Enhanced Attack Report

Goofy Guineapig

Generated on 2025-03-05

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: Goofy Guineapig
Malware Description: The Goofy Guineapig loader is a UPX packed, trojanised NSIS Firefox installer. Once extracted, it masquerades as a Google update component. Goofy Guineapig maintains persistence as a Windows service. Goofy Guineapig provides a framework into which additional plugins may be loaded. The backdoor supports multiple communications methods, including HTTP, HTTPS and KCP. The configuration is embedded in the binary, and the configuration for the binary analysed results in command and control communications occurring over HTTPS. Many defence evasion techniques are implemented throughout execution.

Quick Overview

Milestone 1
1. Create or Modify System Process: Windows Service as used by the malware (T1543.003)

Milestone 2
1. Masquerading: Match Legitimate Name or Location as used by the malware (T1036)
2. Virtualization/Sandbox Evasion: Time Based Evasion as used by the malware (T1497.003)
3. Virtualization/Sandbox Evasion: System Checks as used by the malware (T1497.001)
4. Virtualization/Sandbox Evasion: User Activity Based Checks as used by the malware (T1497.002)
5. Obfuscated Files or Information: Software Packing as used by the malware (T1027)
6. Deobfuscate/Decode Files or Information as used by the malware (T1140)
7. Hide Artifacts: Hidden Window as used by the malware (T1564)
8. Indicator Removal on Host: File Deletion as used by the malware (T1070.010)
9. Hijack Execution Flow: DLL Side-Loading as used by the malware (T1574.001)
10. Process Injection: Process Hollowing as used by the malware (T1055.012)
11. Signed Binary Proxy Execution: Rundll32 as used by the malware (T1218.010)

Milestone 3
1. System Information Discovery as used by the malware (T1049)

Milestone 4
1. Application Layer Protocol: Web Protocols as used by the malware (T1071)
2. Fallback Channels as used by the malware (T1573)
3. Non-Standard Port as used by the malware (T1090.001)

Milestone 1

Pre-Conditions:

- The ability to collect adapters information, host and host name, and computer name information.
- Access to COM interface to access WMI information.
- A UPX packed NSIS installer (trojanised Firefox installer) is available.
- Access to physical memory and disk size information.
- The malware is running on a Windows system.
- The malware has access to the Windows APIs.
- Windows operating system.
- The ability to execute the binary from the location: C:\ProgramData \GoogleUpdate.
- The ability to start a service for persistence.
- The ability to collect operating system caption and Antivirus product display name information.
- The Goopdate.dll DLL is available.
- The malware has access to the COM interface to access WMI information.
- Windows service creation and modification capabilities.
- The malware has access to the physical memory and disk size information.
- The ability to send information about the victim machine over the internet.
- Availability of the necessary permissions to create or modify system processes.
- The malware has the necessary permissions to create or modify system processes.
- The Windows service details are available in Table 6.
- Access to Windows APIs.
- The malware is not running in a debugger or sandbox environment.
- The Config.dat file is available.

Post-Conditions:

- Network traffic logs showing the malware sending information about the victim machine.
- Files created by the malware for storing information about the victim machine.
- Windows service created by the malware.
- Files are moved to a legitimate looking directory and deleted from the initial download location.
- Malware is bundled in a UPX packed NSIS installer which is a trojanised Firefox installer.
- Malware checks if it is running from a specific location and starts a service if not.
- COM access logs for WMI information collection.
- Malware communicates over a non-standard HTTPS port 4443.
- Persistence mechanism is created using a Windows service.
- Malware sends information about the victim machine to an unknown location.
- Windows API logs for information collection.
- Registry entries created by the malware for persistence mechanism.
- Network connections to a non-standard HTTPS port 4443.
- Malware is executed without a persistence mechanism.
- Files created by the malware, including the UPX packed NSIS installer.
- Malware collects information about the victim machine, including operating system caption, antivirus product display name, adapters information, host and host name, and computer name.
- Malware collects information about the victim machine using COM to access WMI information and relevant Windows APIs.
- Logs of the malware's activity, including information collection and persistence mechanism creation.
- Malware implements basic anti-sandbox/anti-VM techniques.

Attack Step 1.1

==

Name: Create or Modify System Process: Windows Service as used by the malware
Description: The malware maintains persistence as a Windows service by creating a new Windows service. Here's a breakdown of how this likely works: Service Creation: The malware uses the Windows API functions to create a new service. This involves specifying a service name, display name, description, and executable file (likely a modified version of itself). Service Configuration: The malware configures the service to start automatically on system boot and potentially set other parameters like startup type, error control, and account under which it runs. Service Registration: The malware registers the newly created service with the Windows service manager. This makes the service visible to the operating system and allows it to be started and controlled. Persistence: Once the service is registered, it will automatically start when the system boots, ensuring the malware continues to run in the background even after user sessions end or the system restarts. Let me know if you'd like more details on specific Windows API functions involved in this process.

MITRE Technique

ID: T1543.003
Name: Create or modify system process: windows service
Description: Adversaries may create or modify Windows services to repeatedly execute malicious payloads as part of persistence. When Windows boots up, it starts programs or applications called services that perform background system functions. Windows service configuration information, including the file path to the service's executable or recovery programs/commands, is stored in the Windows Registry.
More info: https://attack.mitre.org/techniques/T1543/003

Indicators

- Path: C:\ProgramData\GoogleUpdate\GoogleUpdate\tmp.bat

Milestone 2

Pre-Conditions:

