Enhanced Attack Report

Goofy Guineapig

Generated on 2025-03-04

Disclaimer

This report has been generated automatically and should be used for informational purposes only. To generate this report, Large Language Models (LLMs) were used to analyze the provided data. This technology is not perfect and may generate incorrect or misleading results. The results should be reviewed by a human expert before taking any action based on the information provided.

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.

Milestone 1

Pre-Conditions:

- The Windows operating system is installed.
- The necessary permissions to create or modify a Windows service are available.
- The malware Goofy Guineapig is present on the system.
- The system has a network connection.
- The system has a valid Windows service configuration.
- The system has a valid Windows service name.
- The system has a valid Windows service description.
- The system has a valid Windows service binary.
- Windows operating system.
- A valid Windows service configuration.
- A valid Windows service name.
- A valid Windows service description.
- A valid Windows service binary.
- Network connectivity.
- The necessary permissions to create or modify a Windows service.
- The malware Goofy Guineapig.
- A system with a valid Windows service installation process.
- A system with a valid Windows service management process.
- A system with a valid Windows service registry.
- A system with a valid Windows service execution environment.

Post-Conditions:

- Persistence mechanism installed as a Windows service.
- Malicious files copied to the ProgramData directory.
- Malicious files removed from the directory containing the extracted Firefox files.
- Malicious DLL Goopdate.dll loaded by the legitimate signed executable file GoogleUpdate.exe.
- Malicious files bundled in a UPX packed NSIS installer.
- Malware maintains persistence without a registered bootstrap.
- Malware continues to execute without a persistence mechanism.
- Malware collects information about the infected machine.
- Malware runs additional plugins either as part of the current process or by process hollowing dllhost.exe.
- Malware checks the name of each running process on the machine.
- Malware determines if any process containing the string 'dbg', 'debug', or 'ida' is running.
- Malware will not continue execution if a process containing the string 'dbg', 'debug', or 'ida' is running.
- Windows service created in the Services.msc console.
- Malicious files in the ProgramData directory.
- Removed files in the directory containing the extracted Firefox files.
- Loaded DLL Goopdate.dll in the Windows Task Manager.
- UPX packed NSIS installer in the Windows Temp directory.
- Logs of the malware's activities in the Windows Event Viewer.
- Network connections to the C2 server.
- Files created by the malware in the %TEMP% directory.
- Registry keys created by the malware in the Windows Registry.

- Process hollowing of dllhost.exe in the Windows Task Manager.
- Malware's plugin execution in the Windows Task Manager.
- Logs of the malware's plugin execution in the Windows Event Viewer.
- MD5 hash of the concatenated adapter information and host name in the malware's configuration file.
- Malware's configuration file in the %APPDATA% directory.

Attack Step 1.1

==========
Name: Create or Modify System Process: Windows Service as used by Goofy Guineapig
Description: Goofy Guineapig maintains persistence as a Windows service by creating or modifying an existing Windows service. The exact method of creation or modification isn't specified in the provided text. However, it's common for malware to use existing legitimate service frameworks or create new ones to achieve persistent execution. Let me know if you have any other questions.

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 Goofy Guineapig malware is deployed.
- The malware has access to the system.
- The system has a legitimate Firefox installation.
- The system has a legitimate GoogleUpdate executable.
- The malware has the capability to modify system files.
- The malware has the capability to create or modify system processes.
- The system has a Windows operating system.
- The system has a UPX packed NSIS installer.
- The system has a trojanised Firefox installer.
- The system has a legitimate FireFox NSIS installation package.
- Environment: Windows operating system.
- Tools: UPX packed NSIS installer, trojanised Firefox installer, legitimate FireFox NSIS installation package, GoogleUpdate executable.
- Connectivity: None.
- Resources: System files, system processes, legitimate Firefox installation.
- Availability of the Goofy Guineapig malware.
- Availability of the legitimate GoogleUpdate executable.
- Availability of the legitimate FireFox NSIS installation package.
- Availability of the UPX packed NSIS installer.
- Availability of the trojanised Firefox installer.
- Availability of the system's ProgramData directory.
- The system time is registered twice.
- The time delay between the two registrations is more than 100 milliseconds.
- The system has a physical memory size exceeding 2GB.
- The system has a disk size exceeding 1GB.
- The system has a number of logical processors exceeding 2.
- A system with a physical memory size exceeding 2GB.
- A system with a disk size exceeding 1GB.
- A system with a number of logical processors exceeding 2.
- A system with a time registration mechanism.
- A system with a delay mechanism.
- A system with a debugger or analysis tools (to be detected by the evasion mechanism).
- A system with a reverse engineering or debugging process (to be detected by the evasion mechanism).
- A system with a network connection (to perform the evasion mechanism).
- A system with a HTTPS connection (to perform the evasion mechanism).
- A system with a GET request capability (to perform the evasion mechanism).
- A system with a POST request capability (to perform the evasion mechanism).
- A system with a batch script execution capability (to perform the evasion mechanism).
- A system with a file deletion capability (to perform the evasion mechanism).
- A system with a process restart capability (to perform the evasion mechanism).
- A system with a sandbox or virtual machine detection capability (to be detected by the evasion mechanism).
- The system has a disk.
- The system has physical memory.
- The system has logical processors.

