempt to avoid detection.
- Small Sieve beacons and tasking logs.
- Twitter accounts are taken over and controlled.
- Proxy network logs.
- Logs of Automated Penetration Testing Platform.
- A list of running processes on the system is obtained.
- Logs of Public Opinion Guidance and Control Platform (Overseas).
- Victim's IP address and domain name are collected.
- Credentials from Password Stores are dumped with LaZagne and other tools.
- Network connections to Telegram API.
- Files created by Public Opinion Guidance and Control Platform (Overseas).
- Command and control protocol logs.
- The victim's IP address and domain name are collected.
- Credentials from Web Browsers are stolen.
- Files are created on the victim's machine containing the Small Sieve payload and its configuration.
- LaZagne tool is used to dump credentials from Password Stores.

- Credentials from Password Stores are dumped.
- Small Sieve payload is executed on the victim's machine.
- LaZagne tool is installed on the victim's machine.
- Malware is used to collect the victim's IP address and domain name.
- Skype connections on the target machine are checked.
- HTTPS connections are established between the Small Sieve payload and the Telegram API.
- Browser64 tool is installed on the victim's machine.
- A list of running processes is obtained.
- The victim's OS version and machine name are collected.
- Small Sieve beacons and taskings are obfuscated through a hex byte swapping encoding scheme combined with an obfuscated Base64 function.
- Tasking traffic is obfuscated through a custom hex byte swapping encoding scheme.

## Attack Step 4.1

====================================================
**Name:** Application Layer Protocol: Web Protocols as used by the malware
**Description:** The malware, specifically the variant called "Small Sieve," uses the Telegram API over HTTPS for its Command and Control (C2) communications. This means: Beacons: The malware periodically sends small messages (beacons) to the Telegram servers to check in and receive instructions. Tasking: The attackers use the Telegram API to send commands to the infected system, instructing it to perform various malicious actions. HTTPS: The communication between the malware and the Telegram servers is encrypted using HTTPS, making it more difficult to intercept and analyze. This technique allows MuddyWater to maintain a covert presence on compromised systems and control their actions without easily detectable network traffic.

### *MITRE Technique*

**ID:** T1071.001
**Name:** Application layer protocol: web protocols
**Description:** Adversaries may communicate using application layer protocols associated with web traffic to avoid detection/network filtering by blending in with existing traffic. Commands to the remote system, and often the results of those commands, will be embedded within the protocol traffic between the client and server.
**More info:** https://attack.mitre.org/techniques/T1071/001

### *Indicators*

No indicators found.

## Attack Step 4.2

====================================================
**Name:** Data Encoding: Non-Standard Encoding as used by the malware
**Description:** MuddyWater uses malware that employs a custom hex byte swapping encoding scheme to obfuscate tasking traffic.

### *MITRE Technique*

**ID:** T1132.002
**Name:** Data encoding: non-standard encoding

**Description:** Adversaries may encode data with a non-standard data encoding system to make the content of command and control traffic more difficult to detect. Command and control (C2) information can be encoded using a non-standard data encoding system that diverges from existing protocol specifications. Non-standard data encoding schemes may be based on or related to standard data encoding schemes, such as a modified Base64 encoding for the message body of an HTTP request.
**More info:** https://attack.mitre.org/techniques/T1132/002

### *Indicators*

No indicators found.