- The malware has the capability to check the system's disk size.
- Resources:
- The malware has a list of legitimate process names to compare with.
- Tools: None explicitly stated, but the malware likely uses built-in system calls or libraries to perform the action.
- The malware has the capability to execute system calls or use libraries to interact with the system.
- Environment: Windows operating system.
- The system has a disk size greater than 1GB.
- The malware has access to the system's process list.
- Connectivity: None required.
- The system has a physical memory size exceeding 2GB.
- The time difference between the two registrations is less than 100 milliseconds.
- The system is not in a state of suspended animation or paused execution.
- The system has a number of logical processors exceeding 2.
- A system with a functioning time measurement mechanism (e.g., a clock or timer).
- A system with a disk size exceeding 1GB.
- The malware has access to the system's time registration and measurement mechanisms.
- The system is not in a state of dormancy or hibernation.
- A system with a functioning system time registration mechanism.
- The malware has the necessary resources (e.g., CPU, memory) to perform the time-based evasion checks.
- The system time is registered twice.
- The malware is running on the system.
- A system with a logical processor.
- The system has physical memory.
- A system with a physical memory size greater than 2GB.
- The malware has the necessary code to perform the system checks.
- The system is not running a debugger.
- The system has a disk.
- The system is not running other analysis tools.
- The system has a disk size greater than 1GB.
- The system has a physical memory size exceeding 2GB.
- The system has processes running.
- The system is not being debugged.
- Availability of the Goopdate.dll DLL.
- Environment: A Windows system with a physical memory size exceeding 2GB and a disk size exceeding 1GB.
- Tools: None explicitly stated, but likely requires a debugger or analysis tool to detect.
- Connectivity: None explicitly stated, but likely requires network connectivity to communicate with the C2 server.
- The system has a GoogleUpdate directory.
- The system is not running in an automated analysis environment.
- The system has a Firefox installer.
- Resources: None explicitly stated, but likely requires system resources to run processes and check system properties.
- The system has a UPX packed NSIS installer.

- The system has system time.
- The system is not running in a sandbox or virtual machine environment.
- The system has a user interacting with it.
- The system has properties.
- The system is not being reverse engineered.
- The malware has the necessary resources to perform the packing operation.
- Environment: A Windows environment with a 32-bit or 64-bit architecture.
- Tools: UPX packing tool, a legitimate NSIS installer.
- Connectivity: None required.
- Knowledge: The malware must have knowledge of the UPX packing tool syntax and the necessary permissions to pack the software.
- The malware has the necessary privileges to modify the software's file system attributes.
- The malware has the necessary permissions to pack the software.
- Resources: Available disk space, sufficient memory to perform the packing operation.
- The software to be packed is available.
- The malware has access to the UPX packing tool.
- The environment supports the execution of the UPX packing tool.
- The malware is running on a system with a physical memory size exceeding 2GB.
- The malware binary must be available for analysis.
- Environment: A system with a physical memory size exceeding 2GB and a disk size exceeding 1GB.
- A tool capable of analyzing NSIS installers must be available.
- Resources: Access to the malware binary, access to the system where the malware is running, and a tool capable of analyzing the system's memory and disk.
- A tool capable of analyzing the system's memory and disk must be available.
- The malware is UPX packed and packaged in with a legitimate NSIS installer.
- A tool capable of deobfuscating single byte XOR or subtraction must be available.
- Tools: A tool capable of deobfuscating single byte XOR or subtraction, a tool capable of unpacking UPX packed files, and a tool capable of analyzing NSIS installers.
- A tool capable of unpacking UPX packed files must be available.
- The malware checks the name of each running process on the machine.
- The malware contains a hardcoded configuration string under a single byte XOR with the key 0x59.
- The malware contains stack-based strings which are obfuscated with single byte XOR or subtraction throughout the binary.
- Connectivity: None.
- Resources: A legitimate dllhost.exe process, a legitimate process path and name for dllhost.exe, and sufficient memory and CPU resources to perform process hollowing and window hiding.
- The machine must have a legitimate dllhost.exe process that can be used as a cover process.
- The machine has a legitimate process path and name for dllhost.exe.
- Connectivity: None explicitly stated, but the malware likely requires network connectivity to communicate with its C2 server.
- The malware has the option to perform process hollowing.
- Tools: None explicitly stated, but the malware likely requires some form of process manipulation and window hiding capabilities.
- The machine must have a legitimate dllhost.exe process that can be used to hide the malware's presence.
- The malware is running on a machine.
- The ability to download and execute files.
- The initial directory to which the files were downloaded is accessible.
- The files are moved to a legitimate looking directory.
- The ability to move files.
- The malware has the capability to check the physical memory size of the machine.
- The malware has the capability to delete files.

- The malware is initially run in the location to which it is downloaded.
- The malware has the capability to move files.
- The ability to check the window text of the process.
- The directory containing the extracted Firefox files is accessible.
- A legitimate looking directory to move the files to.
- The ability to check the disk size.
- The ability to delete files.
- A Windows environment with a physical memory size exceeding 2GB.
- The malware has the capability to check the window text of the process.
- A network connection to download the malware.
- The ability to check the physical memory size of the machine.
- The malware has the capability to check the disk size.
- The ProgramData directory is accessible.
- A network connection to download the Firefox files.
- The presence of the malware.
- The legitimate executable is signed.
- Connectivity:
- The UPX packed NSIS installer is a trojanised Firefox installer.
- Tools:
- Resources:
- A malicious DLL is loaded by the legitimate executable.
- The Goofy Guineapig loader is present.
- Additional requirements:
- The legitimate executable is a UPX packed NSIS installer.
- The legitimate executable is a GoogleUpdate.exe file.
- Environment: Windows operating system.
- A legitimate executable is installed by the Goofy Guineapig loader.
- The malicious DLL is Goopdate.dll.
- The system has a legitimate dllhost.exe process.
- The malware has access to the system's CPU timestamp counter.
- Environment: A Windows-based system with a legitimate dllhost.exe process and a legitimate GoogleUpdate.exe executable.
- Tools: None explicitly stated, but inferred tools are:
- Resources:
- The system has a legitimate FireFox NSIS installation package.
- The system has a malicious DLL that can be sideloaded.
- The system has a legitimate GoogleUpdate.exe executable.
- The system has a legitimate rundll32.exe executable.
- The malware has been successfully installed on the system.
- Connectivity: None explicitly stated, but inferred connectivity is:
- The system has a legitimate url.dll library.
- Presence of malicious DLL.
- Availability of legitimate FireFox files for masquerading as legitimate processes.
- Availability of GoogleUpdate.exe (signed executable) for sideloading the malicious DLL.
- Availability of legitimate FireFox NSIS installation package for masquerading as legitimate processes.
- Physical memory size of the machine exceeds 2GB.
- The malware has the necessary permissions to execute the rundll32.exe binary.
- Network connectivity for C2 communication.
- The malware has been successfully executed.
- Windows operating system.
- The system is not running in a sandbox or virtual machine environment.
- The malware has access to the Windows API.

- Disk size of the machine is more than 1GB.
- The url.dll binary is present on the system.