- The system's disk size is greater than 1GB.
- The system's physical memory size is greater than 2GB.
- The system's number of logical processors exceeds 2.
- Environment: A system with a disk, physical memory, and logical processors.
- Tools: None explicitly stated, but the system's disk, physical memory, and logical processors are required.
- Connectivity: None explicitly stated.
- Resources:
- The system must be running a version of Goofy Guineapig that implements Virtualization/Sandbox Evasion: System Checks.
- The system has processes running.
- The system has system properties.
- The system has system time.
- The system has running processes.
- The system has an automated analysis environment.
- A system with processes running.
- A system with system properties.
- A system with system time.
- A system with running processes.
- A system with an automated analysis environment.
- A system with a physical memory size exceeding 2GB.
- A system with a disk size exceeding 1GB.
- A system with a number of logical processors exceeding 2.
- A system with a debugger.
- A system with other analysis tools.
- A system with a network connection for potential sandbox detection checks.
- A system with a UPX packed NSIS installer.
- A system with a Firefox installer.
- A system with a GoogleUpdate.exe executable file.
- A system with a Goopdate.dll DLL file.
- The software Goofy Guineapig is available.
- The software Goofy Guineapig is in a format that can be packed.
- The tool UPX is available.
- The tool NSIS is available.
- The legitimate signed executable file GoogleUpdate.exe is available.
- The trojanised Firefox installer is available.
- Environment: A Windows environment is required.
- Tools: UPX, NSIS, and a legitimate signed executable file (GoogleUpdate.exe) are required.
- Connectivity: No specific connectivity requirements are mentioned.
- Resources: A computer with sufficient resources to run the software and tools is required.
- Availability of the software Goofy Guineapig and the trojanised Firefox installer.
- The file or information to be deobfuscated is available.
- The deobfuscation key or method is known.
- The deobfuscation algorithm or technique is understood.
- The environment is stable and secure.
- The necessary tools and resources are available.
- The file or information is in a format that can be deobfuscated.
- The deobfuscation process is reversible.
- The original data is intact and not corrupted.
- The deobfuscation process is performed in a controlled environment.
- The necessary permissions and access rights are granted.
- Environment: A stable and secure environment with access to the necessary tools and resources.

- Tools: A deobfuscation tool or software that can handle single-byte XOR, subtraction, and other obfuscation techniques.
- Connectivity: Access to the internet for research and reference purposes.
- Resources: A computer with sufficient processing power, memory, and storage to handle the deobfuscation process.
- File or information: The Goofy Guineapig binary or other files that require deobfuscation.
- Deobfuscation key or method: Knowledge of the deobfuscation key or method used by Goofy Guineapig.
- Understanding of deobfuscation algorithms: Knowledge of the deobfuscation algorithms or techniques used by Goofy Guineapig.
- Necessary permissions: Access rights and permissions to perform the deobfuscation process.
- Reference materials: Access to the NCSC-MAR-Goofy-Guineapig.pdf report or other reference materials that provide information on Goofy Guineapig's deobfuscation techniques.
- Expertise: Knowledge and expertise in deobfuscation and reverse engineering.
- The Goofy Guineapig malware is present on the system.
- The malware has the necessary permissions to create a hidden window.
- The system has a legitimate dllhost.exe process.
- The malware has the capability to perform process hollowing.
- The system has a CPU timestamp counter.
- The malware has the necessary configuration to perform the action.
- The system has a global structure to store the handle of the spawned process.
- The malware has the necessary data to write into the process memory.
- The system has a legitimate dllhost.exe process that can be suspended and resumed.
- The malware has the capability to impersonate the user associated with the session ID.
- Environment: Windows operating system.
- Tools: None explicitly stated, but the malware itself is the tool.
- Connectivity: None explicitly stated, but the malware likely requires network connectivity to communicate with its command and control server.
- Resources:
- The system must have a legitimate dllhost.exe process.
- The system must have a global structure to store the handle of the spawned process.
- The system must have a legitimate dllhost.exe process that can be suspended and resumed.
- The malware must have the necessary permissions to create a hidden window and perform process hollowing.
- The malware Goofy Guineapig is initially downloaded to a location on the host.
- The malware Goofy Guineapig is able to move files to a legitimate looking directory.
- The malware Goofy Guineapig is able to delete files from the initial download location.
- The malware Goofy Guineapig is able to copy files to the ProgramData directory.
- The ProgramData directory is accessible by the malware Goofy Guineapig.
- The malware Goofy Guineapig is able to delete files from the directory containing the extracted Firefox files.
- The malware Goofy Guineapig is able to delete the batch script that deletes itself.
- The malware Goofy Guineapig is able to restart the GoogleUpdate.exe process from the ProgramData directory.
- A Windows environment with a ProgramData directory.
- The malware Goofy Guineapig.
- Connectivity to the host to download and execute the malware.
- A legitimate signed executable file GoogleUpdate.exe.
- A malicious DLL Goopdate.dll.
- A UPX packed NSIS installer.
- A trojanised Firefox installer.
- A batch script that deletes itself.
- The ability to restart the GoogleUpdate.exe process.

- The ability to delete files from the initial download location and the directory containing the extracted Firefox files.
- The ability to copy files to the ProgramData directory.
- The ability to delete files from the ProgramData directory.
- A legitimate executable is installed by the Goofy Guineapig loader.
- A malicious DLL is loaded by the legitimate executable.
- The legitimate executable is signed.
- The legitimate executable is installed alongside the malicious DLL.
- The Goofy Guineapig loader is present in the system.
- The malicious DLL is sideloaded by the legitimate executable.
- The legitimate executable is dropped as part of the NSIS installer.
- The NSIS installer is a trojanised Firefox installation package.
- Windows environment.
- Presence of a legitimate executable.
- Presence of a malicious DLL.
- Presence of the Goofy Guineapig loader.
- Presence of the NSIS installer.
- Connectivity to the internet (for downloading content).
- Availability of the rundll32.exe and url.dll binaries.
- Availability of the dllhost.exe binary.
- Availability of a Windows service to maintain persistence.
- Presence of a non-standard HTTPS port (4443) for communication.
- Availability of a legitimate Firefox installation package to be trojanised.
- Availability of a signed executable to be used for sideloading the malicious DLL.
- The Goofy Guineapig malware is present on the system.
- The dllhost.exe binary is present on the system.
- The C2 content is available for injection.
- The system has a physical memory size exceeding 2GB.
- The system has a disk size exceeding 1GB.
- The system is not running in a sandbox or virtual machine environment.
- Environment: Windows operating system.
- Tools: None explicitly stated, but likely requires a tool to inject the malicious DLL into the dllhost.exe process.
- Connectivity: None explicitly stated, but likely requires network connectivity to communicate with the C2 server.
- Resources: Available system resources, including memory and disk space.
- Availability of dllhost.exe binary.
- Availability of C2 content for injection.
- Availability of GoogleUpdate.exe executable to sideload the malicious DLL.
- A legitimate executable is installed by the Goofy Guineapig loader.
- A malicious DLL is loaded by the legitimate executable.
- The legitimate executable is signed.
- The Goofy Guineapig loader is present in the environment.
- The malicious DLL is present in the environment.
- The legitimate executable and the malicious DLL are bundled in a NSIS installer.
- The NSIS installer is a trojanised Firefox installation package.
- The Goofy Guineapig malware is present in the environment.
- The Goofy Guineapig malware has the option to perform process hollowing on the dllhost.exe process.
- The Goofy Guineapig malware has the option to communicate over the non-standard HTTPS port 4443.
- Environment: Windows operating system.
- Tools: NSIS installer, Rundll32.exe, url.dll, GoogleUpdate.exe, Goopdate.dll.

- Connectivity: None.
- Resources: None.
- Availability of the Goofy Guineapig malware.
- Availability of the legitimate executable and the malicious DLL.
- Availability of the dllhost.exe process.
- Availability of the HTTPS port 4443.
- Presence of the Goofy Guineapig loader.
- Presence of the Firefox installation package.

Post-Conditions:

- Persistence mechanism installed
- Malicious files copied to ProgramData directory
- Malicious files removed from extracted Firefox files directory
- Windows service created
- Malicious DLL loaded by legitimate executable
- Malicious DLL sideloaded by legitimate executable
- Legitimate Firefox installation files bundled with malware
- Malware deployed through social engineering
- Indicators of compromise left behind
- Malware C2 infrastructure established
- Logs of Windows service creation
- Files in ProgramData directory
- Removed files from extracted Firefox files directory
- Loaded DLL in memory
- Sideloaded DLL in memory
- Bundled Firefox installation files
- Social engineering tactics used
- Indicators of compromise (URLs, IP addresses, etc.)
- Malware C2 infrastructure logs
- Network connections to C2 infrastructure
- Files packed with UPX
- Obfuscated files or information
- Stack-based strings in malware binary
- Logs of legitimate executable execution
- Files created by legitimate executable
- Persistence mechanism installed
- Malicious files copied to ProgramData directory
- Malicious files removed from original directory
- Goopdate.dll loaded by GoogleUpdate.exe
- UPX packed NSIS installer executed
- Firefox installer trojanised
- Time delay mechanism implemented
- Physical memory size check implemented
- Disk size check implemented
- User activity based checks implemented
- Processes indicating reverse engineering or debugging detected
- Malware execution terminated
- Logs of GoogleUpdate.exe and Goopdate.dll execution
- Files copied to ProgramData directory
- Files removed from original directory
- UPX packed NSIS installer executable

- Firefox installer trojanised
- Time delay mechanism logs
- Physical memory size check logs
- Disk size check logs
- User activity based checks logs
- Process logs indicating reverse engineering or debugging
- Malware execution termination logs
- Batch script logs
- Deleted batch script file
- Network connections established by GoogleUpdate.exe and Goopdate.dll
- Network connections established by UPX packed NSIS installer
- Network connections established by Firefox installer trojanised
- Registry modifications for persistence mechanism
- Files created in ProgramData directory
- Files deleted from original directory
- System configuration changes for time delay mechanism
- Persistence mechanism installed
- Malicious files copied to ProgramData directory
- Malicious files removed from extracted Firefox files directory
- Disk size check fails if disk size is less than 1GB
- Physical memory size check fails if physical memory size is less than 2GB
- Logical processor check fails if number of logical processors is less than or equal to 2
- Processes running on the system indicate reverse engineering or debugging
- Anti-sandbox/anti-VM techniques implemented
- Session ID field in the header is set for request ID 0x2E
- Logs of persistence mechanism installation
- Files copied to ProgramData directory
- Files removed from extracted Firefox files directory
- Disk size check logs
- Physical memory size check logs
- Logical processor check logs
- Process logs indicating reverse engineering or debugging
- Anti-sandbox/anti-VM technique logs
- Session ID field logs in the header for request ID 0x2E
- Network connections for file transfer to ProgramData directory
- Network connections for file removal from extracted Firefox files directory
- System configuration changes for anti-sandbox/anti-VM techniques
- System configuration changes for session ID field in the header
- Files created for task IDs in different versions of the backdoor
- Files created for placeholders for future tasking
- Persistence mechanism installed
- Malicious files copied to ProgramData directory
- Malicious files removed from extracted Firefox files directory
- Service started for persistence
- Physical memory size of the machine exceeds 2GB
- Disk size exceeds 1GB
- Malicious DLL Goopdate.dll loaded by GoogleUpdate.exe
- UPX packed NSIS installer executed
- Trojanised Firefox installer executed
- Malware checks for processes running on the system
- Malware checks for debugger running on the system
- Malware checks for system time
- Malware checks for running processes and system properties

- Logs of service started for persistence
- Files copied to ProgramData directory
- Files removed from extracted Firefox files directory
- Loaded DLL Goopdate.dll
- Executable files of UPX packed NSIS installer
- Executable files of Trojanised Firefox installer
- Network connections to download UPX packed NSIS installer
- Network connections to download Trojanised Firefox installer
- System registry modifications for persistence mechanism
- System files modified to check for debugger and system time
- System files modified to check for running processes and system properties
- System logs of malware execution
- System logs of sandbox detection checks
- Files created by malware in ProgramData directory
- Files created by malware in extracted Firefox files directory
- Persistence of malware through service creation
- Malicious DLL loaded into memory
- Obfuscated files and information
- Deobfuscated/decoded files or information
- Sandbox detection and evasion
- Malware implementation of anti-VM techniques
- Creation of MD5 hash of concatenated adapter information and host name
- Logs of service creation
- Presence of malicious DLL (Goopdate.dll) in memory
- Obfuscated files (UPX packed NSIS installer)
- Decoded files or information (stack-based strings)
- Network connections to collect adapter information and host name
- Creation of MD5 hash file
- Files created by the NSIS installer (trojanised Firefox installer)
- Registry entries for service creation and persistence
- Presence of malware in system memory
- Logs of anti-VM techniques implementation
- The malware is installed on the infected machine.
- The malware is UPX packed and packaged in with a legitimate NSIS installer.
- The malware contains stack-based strings which are obfuscated with single byte XOR or subtraction throughout the binary.
- The malware contains RC4 encrypted binary embedded in the shellcode.
- The malware communicates over UDP using the KCP protocol.
- The malware can collect information about the infected machine or run additional plugins.
- The malware can execute plugins either as part of the current process, or by process hollowing dllhost.exe to execute the plugin.
- Logs of the malware installation and execution.
- Files created by the malware, including the UPX packed binary and the RC4 encrypted binary.
- Network connections to the C2 server over UDP using the KCP protocol.
- Network connections to the URL string in the backdoor under one-byte XOR obfuscation.
- Files created by the plugins executed by the malware.
- Registry entries created by the malware.
- Files created by the process hollowing dllhost.exe.
- Network connections to the HTTPS C2 server over RC4 encrypted communication.
- Files created by the RC4 encrypted binary.
- Logs of the plugin execution and data collection.
- Persistence mechanism installed
- Malicious files copied to ProgramData directory