Post-Conditions:

- Implementation of basic anti-sandbox/anti-VM techniques.
- Network connections established by the malware.
- Network connections established by the legitimate signed executable file GoogleUpdate.exe.
- Logs of the anti-sandbox/anti-VM techniques implemented by the malware.
- Logs of the Windows service created by the malware.
- Checking of the physical memory size and disk size of the machine.
- Checking of the name of each running process on the machine.
- Decoding of shellcode from config.dat.
- Files created by the decoding of binary extracted from the shellcode.
- Creation of a trojanised Firefox installer.
- Creation of a UPX packed NSIS installer.
- Creation of a malicious DLL masquerading as Google update.
- Files created by the Goopdate.dll.
- Decoding of binary extracted from the shellcode.
- Persistence of the malware as a Windows service.
- Logs of the loading of the malicious DLL Goopdate.dll by GoogleUpdate.exe.
- Deletion of files from the initial download location.
- Network connections to the non-standard HTTPS port 4443.
- Modification of system time checks for evasion.
- Log entries of file deletion and movement.
- Potential creation of a debugger or analysis tools.
- Comparison of physical memory size and disk size in system logs.
- Persistence of a malicious DLL (Goofy Guineapig) in the system.
- Creation of a Windows service for persistence.
- Implementation of anti-sandbox/anti-VM techniques.
- Comparison of physical memory size and disk size for evasion.
- Comparison of logical processor count for evasion.
- Comparison of logical processor count in system logs.
- Log entries of malware exit due to evasion checks failing.
- Movement of files to a legitimate-looking directory.
- Exit of the malware if any evasion checks fail.
- Files created or modified by the malware (e.g. YARA rule files).
- Malware executable files left behind on the system.
- Network connections to the malware's C2 server.
- Processes running on the system checked for reverse engineering or debugging.
- Files created or modified by the malware's sandbox evasion techniques.
- Malware exits if any of the checks fail.
- Files deleted from the directory containing the extracted Firefox files.
- Process checks logs (reverse engineering or debugging).
- Name of each running process on the machine checked.
- Logs of persistence mechanism installation.
- Registry keys modified by the malware.
- Files copied to the ProgramData directory.
- Malware exits if any process containing the string ‘dbg’, ‘debug’, or ‘ida’ is determined to be running.
- Network connections to the API call arguments detection.
- System configuration files modified by the malware (e.g. Windows Defender settings).

- Persistence mechanism installed on the system.
- System logs of the malware's execution (e.g. Windows Event Viewer logs).
- Internet set option API call arguments detected in Goofy Guineapig.
- Malicious files copied to the ProgramData directory.
- Disk size, physical memory size, and number of logical processors checked.
- Files downloaded by the malware.
- Deletion of files from the initial download location.
- Log files indicating the creation of a legitimate-looking directory.
- Logs indicating the starting of a service for persistence.
- Persistence of a malicious Windows service.
- Creation of a legitimate-looking directory to store deleted files.
- Logs indicating the enumeration of logon sessions.
- Checking of process names for indication of debugging or reverse engineering.
- Querying for usernames and client protocol types associated with logon sessions.
- Checking of system time for indication of automated analysis environment.
- Logs indicating the presence of multiple mistakes throughout the binary.
- Logs indicating the checking of system properties for indication of automated analysis environment.
- Creation of a UPX packed NSIS installer.
- Starting of a service for persistence.
- Enumeration of logon sessions on the infected machine.
- Trojanisation of a Firefox installer.
- Communication over a non-standard HTTPS port (4443).
- Logs indicating the querying of usernames and client protocol types.
- Poor coding/testing practices and Op Sec.
- Presence of multiple mistakes throughout the binary.
- Network connections to the developer.mozilla.org website.
- Files created by the Goopdate.dll DLL.
- User activity based checks for reverse engineering or debugging.
- The malware checks for the presence of debuggers and analysis tools.
- Network connections made by the malware, including connections to the command and control server.
- Logs of the malware's deletion of files from the initial download location.
- The malware deletes files from the initial download location.
- The malware masquerades as legitimate processes.
- Files created by the malware, including the packed and obfuscated binary.
- Registry entries created by the malware to check for the presence of specific processes.
- The malware is sideloaded by a legitimate executable.
- The malware checks for the presence of sandbox or virtual machine environments.
- Files created by the malware to sideload the malicious DLL.
- Network connections made by the malware to download additional malware components.
- The malware is installed on the system.
- The malware is packed and obfuscated.
- The malware checks for the presence of processes containing specific strings.
- Logs of malware execution halted due to specific process window text.
- Malware execution halted due to presence of debugging tools.
- Deletion of files from the initial download location.
- Presence of RC4 encrypted files or communications.
- RC4 encryption of embedded binaries and C2 communications.
- Malware execution halted due to presence of specific process names.
- Presence of UPX packed files.
- Creation of a legitimate-looking directory to store files.
- Obfuscation of files or information using UPX packing and single-byte XOR.
- Logs of automated randomization of XOR keys at build or deployment time.

- HTTPS communications over non-standard port 4443.
- Logs of deleted files from the initial download location.
- Presence of C2 communications over HTTPS and RC4 encrypted.
- Persistence of a malicious Windows service.
- Hiding of malware execution.
- Malware communication over non-standard HTTPS port 4443.
- Creation of a hidden process.
- Logs of process creation and modification.
- Files created by the malware in legitimate-looking directories.
- Files deleted from initial download location.
- Evasion of sandbox and virtual machine detection.
- Network connections over port 4443.
- Malicious DLL loading.
- Windows service logs.
- Modified process listings.
- Modification of process listings to hide malware.
- Malware persistence through process hollowing.
- Logs of malware communication.
- Malware execution despite debugger presence.
- Removal of malware artifacts.
- Persistence of a malicious Windows service.
- Malicious files are copied to the ProgramData directory.
- Deleted files in the initial download location.
- Network connections made by the malware.
- The initial directory to which the files were downloaded will only contain the files the recipient intended to download.
- Files in the ProgramData directory.
- System calls made by the malware.
- Logs of the malware's activity.
- Registry entries created by the malware.
- The malware checks for process name and window text to evade detection.
- Process creation and termination logs.
- The malware checks for physical memory size and disk size to evade detection.
- The secondary check is likely to be ineffective due to flawed logic.
- Memory dumps of the malware.
- The malware implements basic anti-sandbox/anti-VM techniques.
- The malware may trigger process exit if the window text is caught.
- The ProgramData directory may be overlooked due to being a hidden directory by default.
- Malicious files are deleted from the initial download location.
- Persistence of malicious DLL (Goopdate.dll) on the system.
- Modification of legitimate executable (GoogleUpdate.exe) to load malicious DLL.
- Trojanisation of legitimate FireFox NSIS installation package.
- Execution of malicious code through rundll32.exe and url.dll.
- Creation of legitimate-looking directory for file storage.
- Hollowing of dllhost.exe binary for process injection.
- Creation of a Windows service for persistence.
- YARA rule (GoofyGuineapig_decodestring) installed on system.
- Network connections to C2 server.
- Presence of UPX packed NSIS installer.
- Modification of system binary (url.dll) for proxy execution.
- Deletion of files from initial download location.
- Network connections to the C2 server.
- Deletion of files from the initial download location.