- Malicious files removed from extracted Firefox files directory
- Malicious DLL (Goopdate.dll) loaded by legitimate executable (GoogleUpdate.exe)
- Process hollowing performed on dllhost.exe process
- Payload executable appears to run under legitimate process path and name
- Hidden window created
- Named pipe created with computer name MD5 hashed twice
- Legitimate dllhost.exe process created in suspended state
- Command data written into process memory
- Thread context changed to point at new data
- Thread resumed, causing new dllhost.exe process to execute payload data
- Logs of persistence mechanism installation
- Files copied to ProgramData directory
- Files removed from extracted Firefox files directory
- Loaded DLL (Goopdate.dll) in process memory
- Process hollowing logs in dllhost.exe process
- Hidden window created in task manager
- Named pipe created with computer name MD5 hashed twice
- Logs of legitimate dllhost.exe process creation in suspended state
- Command data written into process memory logs
- Thread context change logs
- Thread resume logs
- Network connections to C2 server
- Files created in UPX packed NSIS installer
- Trojanized Firefox installer logs
- Logs of Goopdate.dll loading by GoogleUpdate.exe
- Process exit logs due to window text check
- Logs of time-based evasion (CPU timestamp counter reads)
- Global structure logs for child process handling
- Logs of process termination (0x2F)
- Logs of payload execution in dllhost.exe process
- Malicious files are copied to the ProgramData directory.
- Malicious files are deleted from the original download location.
- The initial directory to which the files were downloaded will only contain the files the recipient likely intended to download.
- Malicious files are present in the ProgramData directory.
- The GoogleUpdate.exe process is re-started from the ProgramData directory.
- The batch script deletes itself.
- The malicious DLL Goopdate.dll is loaded by the legitimate signed executable file GoogleUpdate.exe.
- The files are moved to a legitimate looking directory.
- Malicious files in the ProgramData directory.
- Deleted files in the original download location.
- Logs of the GoogleUpdate.exe process re-starting from the ProgramData directory.
- Batch script logs.
- Loaded DLL (Goopdate.dll) in the GoogleUpdate.exe process.
- Network connections from the GoogleUpdate.exe process to the C2 server.
- Files in the legitimate looking directory.
- UPX packed NSIS installer in the original download location.
- Trojanised Firefox installer in the original download location.
- Logs of the process exit due to the window text check.
- Persistence of malicious DLL (Goopdate.dll) in the system.
- Creation of a Windows service for persistence.
- Hijacking of legitimate executable (GoogleUpdate.exe) for malicious purposes.

- Trojanisation of a legitimate Firefox installation package.
- Malicious communication over non-standard HTTPS port 4443.
- Process hollowing on dllhost.exe binary.
- Injection of content downloaded by C2 into dllhost.exe.
- Utilisation of rundll32.exe and url.dll for malicious purposes.
- Presence of Goopdate.dll in the system.
- Existence of a Windows service for persistence.
- Modification of GoogleUpdate.exe for malicious purposes.
- Presence of a trojanised Firefox installation package.
- Network connections over non-standard HTTPS port 4443.
- Modification of dllhost.exe binary for process hollowing.
- Presence of injected content in dllhost.exe.
- Logs of rundll32.exe and url.dll usage.
- Presence of malicious files in the system (e.g. shellcode).
- Network connections to C2 server.
- Malicious DLL (Goopdate.dll) is loaded by the legitimate signed executable file GoogleUpdate.exe.
- Malicious DLL is sideloaded by the legitimate, signed, executable GoogleUpdate.exe.
- Process Hollowing is performed on the dllhost.exe binary.
- Malicious DLL is injected into the dllhost.exe process.
- Rundll32.exe and url.dll are used to execute the legitimate binary which loads the malicious DLL.
- Malicious DLL is dropped alongside legitimate FireFox files.
- Named pipe is created with the name of the computer name MD5 hashed twice.
- An instance of the legitimate dllhost.exe process is created in the suspended state.
- Command data is written into the process memory.
- Thread context is changed to point at the new data and the thread is resumed.
- Malicious DLL (Goopdate.dll) is present in the system.
- GoogleUpdate.exe is modified to load the malicious DLL.
- dllhost.exe process is created in the suspended state.
- Command data is written into the dllhost.exe process memory.
- Named pipe is created with the name of the computer name MD5 hashed twice.
- Rundll32.exe and url.dll are present in the system.
- Malicious DLL is injected into the dllhost.exe process.
- Logs of the malicious activity are present in the system.
- Network connections to the C2 server are established.
- Files are created or modified to support the malicious activity (e.g. dllhost.exe, Rundll32.exe, url.dll).
- Persistence mechanism installed
- Malicious files copied to ProgramData directory
- Malicious DLL loaded by legitimate executable
- Process hollowing performed on dllhost.exe
- Malicious DLL sideloaded by GoogleUpdate.exe
- Malicious files dropped alongside legitimate Firefox files
- Non-standard HTTPS port 4443 opened
- Time-based evasion implemented
- Logs of process hollowing on dllhost.exe
- Logs of malicious DLL loading by legitimate executable
- Logs of malicious DLL sideloading by GoogleUpdate.exe
- Logs of non-standard HTTPS port 4443 usage
- Logs of time-based evasion implementation
- Malicious files in ProgramData directory
- Malicious DLL in system directory
- Modified dllhost.exe process in process listings
- Network connections to non-standard HTTPS port 4443