- Logs of legitimate executable installation by the Goofy Guineapig loader.
- Logs of thread context change and thread resumption.
- Creation of a Windows service for persistence.
- Creation of a new dllhost.exe process in the suspended state.
- Injection of content downloaded by the C2.
- Execution of legitimate binary to load malicious DLL.
- Network connections to the FireFox NSIS installation package.
- Impersonation of the user associated with the session ID.
- Logs of CPU timestamp counter manipulation.
- Persistence of malicious DLL in the system.
- Execution of payload data through the new dllhost.exe process.
- Masquerading as legitimate processes through trojanized FireFox NSIS installation package.
- Hijack of execution flow through DLL side-loading.
- Logs of rundll32.exe and url.dll execution.
- Sideloaded by legitimate, signed, executable GoogleUpdate.exe.
- Network connections to the GoogleUpdate.exe server.
- Time-based evasion through CPU timestamp counter manipulation.
- Logs of GoogleUpdate.exe execution.
- Process hollowing on the dllhost.exe binary.
- Utilization of rundll32.exe and url.dll for persistence.
- Files created in the legitimate looking directory.
- Poor coding/testing practices and Op Sec.
- Presence of multiple mistakes throughout the binary.
- Network connections to the C2 server.
- Malicious DLL sideloaded by the legitimate GoogleUpdate.exe executable.
- Logs of physical memory size exceeding 2GB.
- Rundll32.exe and url.dll used to execute the legitimate binary.
- Malicious DLL dropped alongside legitimate FireFox files.
- Adapter information and host name concatenated and an MD5 hash taken.
- Logs of plugin loading and unloading.
- Files dropped alongside legitimate FireFox files.
- Files copied to the ProgramData directory.
- Concatenated adapter information and host name logs.
- Sideloaded malicious DLL logs.
- Rundll32.exe and url.dll execution logs.
- Disk size exceeds 1GB.
- Anti-sandbox/anti-VM technique logs.
- Persistence mechanism installed on the system.
- Malicious DLL loaded by a legitimate signed executable.
- Process hollowing performed on the dllhost.exe binary.
- Logs of request IDs 0x19 and 0x1A execution.
- Physical memory size of the machine exceeds 2GB.
- Malicious files copied to the ProgramData directory.
- Logs of plugin_run function execution.
- Basic anti-sandbox/anti-VM techniques implemented.
- MD5 hash logs.
- Content injected into the dllhost.exe process.

Attack Step 2.1

==
Name: Masquerading: Match Legitimate Name or Location as used by the malware

Description: The malware masquerades as legitimate software by using the following tactics: File Names: It disguises itself as a Firefox installer by using the filename "Firefox-latest.exe" and a trojanized NSIS FireFox installer named "setup-stub.exe". Functionality: It further mimics legitimate behavior by including a malicious DLL named "Goopdate.dll" that pretends to be a Google update. This deceptive naming and functionality aim to trick users into believing the malware is harmless software, increasing the likelihood of successful execution. Let me know if you'd like more details on any specific aspect of the malware's behavior.

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

- The file 'Firefox-latest.exe' was observed.
- The file 'setup-stub.exe' was observed.
- The file 'Goopdate.dll' was observed.
- The file 'config.dat' was observed.

Attack Step 2.2

==
Name: Virtualization/Sandbox Evasion: Time Based Evasion as used by the malware
Description: The provided YARA rule and context describe a malware called "Goofy Guineapig" and its evasion techniques. Here's a breakdown of how the Virtualization/Sandbox Evasion: Time Based Evasion technique works, based on the information given: Time Check: Goofy Guineapig checks the system's time register at two points. Delay: The malware intentionally introduces a delay of more than 100 milliseconds between these two time checks. Execution Halt: If the elapsed time between the checks exceeds the predefined threshold (more than 100 milliseconds), the malware will halt its execution. Why this is an evasion technique: Sandboxing: Virtualized environments and sandboxes often manipulate system time to speed up analysis. By checking the time difference, Goofy Guineapig can detect this manipulation and avoid executing in a potentially controlled environment. Delaying Execution: The delay itself can be a tactic to make analysis more difficult. Analysts might be observing the malware's behavior in real-time, and a significant delay could make it harder to track its actions. Important Note: The provided context doesn't explicitly state the exact code or mechanism used for the time check. It only describes the general behavior. Let me know if you have any other questions about malware analysis or YARA rules!

MITRE Technique

ID: T1497.003
Name: Virtualization/sandbox evasion: time based evasion
Description: Adversaries may employ various time-based methods to detect and avoid virtualization and analysis environments. This may include enumerating time-based properties, such as uptime or the system clock, as well as the use of timers or other triggers to avoid a virtual machine environment (VME) or sandbox, specifically those that are automated or only operate for a limited amount of time.

More info: https://attack.mitre.org/techniques/T1497/003

Indicators

No indicators found.

Attack Step 2.3

==
Name: Virtualization/Sandbox Evasion: System Checks as used by the malware
Description: The malware performs Virtualization/Sandbox Evasion: System Checks by: Checking the disk size: It queries the system's total disk space. Checking the physical memory size: It determines the amount of physical RAM available. Checking the number of logical processors: It counts the number of CPU cores. If any of these checks reveal values that are atypical of a real system (often smaller or artificially limited in sandboxes), the malware will abort its execution, preventing further analysis. This technique aims to detect if the malware is running in a controlled environment like a sandbox, where its behavior might be monitored or restricted.

MITRE Technique

ID: T1497.001
Name: Virtualization/sandbox evasion: system checks
Description: Adversaries may employ various system checks to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads. Adversaries may use the information learned from Virtualization/Sandbox Evasion during automated discovery to shape follow-on behaviors.
More info: https://attack.mitre.org/techniques/T1497/001

Indicators

No indicators found.