- Modified GoogleUpdate.exe executable
- Modified Firefox installation package
- Logs of CPU timestamp counter reads
- Modified system registry entries for persistence mechanism
- Modified system services for persistence mechanism

Attack Step 2.1

==========
Name: Masquerading: Match Legitimate Name or Location as used by Goofy Guineapig
Description: Goofy Guineapig masquerades as a legitimate Firefox installer and a Google updater to avoid detection. Here's how it's performed: Trojanized Firefox Installer: The malware is packaged within a seemingly legitimate Firefox installer. This tricks users into believing they are downloading and installing the legitimate Firefox browser. Google Update Mimicry: The malware also utilizes a legitimate GoogleUpdate.exe file, which is used for updating Google Chrome and other Google products. By loading its malicious DLL (Goopdate.dll) through this legitimate executable, Goofy Guineapig further blends in with normal system processes. This dual-pronged approach leverages the trust associated with well-known software to bypass security measures and gain a foothold on the victim's system.

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.
- The URL HTTPS://static.tcplog.com was observed.

Attack Step 2.2

==========
Name: Virtualization/Sandbox Evasion: Time Based Evasion as used by Goofy Guineapig
Description: Goofy Guineapig performs Virtualization/Sandbox Evasion: Time Based Evasion by: Checking the time register twice: It accesses the system's timekeeping mechanism. Measuring the delay: It calculates the time difference between the two checks. Comparing the delay to a threshold: It compares the measured delay to a predefined value (more than 100 milliseconds). Conditional execution: If the delay exceeds the threshold, it assumes the environment is likely a sandbox or analysis tool and halts its execution. This technique exploits the fact that sandboxes often artificially speed up time to accelerate testing. By detecting an unusually fast time progression, Goofy Guineapig attempts to avoid execution within a controlled environment.

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 Goofy Guineapig
Description: Goofy Guineapig 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 or processing units. If any of these checks reveal values that are atypical of a real system (often smaller or limited in sandboxes), Goofy Guineapig will halt its execution, avoiding further analysis.

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 Goofy Guineapig
Description: Goofy Guineapig performs Virtualization/Sandbox Evasion: User Activity Based Checks by monitoring the running processes on the infected system. Here's how it works: Process Monitoring: The malware actively scans the list of currently running processes. Keyword Detection: It specifically looks for process names or titles containing strings like "dbg", "debug", or "ida". These are common indicators used in debugging tools and reverse engineering environments. Execution Termination: If Goofy Guineapig detects any of these keywords, it interprets this as a sign that it's being analyzed in a

sandbox or debugging environment. As a result, it immediately terminates its own execution to avoid further scrutiny. Essentially, Goofy Guineapig tries to avoid detection by looking for telltale signs of analysis and shutting down if it suspects 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 file 'tmp.bat' is located in the directory 'C:\ProgramData\GoogleUpdate\GoogleUpdate'.
- A HTTP request with the User-Agent header 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36' was made.

Attack Step 2.5

==========
Name: Obfuscated Files or Information: Software Packing as used by Goofy Guineapig
Description: Goofy Guineapig employs UPX (Ultimate Packer for eXecutables) software packing. Here's how the action is performed: UPX Packing: The malicious code of Goofy Guineapig is compressed and encrypted using the UPX packer. This makes the malware's code harder to analyze and understand by security researchers. NSIS Installer Packaging: Goofy Guineapig is further disguised by being packaged within a legitimate Nullsoft Scriptable Install System (NSIS) installer. This creates a layer of legitimacy, making it more likely to be accepted by users. The combination of UPX packing and NSIS packaging serves two primary purposes: Evasion: By obfuscating the malware's code, UPX packing makes it more difficult for antivirus software and other security tools to detect and analyze the threat. Delivery: The NSIS installer provides a common and seemingly harmless way to distribute the malware, increasing the chances of it being executed by unsuspecting users. Let me know if you have any other questions about Goofy Guineapig or malware analysis techniques.

MITRE Technique

ID: T1027.001
Name: Obfuscated files or information: binary padding
Description: Adversaries may use binary padding to add junk data and change the on-disk representation of malware. This can be done without affecting the functionality or behavior of a binary, but can increase the size of the binary beyond what some security tools are capable of handling due to file size limitations.
More info: https://attack.mitre.org/techniques/T1027/001

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 '6f5c07c50ce4976ddb3879ce65d3b2f96693dc4c'.
- The file 'setup-stub.exe' has a SHA-256 hash of '97f66bcd7d73917a8b59d9a1dcac21a58936bcafa91e757a9dfb8e5c320af40f56'.
- 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 2.6

==========
Name: Deobfuscate/Decode Files or Information as used by Goofy Guineapig
Description: Goofy Guineapig obfuscates its stack-based strings using two primary methods: Single-Byte XOR: Each character in the string is XORed with a single byte value. This effectively scrambles the characters, making them unreadable without knowing the XOR key. Subtraction: Similar to XOR, subtraction is used to shift the characters in the string. A specific value is subtracted from each character's ASCII code, resulting in a shifted representation of the original string. To deobfuscate these strings, an analyst would need to: Identify the XOR key or subtraction value: This can be achieved by analyzing the binary's code for patterns or hardcoded values used in the obfuscation process. Apply the inverse operation: Once the key or value is known, the XOR or subtraction operation can be reversed on each character in the obfuscated string, revealing the original, meaningful data. Let me know if you'd like a more in-depth explanation of either XOR or subtraction as a string obfuscation technique.

MITRE Technique

ID: T1027.008
Name: Obfuscated files or information: stripped payloads
Description: Adversaries may attempt to make a payload difficult to analyze by removing symbols, strings, and other human readable information. Scripts and executables may contain variables names and other strings that help developers document code functionality. Symbols are often created by an operating systemâ€™s linker when executable payloads are compiled. Reverse engineers use these symbols and strings to analyze code and to identify functionality in payloads.
More info: https://attack.mitre.org/techniques/T1027/008

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 '6f5c07c50ce4976ddb3879ce65d3b2f96693dc4c'.