Attack Step 2.4

==
Name: Virtualization/Sandbox Evasion: User Activity Based Checks as used by the malware
Description: The malware Goofy Guineapig performs Virtualization/Sandbox Evasion: User Activity Based Checks by monitoring running processes on the infected system. Here's how it works: Process Monitoring: Goofy Guineapig actively scans the list of currently running processes. Keyword Detection: It specifically looks for process names or names containing keywords like "dbg", "debug", or "ida". These keywords are often associated with debugging tools and reverse engineering environments. Execution Termination: If Goofy Guineapig detects any process matching these keywords, it interprets this as a sign that it is being analyzed in a sandbox or reverse engineering environment. As a result, it terminates its own execution to avoid further scrutiny. This technique aims to prevent analysts from effectively examining the malware's behavior by halting its operation when it senses it's under observation.

MITRE Technique

ID: T1497.002
Name: Virtualization/sandbox evasion: user activity based checks
Description: Adversaries may employ various user activity checks to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads. Adversaries may use the information learned from Virtualization/Sandbox Evasion during automated discovery to shape follow-on behaviors.
More info: https://attack.mitre.org/techniques/T1497/002

Indicators

- The malware modifies the 'User-Agent' header in HTTP requests.

Attack Step 2.5

==
Name: Obfuscated Files or Information: Software Packing as used by the malware
Description: The malware, Goofy Guineapig, obfuscates its code by using UPX packing. This means the actual malicious code is compressed and encrypted within the executable file. When the file is run, UPX unpacks and decrypts the code, allowing it to execute. Furthermore, Goofy Guineapig is distributed using a legitimate NSIS installer. This installer is used to package and deploy the UPX-packed malware, making it appear more legitimate and potentially bypassing some security measures. Let me know if you have any other questions about malware analysis or Goofy Guineapig!

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

- The file 'Firefox-latest.exe' has a MD5 hash of 'a21dec89611368313e138480b3c94835'.
- The file 'Firefox-latest.exe' has a SHA-1 hash of '2b8aaab068ef15cb05789d a320b7099932a0a4166'.
- The file 'Firefox-latest.exe' has a SHA-256 hash of '19cef7f32e42cc674f7c76be3a5c691c543f4e018486c29153e7dde1a48af34c'.
- The file 'setup-stub.exe' has a MD5 hash of '180e0bb4b570c215bfe7abdf209402aa'.
- The file 'setup-stub.exe' has a SHA-1 hash of '6f5c07c50ce4976ddbb3879ce65d3b2f96693dc4c'.
- The file 'setup-stub.exe' has a SHA-256 hash of '97f66bcd d73917a8b59d9a1dcac21a58936bca f91e757a9dfb8e5c320af40f56'.
- The file 'Goopdate.dll' has a MD5 hash of 'f98537517212068d0c57968876fc8204'.
- The file 'Goopdate.dll' has a SHA-1 hash of '7961930d13cb8d5056db64b6749356915fb4c272'.
- The file 'Goopdate.dll' has a SHA-256 hash of '12a29373c1f493f7757b755099bd e4770c310af3fde376176b6d792cd1c5e150'.
- The file 'config.dat' has a MD5 hash of '3dc1096e73db4886fb66ed9413ca994c'.
- The file 'config.dat' has a SHA-1 hash of '628ce6721b97fa12590356712fbffc5ae030781ce'.

- The file 'config.dat' has a SHA-256 hash of '3a1af09a0250c602569d458e79db90a45e305b76d8423b81ee ca14c69847b81c'.

Attack Step 2.6

==
Name: Deobfuscate/Decode Files or Information as used by the malware
Description: The malware Goofy Guineapig obfuscates its stack-based strings using single-byte XOR or subtraction operations applied throughout the binary. Here's a breakdown of how this deobfuscation/decoding action is performed: Stack-Based Strings: Goofy Guineapig stores its strings in memory on the program's stack. This is a common technique to hide strings from static analysis, as they are not directly embedded in the executable's code. XOR or Subtraction: The strings are encoded using either XOR (exclusive OR) or subtraction operations with a specific key. This means each character in the original string is transformed into a different character based on the key. Decoding at Runtime: When the malware needs to use a string, it performs the reverse XOR or subtraction operation on the encoded characters, effectively "decoding" the string back to its original form. This decoding happens dynamically at runtime, making it harder to detect the strings during static analysis. Why this is effective: Obfuscation: XOR and subtraction are simple yet effective ciphers that can make strings difficult to identify without knowing the key. Runtime Execution: Decoding at runtime means the strings are not readily visible in the malware's code, making static analysis more challenging. How to detect this: Dynamic Analysis: Observing the malware's behavior during runtime can reveal the decoding process. String Analysis Tools: Specialized tools can analyze the binary and identify potential encoded strings, even if they are not immediately visible. Signature-Based Detection: Creating signatures based on the XOR or subtraction patterns used for encoding can help detect the malware. Let me know if you have any other questions.

MITRE Technique

ID: T1140
Name: Deobfuscate/decode files or information
Description: Adversaries may use Obfuscated Files or Information to hide artifacts of an intrusion from analysis. They may require separate mechanisms to decode or deobfuscate that information depending on how they intend to use it. Methods for doing that include built-in functionality of malware or by using utilities present on the system.
More info: https://attack.mitre.org/techniques/T1140/

Indicators

- The file 'Firefox-latest.exe' has a MD5 hash of 'a21dec89611368313e138480b3c94835'.
- The file 'Firefox-latest.exe' has a SHA-1 hash of '2b8aab068ef15cb05789da320b7099932a0a4166'.
- The file 'Firefox-latest.exe' has a SHA-256 hash of '19cef7f32e42cc674f7c76be3a5c691c543f4e018486c29153e7dde1a48af34c'.
- The file 'setup-stub.exe' has a MD5 hash of '180e0bb4b570c215bfe7abdf209402aa'.
- The file 'setup-stub.exe' has a SHA-1 hash of '6f5c07c50ce4976ddbb3879ce65d3b2f96693dc4c'.
- The file 'setup-stub.exe' has a SHA-256 hash of '97f66bcd d73917a8b59d9a1dcac21a58936bca f91e757a9dfb8e5c320af40f56'.
- The file 'Goopdate.dll' has a MD5 hash of 'f98537517212068d0c57968876fc8204'.
- The file 'Goopdate.dll' has a SHA-1 hash of '7961930d13cb8d5056db64b6749356915fb4c272'.
- The file 'Goopdate.dll' has a SHA-256 hash of '12a29373c1f493f7757b755099bd e4770c310af3fde376176b6d792cd1c5e150'.
- The file 'config.dat' has a MD5 hash of '3dc1096e73db4886fb66ed9413ca994c'.