- 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 '628ce6721b97fa12590356712fbfc5ae030781ce'.
- The file 'config.dat' has a SHA-256 hash of '3a1af09a0250c602569d458e79db90a45e305b76d8423b81eeeca14c69847b81c'.

Attack Step 2.7

==========

Name: Hide Artifacts: Hidden Window as used by Goofy Guineapig
Description: Goofy Guineapig hides its artifacts by using a technique called process hollowing on the legitimate dllhost.exe process. Here's how it works: Target Selection: Goofy Guineapig identifies the dllhost.exe process as its target for hollowing. Process Hollowing: Goofy Guineapig creates a new process instance of dllhost.exe. It then replaces the original code of the dllhost.exe process with its own malicious code. This effectively "hollows out" the legitimate dllhost.exe process, making it a vessel for the malware's activities. Hidden Execution: Crucially, Goofy Guineapig creates this hollowed dllhost.exe process hidden. This means the process doesn't appear in the usual task manager or process listing views, making it harder to detect. Malicious Activity: The hollowed dllhost.exe process now executes the malware's code, allowing it to perform its malicious tasks without raising immediate suspicion. Let me know if you'd like more details on any specific aspect of process hollowing!

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

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

Attack Step 2.8

==========

Name: Indicator Removal on Host: File Deletion as used by Goofy Guineapig
Description: Goofy Guineapig 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 legitimate-looking directory on the system. File Deletion: Finally, Goofy Guineapig deletes the original files from the initial download location, effectively removing traces of its presence. This action aims to evade detection by security tools that might be monitoring file changes in the download directory.

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

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

Attack Step 2.9

==========
Name: Hijack Execution Flow: DLL Side-Loading as used by Goofy Guineapig
Description: Here's a breakdown of how Goofy Guineapig hijacks execution flow using DLL side-loading: 1. The Setup: Legitimate Executable: Goofy Guineapig's loader installs a legitimate, signed executable (in this case, GoogleUpdate.exe). This executable is chosen because it's trusted by the system and is likely to be run. Malicious DLL: Alongside the legitimate executable, Goofy Guineapig also drops a malicious DLL (the actual payload). 2. The Trick: Side-Loading: When the legitimate executable (GoogleUpdate.exe) runs, Goofy Guineapig's loader ensures that the malicious DLL is loaded into the process's memory space. This happens without the user's knowledge or consent. 3. The Result: Compromised Process: The legitimate executable is now effectively compromised. The malicious DLL can: Execute its own code: It gains access to the resources and privileges of the legitimate process. Modify the legitimate process's behavior: It can alter the way the legitimate executable functions, potentially stealing data, installing other malware, or performing other malicious actions. Why This Works: Trust: The system trusts the legitimate executable, so it doesn't question the presence of the malicious DLL. Hidden Execution: The malicious DLL runs within the context of the legitimate process, making it harder to detect. Key Points: DLL side-loading is a common technique used by malware to gain a foothold on a system. It exploits the trust that operating systems place in legitimate software. Detecting and preventing DLL side-loading requires careful monitoring of process behavior and file system activity. Let me know if you have any other questions.

MITRE Technique

ID: T1574.002
Name: Hijack execution flow: dll side-loading
Description: Adversaries may execute their own malicious payloads by side-loading DLLs. Similar to DLL Search Order Hijacking, side-loading involves hijacking which DLL a program loads. But rather than just planting the DLL within the search order of a program then waiting for the victim application to be invoked, adversaries may directly side-load their payloads by planting then invoking a legitimate application that executes their payload(s).
More info: https://attack.mitre.org/techniques/T1574/002

Indicators

- The file 'Goopdate.dll' was observed.
- The file 'config.dat' was observed.

Attack Step 2.10

==========
Name: Process Injection: Process Hollowing as used by Goofy Guineapig
Description: Goofy Guineapig performs Process Injection: Process Hollowing by targeting the dllhost.exe binary. Here's a breakdown of the action: Download: Goofy Guineapig downloads malicious content from its Command and Control (C2) server. Process Hollowing: It injects this downloaded content into a running instance of dllhost.exe. This process involves: Allocating Memory: Goofy Guineapig allocates a new memory region within the dllhost.exe process. Overwriting Existing Code: It overwrites the original code of dllhost.exe with its malicious payload. Redirecting Execution: The execution flow of dllhost.exe is redirected to the beginning of the injected malicious code. Persistence: The hollowed dllhost.exe now runs the malicious code, effectively giving Goofy Guineapig persistence on the infected system. Why dllhost.exe? dllhost.exe is a legitimate Windows process responsible for hosting Dynamic Link Libraries (DLLs). By hollowing it, Goofy Guineapig can: Blend In: The malicious activity appears to originate from a legitimate system process, making it harder to detect. Access Resources: dllhost.exe has certain privileges that the malware can leverage. Let me know if you have any other questions.

MITRE Technique

ID: T1055
Name: Process injection
Description: Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process.
More info: https://attack.mitre.org/techniques/T1055/

Indicators

- Path: C:\ProgramData\GoogleUpdate\GoogleUpdate\tmp.bat
- User Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36

Attack Step 2.11

==========
Name: Signed Binary Proxy Execution: Rundll32 as used by Goofy Guineapig
Description: Goofy Guineapig uses the rundll32.exe utility and the url.dll library to execute a legitimate binary. This legitimate binary is then tricked into loading a malicious DLL, establishing persistence on the infected system. Here's a breakdown of the process: Leveraging rundll32.exe: rundll32.exe is a legitimate Windows tool used to run functions within DLL files. Using url.dll: Goofy Guineapig instructs rundll32.exe to use the url.dll library. This library typically handles URL protocol handling. Loading the Malicious DLL: Through a combination of arguments and manipulation, Goofy Guineapig convinces the legitimate binary to load its malicious DLL. This DLL then executes its malicious payload, establishing persistence. Essentially, Goofy Guineapig exploits the functionality of legitimate system tools to achieve its malicious goals.

MITRE Technique

ID: T1218.011
Name: System binary proxy execution: rundll32
Description: Adversaries may abuse rundll32.exe to proxy execution of malicious code. Using rundll32.exe, vice executing directly (i.e. Shared Modules), may avoid triggering security tools that may not monitor execution of the rundll32.exe process because of allowlists or false positives from normal operations. Rundll32.exe is commonly associated with executing DLL payloads (ex: rundll32.exe {DLLname, DLLfunction}).
More info: https://attack.mitre.org/techniques/T1218/011

Indicators

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

Milestone 3

Pre-Conditions:

- The malware Goofy Guineapig is present on the infected machine.
- The malware has access to the Windows operating system.
- The malware has access to the Windows APIs.
- The malware has access to the COM interface.
- The malware has access to the WMI information.
- The malware has access to the network.
- The malware has access to the HTTPS protocol.
- The malware has access to the HTTP protocol.
- The malware has access to the TCP socket.
- The malware has access to the MD5 hashing algorithm.
- Windows operating system.
- Access to the Windows APIs.
- Access to the COM interface.
- Access to the WMI information.
- Access to the network.
- Access to the HTTPS protocol.
- Access to the HTTP protocol.
- Access to the TCP socket.
- Access to the MD5 hashing algorithm.
- A mechanism to collect and store the system information (e.g. a database or a file).
- A mechanism to transmit the system information over the network (e.g. a network connection).
- A mechanism to receive and process the system information (e.g. a C2 server).

Post-Conditions:

- Malicious DLL (Goopdate.dll) is loaded by a legitimate executable (GoogleUpdate.exe).
- A trojanised Firefox installer is created.
- Malware collects system information about the infected machine.
- Malware sends collected information in HTTP headers.
- Malware uses HTTPS for C2 communications.
- Malware collects adapter information and host name, concatenates them, and takes an MD5 hash.
- Malware collects operating system caption and antivirus product display name using COM to access WMI information.
- Malware collects host and computer name using Windows APIs.
- Loaded DLL (Goopdate.dll) in the process list.
- Presence of a trojanised Firefox installer on the system.
- HTTP headers containing system information.
- HTTPS connections to C2 servers.
- MD5 hash of concatenated adapter information and host name in the malware's data storage.
- WMI information accessed by the malware.
- Windows API calls made by the malware to collect system information.
- Malware's configuration files containing collected system information.
- Network connections to C2 servers.
- Logs of malware's activities, including system information collection and C2 communications.

Attack Step 3.1

==========
Name: System Information Discovery as used by Goofy Guineapig
Description: Goofy Guineapig performs System Information Discovery by collecting various details about the infected machine and embedding them within an obfuscated "Authorization" string in the HTTP header of each communication with its Command and Control (C2) server. Here's a breakdown of the process: Data Collection: Goofy Guineapig gathers information about the infected system using a combination of methods: COM and WMI: It leverages COM (Component Object Model) to access Windows Management Instrumentation (WMI), allowing it to retrieve the operating system caption and the name of the installed antivirus product. Windows APIs: It utilizes various Windows APIs to collect other system information, such as adapter details, host and hostname, and computer name. Obfuscation: The collected system information is then obfuscated before being included in the "Authorization" string. This likely involves techniques like: Encoding: Converting the data into a different format, such as hexadecimal or base64. XORing: Applying a bitwise XOR operation with a secret key to scramble the data. Transmission: The obfuscated "Authorization" string is sent to the C2 server within the HTTP header of each communication. This allows the malware to transmit sensitive system information back to its controllers without raising immediate suspicion. Analysis: The C2 server receives the obfuscated data and likely decodes it to extract the valuable system information. This information can then be used for various malicious purposes, such as: Targeting: Identifying vulnerable systems for further attacks. Credential theft: Determining the type of antivirus software installed to potentially bypass security measures. Command and control: Tailoring future commands based on the system's configuration. Let me know if you have any other questions.

MITRE Technique

ID: T1082
Name: System information discovery
Description: An adversary may attempt to get detailed information about the operating system and hardware, including version, patches, hotfixes, service packs, and architecture. Adversaries may use the information from System Information Discovery during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.
More info: https://attack.mitre.org/techniques/T1082/

Indicators

- The file 'config.dat' was located at 'C:\ProgramData\GoogleUpdate\GoogleUpdate'.
- 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 used.

Milestone 4

Pre-Conditions:

- The system has a physical memory size exceeding 2GB.
- The disk is more than 1GB in size.
- A debugger is not running.
- Other analysis tools are not running.
- The system has a network connection.
- The system has a HTTPS protocol implementation.
- The system has a HTTP protocol implementation.
- The system has a TCP protocol implementation.
- The system has a UDP protocol implementation.
- The system has a KCP protocol implementation.
- Environment: A Windows system with a physical memory size exceeding 2GB and a disk size more than 1GB.
- Tools: A debugger and other analysis tools are not required, but a tool to analyze network traffic is necessary.
- Connectivity: A network connection is required.
- Resources: A HTTPS protocol implementation, a HTTP protocol implementation, a TCP protocol implementation, a UDP protocol implementation, and a KCP protocol implementation are required.
- The system must have a way to send and receive HTTPS GET and POST requests.
- The system must have a way to send and receive HTTP requests.
- The system must have a way to send and receive UDP packets using the KCP protocol.
- The system must have a way to send and receive raw TCP socket communications.
- An embedded configuration string is present.
- The embedded configuration string contains UDP rather than HTTP (S).
- The embedded configuration string is hardcoded in the binary under a single byte XOR with the key 0x59.
- The embedded configuration string is split by the pipe character.
- The following strings are searched for in order to determine the communication type that should be utilised: 'HTTP', 'http', 'UDP', 'udp'.
- A binary with the embedded configuration string is available.
- The binary is loaded by a legitimate signed executable file (GoogleUpdate.exe).
- The binary is bundled in a UPX packed NSIS installer which is a trojanised Firefox installer.
- The environment has a physical memory size exceeding 2GB.
- The environment has a disk size exceeding 1GB.
- The malware is able to communicate over UDP using the KCP protocol or direct socket communications.
- The malware is able to communicate over a non-standard port (4443).
- The malware is able to communicate using HTTPS with no proxy.
- The malware has access to the internet to communicate with the URL specified in the embedded configuration string.
- The malware has the necessary resources to process and handle the communication data.
- The embedded configuration URL is present.
- The embedded configuration URL selects HTTPS as the transport method.
- The embedded configuration URL does not specify a proxy.
- The transport method is selected based on the embedded configuration URL.
- The underlying command data structure and processing remain consistent across transport protocols.