- The file 'config.dat' has a SHA-1 hash of '628ce6721b97fa12590356712fbffc5ae030781ce'.
- The file 'config.dat' has a SHA-256 hash of '3a1af09a0250c602569d458e79db90a45e305b76d8423b81eeeca14c69847b81c'.

Attack Step 2.7

==
Name: Hide Artifacts: Hidden Window as used by the malware
Description: The malware Goofy Guineapig hides its artifacts by using process hollowing on the dllhost.exe process. Here's how it works: Process Hollowing: Goofy Guineapig injects its malicious code into a legitimate process, dllhost.exe, effectively replacing the original code with its own. Hidden Creation: During this process hollowing, the malware ensures that the newly hollowed dllhost.exe process is created hidden. This prevents the process from appearing in standard task lists or system monitoring tools. Obfuscated Activity: By running its malicious code within the context of a legitimate process like dllhost.exe, Goofy Guineapig attempts to blend in and avoid detection. The hidden nature of the process further enhances its ability to evade scrutiny. Let me know if you have any other questions about Goofy Guineapig or malware techniques.

MITRE Technique

ID: T1564
Name: Hide artifacts
Description: Adversaries may attempt to hide artifacts associated with their behaviors to evade detection. Operating systems may have features to hide various artifacts, such as important system files and administrative task execution, to avoid disrupting user work environments and prevent users from changing files or features on the system. Adversaries may abuse these features to hide artifacts such as files, directories, user accounts, or other system activity to evade detection.
More info: https://attack.mitre.org/techniques/T1564/

Indicators

No indicators found.

Attack Step 2.8

==
Name: Indicator Removal on Host: File Deletion as used by the malware
Description: The malware performs Indicator Removal on Host: File Deletion by following these steps: Initial Execution: The malware first runs from the directory where it was downloaded. File Relocation: It then copies the malicious files to a directory that appears legitimate. This could be a system folder or a folder commonly used by legitimate applications. Deletion: Finally, the malware deletes the original files from the initial download location. This action aims to remove traces of its presence and make it harder to detect. By relocating and deleting the files, the malware attempts to evade detection by security tools that might be looking for malicious files in the initial download location.

MITRE Technique

ID: T1070.010
Name: Indicator removal: relocate malware
Description: Once a payload is delivered, adversaries may reproduce copies of the same malware on the victim system to remove evidence of their presence and/or avoid defenses. Copying malware payloads to new locations may also be combined with File Deletion to cleanup older artifacts.

More info: https://attack.mitre.org/techniques/T1070/010

Indicators

- Goofy Guineapig sample (MD5: a21dec89611368313e138480b3c94835 Filename: 'Firefox-latest.exe ')
- Goofy Guineapig sample (MD5: 180e0bb4b570c215bfe7abdf209402aa Filename: 'setup-stub.exe')
- Goofy Guineapig sample (MD5: f98537517212068d0c57968876fc8204 Filename: 'Goopdate.dll')
- Goofy Guineapig sample (MD5: 06e1992e6c52af33117d142bdbeef74d)
- Goofy Guineapig sample (MD5: abbe7d13b13ea4315543bdad187f14b3)
- Goofy Guineapig sample (MD5: 3dc1096e73db4886fb66ed9413ca994c Filename: 'config.dat')

Attack Step 2.9

==
Name: Hijack Execution Flow: DLL Side-Loading as used by the malware
Description: Here's a breakdown of how "Hijack Execution Flow: DLL Side-Loading" is performed by the Goofy Guineapig malware, based on the provided context: 1. The Setup: Legitimate Executable: Goofy Guineapig's loader installs a legitimate, signed executable file (in this case, GoogleUpdate.exe) onto the victim's system. This executable appears normal and is trusted by the operating system. Malicious DLL: Simultaneously, the loader also installs a malicious DLL (the "Goofy Guineapig" DLL) alongside the legitimate executable. 2. The Hijack: Loading the DLL: When the legitimate executable (GoogleUpdate.exe) is run, the operating system loads the associated DLLs. Because the malicious DLL is in the same directory, it gets loaded alongside the legitimate one. Execution: Once loaded, the malicious DLL can execute its code within the context of the legitimate process. This means it can leverage the legitimate executable's privileges and avoid detection by security software that might focus on suspicious standalone processes. Why This is Effective: Trust: The legitimate executable acts as a shield. Since the executable is trusted, the operating system doesn't scrutinize the loaded DLL as closely. Persistence: The malicious DLL can now run persistently as part of the legitimate process, making it harder to remove. Evasion: The malware can use the legitimate process's resources and network connections to communicate with its command-and-control (C2) server without raising red flags. Key Points: Side-loading is a technique where malicious code is injected into a legitimate process by exploiting the way DLLs are loaded. DLLs (Dynamic Link Libraries) are shared code modules that programs can use. They are often located in system directories or alongside executable files. Let me know if you have any other questions.

MITRE Technique

ID: T1574.001
Name: Hijack execution flow: dll search order hijacking
Description: Adversaries may execute their own malicious payloads by hijacking the search order used to load DLLs. Windows systems use a common method to look for required DLLs to load into a program. Hijacking DLL loads may be for the purpose of establishing persistence as well as elevating privileges and/or evading restrictions on file execution.
More info: https://attack.mitre.org/techniques/T1574/001

Indicators

- The file 'Goopdate.dll' was observed.

Attack Step 2.10

==
Name: Process Injection: Process Hollowing as used by the malware
Description: The malware, Goofy Guineapig, performs process injection via process hollowing on the dllhost.exe binary. Here's a breakdown of how it's done: Target Selection: Goofy Guineapig targets the dllhost.exe process, a legitimate Windows process responsible for hosting dynamic link libraries (DLLs). Process Hollowing: Memory Allocation: The malware allocates a new memory region within the target process's address space, large enough to hold the malicious code it intends to inject. Code Copying: It then copies the malicious code downloaded from the Command and Control (C2) server into this allocated memory region. Overwriting Original Code: Goofy Guineapig overwrites the original code of dllhost.exe with a "hollowed" shell, essentially replacing the legitimate functionality with its own malicious payload. Execution: Finally, the malware redirects the execution flow of dllhost.exe to the beginning of the injected malicious code, effectively taking control of the process. Persistence: By injecting its code into a legitimate process like dllhost.exe, Goofy Guineapig achieves persistence, as the injected code runs alongside the legitimate process and survives system restarts. This technique allows the malware to hide its malicious nature by operating within a trusted process, making it harder to detect and analyze.

MITRE Technique

ID: T1055.012
Name: Process injection: process hollowing
Description: Adversaries may inject malicious code into suspended and hollowed processes in order to evade process-based defenses. Process hollowing is a method of executing arbitrary code in the address space of a separate live process.
More info: https://attack.mitre.org/techniques/T1055/012

Indicators

- The file 'tmp.bat' is located in the directory 'C:\ProgramData\GoogleUpdate\GoogleUpdate'.

Attack Step 2.11

==
Name: Signed Binary Proxy Execution: Rundll32 as used by the malware
Description: The malware utilizes the rundll32.exe utility and the url.dll library to achieve persistence. Here's a breakdown of the process: Leveraging rundll32.exe: rundll32.exe is a legitimate Windows tool used to execute functions from DLL files. The malware exploits this by specifying a path to url.dll and a function within it. url.dll Function: The function within url.dll is designed to handle URL protocol execution. The malware manipulates this function to load and execute its malicious DLL. Loading the Malicious DLL: When rundll32.exe executes the specified function in url.dll, it effectively loads the malware's DLL into memory. This allows the malware to run alongside legitimate processes and maintain persistence. In essence, the malware disguises its malicious DLL as a legitimate URL protocol handler, tricking rundll32.exe into loading and executing it. Let me know if you'd like more details on any specific aspect of this process!

MITRE Technique

ID: T1218.010
Name: System binary proxy execution: regsvr32
Description: Adversaries may abuse Regsvr32.exe to proxy execution of malicious code. Regsvr32.exe is a command-line program used to register and unregister object linking and embedding controls, including dynamic link libraries (DLLs), on Windows systems. The Regsvr32.exe binary may also be signed by Microsoft.

More info: https://attack.mitre.org/techniques/T1218/010

Indicators

- Path: C:\ProgramData\GoogleUpdate\GoogleUpdate\tmp.bat

Milestone 3

Pre-Conditions:

- The malware has access to the WMI information.
- Availability of the malware on the victim machine.
- The malware is loaded on the victim machine.
- Tools: COM interface, Windows APIs, WMI information.
- The victim machine has a physical memory size exceeding 2GB.
- The malware has access to the Windows APIs.
- Resources: Access to the victim machine's physical memory and disk.
- Environment: Windows operating system.
- Connectivity: None.

Post-Conditions:

- Deletion of files from the initial download location.
- Implementation of anti-sandbox and anti-VM techniques.
- Windows service logs.
- Transmission of system information via HTTPS.
- Creation of a legitimate-looking directory to store files.
- Modification of system properties to evade detection.
- YARA rule files.
- Network connections to C2 servers via HTTPS on port 4443.
- Collection of system information about the infected machine.
- System property modifications.
- Creation of a YARA rule for detection.
- Files in the legitimate-looking directory.
- Log entries of system information collection.
- Persistence of a malicious Windows service.

Attack Step 3.1

==
Name: System Information Discovery as used by the malware
Description: Let's break down how the "Goofy Guineapig" malware performs System Information Discovery and the techniques it uses. System Information Discovery Goofy Guineapig is designed to gather information about the infected machine and send it back to its command-and-control (C2) server. Here's how it does this: Obscured Data: Instead of sending raw system data, Goofy Guineapig encodes the information within an "Authorization" string in the HTTP header of its communications with the C2 server. This obfuscation attempts to make the data harder to detect by security tools. Data Types: The specific system information Goofy Guineapig collects likely includes: Machine Name: The name of the infected computer. Operating System: Details about the version and edition of Windows running on the machine. User Information: Potentially the logged-in username, although this might be more limited due to security restrictions. Network Configuration: Information about the network interfaces, IP addresses, and possibly even domain membership. Other System Details: Goofy Guineapig might also gather details about installed software, hardware specifications, or other system-level data. Transmission: The encoded system information is sent to the C2 server over HTTPS, using a non-standard port (4443) to further obscure its activity. Defense Evasion Techniques

The report highlights several defense evasion techniques employed by Goofy Guineapig: Process Hollowing: Goofy Guineapig can use process hollowing to execute plugins within the dllhost.exe process. This makes it harder to track the malware's activity as it hides within a legitimate process. Virtualization/Sandbox Evasion: The malware checks for signs that it's running in a virtualized environment or sandbox, and will terminate itself if detected. User Activity Checks: Goofy Guineapig monitors system activity for indicators of reverse engineering or debugging, such as the presence of debuggers or unusual process behavior. Importance of Detection Understanding how Goofy Guineapig performs system information discovery is crucial for detection and mitigation. Security tools should: Monitor Network Traffic: Look for suspicious HTTPS connections to non-standard ports, especially those containing obfuscated data in HTTP headers. Analyze Process Behavior: Identify suspicious processes, particularly those using process hollowing techniques or exhibiting unusual activity patterns. Detect Defense Evasion Techniques: Implement security measures that can detect and prevent virtualization/sandbox evasion and user activity checks. Let me know if you have any more questions about Goofy Guineapig or malware analysis in general!

MITRE Technique

ID: T1049
Name: System network connections discovery
Description: Adversaries may attempt to get a listing of network connections to or from the compromised system they are currently accessing or from remote systems by querying for information over the network.
More info: https://attack.mitre.org/techniques/T1049/

Indicators

- User Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36 was observed.