- The HTTPS port 4443 is available.
- The system has a network connection.
- The system has a Windows service.
- Environment: Windows operating system.
- Tools: None explicitly stated, but likely requires a network connection and a system with a Windows service.
- Connectivity: Network connection.
- Resources: Available HTTPS port 4443.
- Availability of the embedded configuration URL.
- The system must be able to load a malicious DLL.
- The system must be able to maintain persistence using a Windows service.
- The system must be able to communicate over port 4443.

Post-Conditions:

- Malicious DLL (Goopdate.dll) is loaded by a legitimate executable (GoogleUpdate.exe).
- A trojanised Firefox installer is created.
- The infected machine sends information about itself in each C2 packet.
- HTTPS communications are used for C2 communications.
- The binary implements basic anti-sandbox/anti-VM techniques.
- The final command in the temp.bat script deletes the binary itself.
- A UPX packed NSIS installer is created.
- A temp.bat script is created.
- A file named Goopdate.dll is created.
- A file named GoogleUpdate.exe is created.
- A Firefox installer is modified to be trojanised.
- Network connections to HTTPS servers (e.g. static.tcplog.com:4443) are established.
- Logs of C2 communications are created.
- Files with obfuscated 'Authorization' strings are created.
- Files with null-padded file paths are created.
- A raw TCP socket connection is established.
- Malicious DLL (Goopdate.dll) is loaded by a legitimate executable (GoogleUpdate.exe).
- A UPX packed NSIS installer is installed, which is a trojanized Firefox installer.
- The malware collects information about the infected machine.
- The malware communicates with a C2 server over UDP using the KCP protocol or direct socket communications.
- The malware communicates with a C2 server over HTTPS on port 4443.
- The malware uses a proxy option for HTTP/S communications.
- The malware includes computer information in the 'Authorization:' header in the HTTP headers.
- The malware collects adapter information and host name, concatenates them, and takes an MD5 hash of the result.
- Loaded DLL (Goopdate.dll) is visible in the process list.
- The UPX packed NSIS installer is installed on the system.
- Network connections to the C2 server over UDP using the KCP protocol or direct socket communications.
- Network connections to the C2 server over HTTPS on port 4443.
- Proxy logs for HTTP/S communications.
- HTTP headers with computer information in the 'Authorization:' header.
- Files created by the malware, such as logs or configuration files.
- Registry entries created by the malware.
- Network traffic logs showing the communication with the C2 server.
- System logs showing the malware's activity.

- Files created by the malware, such as logs or configuration files.
- MD5 hash of the concatenated adapter information and host name.
- Malicious DLL (Goopdate.dll) is loaded by a legitimate signed executable (GoogleUpdate.exe).
- Persistence is maintained using a Windows service.
- Malware communicates over a non-standard HTTPS port (4443).
- Malware uses UDP and KCP protocol, or direct socket communications.
- Malware collects information using Windows APIs.
- Malware stores adapter information and host name in an MD5 hash.
- Loaded DLL (Goopdate.dll) in the Windows process list.
- Windows service created for persistence.
- Network connections to non-standard HTTPS port (4443).
- UDP and KCP protocol, or direct socket communications.
- Logs of Windows API calls made by the malware.
- MD5 hash stored in a file or registry key.
- UPX packed NSIS installer (trojanised Firefox installer) on the system.
- Malicious files (Goopdate.dll, GoogleUpdate.exe, UPX packed NSIS installer) on the system.
- Registry keys modified for Windows service creation.
- System logs indicating malware activity.

Attack Step 4.1

==========
Name: Application Layer Protocol: Web Protocols as used by Goofy Guineapig
Description: Goofy Guineapig utilizes HTTPS as the protocol for its communication with its Command and Control (C2) server.

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

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

Attack Step 4.2

==========
Name: Fallback Channels as used by Goofy Guineapig
Description: Goofy Guineapig utilizes fallback channels for communication, employing a strategy based on an embedded configuration string. Here's how it works: Configuration String: Goofy Guineapig possesses a built-in configuration string that dictates its preferred communication methods. Communication Protocols: Based on the configuration string, the malware can utilize the following fallback channels: UDP: A common internet protocol for sending datagrams. KCP: A fast and reliable transport protocol designed for low-latency and high-throughput applications. Direct Socket

Communications: Establishing direct connections between the malware and its command-and-control (C&C;) server using sockets. Dynamic Selection: The malware dynamically selects the communication method based on the configuration string, allowing it to adapt to network conditions and security measures. This fallback channel approach enhances Goofy Guineapig's resilience by providing alternative communication pathways if one method is blocked or disrupted.

MITRE Technique

ID: T1008
Name: Fallback channels
Description: Adversaries may use fallback or alternate communication channels if the primary channel is compromised or inaccessible in order to maintain reliable command and control and to avoid data transfer thresholds.
More info: https://attack.mitre.org/techniques/T1008/

Indicators

- The URL HTTPS://static.tcplog.com was observed.
- The file tmp.bat was located at C:\ProgramData\GoogleUpdate\GoogleUpdate\tmp.bat.

Attack Step 4.3

==========
Name: Non-Standard Port as used by Goofy Guineapig
Description: Goofy Guineapig communicates over the non-standard HTTPS port 4443. This means it deviates from the standard port (port 443) typically used for HTTPS traffic. Instead, it uses port 4443 to establish secure connections, potentially making it harder to detect by traditional security measures that focus on standard ports.

MITRE Technique

ID: T1571
Name: Non-standard port
Description: Adversaries may communicate using a protocol and port pairing that are typically not associated. For example, HTTPS over port 8088 or port 587 as opposed to the traditional port 443. Adversaries may make changes to the standard port used by a protocol to bypass filtering or muddle analysis/parsing of network data.
More info: https://attack.mitre.org/techniques/T1571/

Indicators

No indicators found.