Milestone 4

Pre-Conditions:

- The victim machine has a network connection.
- Resources: A victim machine with a valid operating system and sufficient resources to run the malware.
- The malware has the necessary encryption keys to use RC4 encryption.
- The malware has access to the victim machine's network settings.
- The malware has the necessary configuration to use HTTPS.
- Environment: A Windows environment (inferred from the context)
- Tools: None directly stated, but likely requires a network analysis tool or a sandbox environment to analyze the malware's behavior.
- The malware has been successfully executed on the victim machine.
- Connectivity: A stable internet connection (inferred from the context)
- The malware has the necessary permissions to communicate over the web.
- A valid URL or domain name to communicate with (inferred from the context)
- The embedded configuration string contains the string 'KCP protocol'.
- The system must not have a debugger or analysis tools running.
- Connectivity: A network connection is required for the malware to communicate over UDP.
- Tools: None explicitly stated, but the malware is likely to be running on the system.
- Environment: A Windows environment.
- The embedded configuration string contains the string 'direct socket communications'.
- The embedded configuration string contains UDP rather than HTTP (S).
- The system must not have a process named 'dbg', 'debug', or 'ida' running.
- The embedded configuration string must be present in the malware's configuration.
- Resources: The system must have a physical memory size exceeding 2GB and a disk size exceeding 1GB.
- The malware is loaded by a legitimate signed executable.
- Connectivity: None explicitly stated, but the malware communicates over HTTPS, which implies that internet connectivity is required.
- The malware has the necessary permissions to collect information about the system.
- Tools: None explicitly stated, but the malware uses Windows APIs, which implies that the Windows operating system is the tool.
- The malware has access to the Windows APIs.
- The malware maintains persistence using a Windows service.
- Resources: None explicitly stated, but the malware uses system resources such as memory and disk space.

Post-Conditions:

- Network connections to static.tcplog.com
- C2 communications over HTTPS using GET and POST requests to static.tcplog.com
- RC4-encrypted payloads sent in HTTP POST body
- Malware executable files (UPX packed)
- Malware checks for running processes containing 'dbg', 'debug', or 'ida'
- Malware infection on the machine
- Malware sends system information in C2 packets as an obfuscated 'Authorization' string
- Malware configuration files (e.g. YARA rules)
- Malware uses flawed logic in secondary process check

- Malware uses UPX packed loader
- Malware checks for debugger and sandbox/VM presence
- Network connections to UDP ports.
- Malicious direct socket communications.
- Windows service logs indicating persistence.
- Malicious process checks for debugger and analysis tools.
- Persistence of a malicious Windows service.
- Malicious MD5 hash calculations.
- Malicious DLL loaded by a legitimate signed executable.
- Malicious UDP traffic using KCP protocol.
- Malicious configuration string hardcoded in the binary.
- Files created by the malicious DLL.
- Network connections to static.tcplog.com:4443.
- Malicious computer information sent in HTTP headers.
- Malicious process checks for running processes containing 'dbg', 'debug', or 'ida'.
- MD5 hash calculations logs.
- Process logs indicating checks for debugger and analysis tools.
- Malicious HTTP(S) traffic over non-standard port 4443.
- Windows service logs for persistence.
- Network connections to UDP and KCP protocol or direct socket communications.
- Malicious communication over non-standard HTTPS port 4443.
- Malicious communication using UDP and KCP protocol or direct socket communications.
- Logs of sandbox detection evasion techniques.
- Creation of an MD5 hash of the concatenated adapter information and host name.
- Logs of fallback channels communication.
- Persistence of a malicious DLL (Goofy Guineapig) in the system.
- Collection of host information and adapter information.
- Creation of a Windows service for persistence.
- Malicious code execution through a UPX packed setup-stub.exe.
- Malicious code execution through a Trojanised NSIS FireFox installer.
- Malicious code execution through a legitimate signed executable.
- Network connections to static[.]tcplog[.]com.
- Malicious communication using fallback channels.
- Files created by the malicious code execution (e.g. Config.dat, Goopdate.dll, etc.).
- Files created by the UPX packed setup-stub.exe.
- Files created by the malicious DLL (Goofy Guineapig).

Attack Step 4.1

==
Name: Application Layer Protocol: Web Protocols as used by the malware
Description: The malware, Goofy Guineapig, utilizes HTTPS (Hypertext Transfer Protocol Secure) for its communication with its Command and Control (C2) server. This means that the malware establishes encrypted connections over port 443, the standard port for HTTPS, to send and receive data with its C2 server. This encryption helps the malware to conceal its activities from network monitoring and analysis. Let me know if you have any other questions about this malware or its tactics.

MITRE Technique

ID: T1071
Name: Application layer protocol

Description: Adversaries may communicate using OSI application layer protocols 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/

Indicators

- The URL HTTPS://static.tcplog.com was observed.
- The User Agent Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36 was detected.

Attack Step 4.2

==
Name: Fallback Channels as used by the malware
Description: The malware Goofy Guineapig utilizes fallback channels for communication when its primary method is unavailable. Here's how it's performed: UDP: Goofy Guineapig can communicate using the User Datagram Protocol (UDP). This is a connectionless protocol, meaning it doesn't establish a persistent connection before sending data. KCP: The malware also supports the KCP (Kylix Crypto Protocol) for communication. KCP is a fast and reliable UDP-based protocol designed for low-latency and high-throughput applications. Direct Socket Communications: Goofy Guineapig can establish direct socket communications. This allows for more control over the communication process, but it might be less efficient than UDP or KCP. The specific communication method used depends on an embedded configuration string within the malware. This string likely dictates which fallback channel to prioritize or use in specific situations. Let me know if you have any other questions about Goofy Guineapig or malware communication techniques!

MITRE Technique

ID: T1573
Name: Encrypted channel
Description: Adversaries may employ an encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Despite the use of a secure algorithm, these implementations may be vulnerable to reverse engineering if secret keys are encoded and/or generated within malware samples/configuration files.
More info: https://attack.mitre.org/techniques/T1573/

Indicators

- URL: HTTPS://static.tcplog.com is observed.
- Path: C:\ProgramData\GoogleUpdate\GoogleUpdate\tmp.bat is located.

Attack Step 4.3

==
Name: Non-Standard Port as used by the malware
Description: The malware, Goofy Guineapig, communicates with its command and control (C2) server using HTTPS over the non-standard port 4443. This means it deviates from the typical port 443 used for standard HTTPS traffic, making it harder to detect through traditional security measures that might focus on known ports. Let me know if you have any other questions about this malware or its tactics.

MITRE Technique

ID: T1090.001
Name: Proxy: internal proxy
Description: Adversaries may use an internal proxy to direct command and control traffic between two or more systems in a compromised environment. Many tools exist that enable traffic redirection through proxies or port redirection, including HTRAN, ZXProxy, and ZXPortMap. Adversaries use internal proxies to manage command and control communications inside a compromised environment, to reduce the number of simultaneous outbound network connections, to provide resiliency in the face of connection loss, or to ride over existing trusted communications paths between infected systems to avoid suspicion. Internal proxy connections may use common peer-to-peer (p2p) networking protocols, such as SMB, to better blend in with the environment.
More info: https://attack.mitre.org/techniques/T1090/001

Indicators

No indicators found.