Enhanced Attack Report

Goofy Guineapig

Generated on 2025-03-06

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.

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 (T1036.005)
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 (T1027.002)
6. Deobfuscate/Decode Files or Information as used by the malware (T1140)
7. Hide Artifacts: Hidden Window as used by the malware (T1564.003)
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 (T1071.001)
2. Fallback Channels as used by the malware (T1573)
3. Non-Standard Port as used by the malware (T1571)

Milestone 1

Pre-Conditions:

- A system with the necessary privileges to create or modify a Windows service.
- The presence of the necessary APIs (e.g. Windows APIs) to interact with the system.
- The ability to execute the malware on the system.
- The presence of the necessary tools (e.g. NSIS installer) to create or modify a Windows service.
- Connectivity to the system (e.g. network connectivity) is not required.
- The malware has been executed on the system.
- The system has the necessary resources to support the creation or modification of a Windows service.
- The system has a Windows operating system.
- A system with a disk size exceeding 1GB.
- The system has a physical memory size exceeding 2GB.
- The system does not have a process containing the string 'dbg', 'debug', or 'ida' running.

Post-Conditions:

- Modified CPU timestamp counter values
- Network connections to C2 server
- Sandbox detection and anti-analysis techniques implemented
- Persistence mechanism through a Windows service
- Files created by UPX packed NSIS installer
- Logs of process creation and modification
- Malicious DLL (Goopdate.dll) loaded by a legitimate executable (GoogleUpdate.exe)
- Time-based evasion through CPU timestamp counter manipulation
- MD5 hash of concatenated adapter information and host name
- Process hollowing on the dllhost.exe process
- Malware execution halted if debugger or analysis tools detected
- Registry modifications for persistence mechanism

Attack Step 1.1

==========

Name: Create or Modify System Process: Windows Service as used by the malware
Description: Goofy Guineapig establishes persistence on the infected machine by creating and configuring a Windows service. Here's a breakdown of how this action is likely performed: 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 path (which points to the Goofy Guineapig DLL itself). Service Configuration: The malware configures the service's startup type (e.g., automatic, manual) and other parameters like service account and dependencies. This ensures the service starts automatically when the system boots or on demand. Registration: The malware registers the service with the Windows service control manager, making it a permanent part of the system. Persistence: Once registered, the Windows service will automatically start and run in the background, keeping the Goofy Guineapig malware active even after the user logs off or restarts the system. Why this is effective: System-Level Execution: Windows services run with elevated privileges, allowing the malware to operate with greater access to system resources. Automatic Startup: The automatic startup ensures the malware persists even if the user manually deletes its files. Stealth: Services often run in

the background without user interaction, making them harder to detect and remove. 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 malware has the capability to bundle itself with legitimate files.
- The malware has access to a legitimate GoogleUpdate.exe executable.
- Environment: Windows operating system.
- The malware has the capability to modify or create a Windows service.
- Tools: NSIS installer, FireFox installation package, GoogleUpdate.exe executable.
- The malware has the capability to deploy itself through social engineering.
- Capability to modify or create a Windows service.
- The malware has access to a legitimate FireFox installation package.
- Resources: Access to legitimate FireFox and GoogleUpdate.exe files, capability to modify or create a Windows service.
- Connectivity: None.
- Availability of a legitimate FireFox NSIS installation package.
- The malware has the capability to sideload a malicious DLL.
- The malware has the capability to masquerade as a legitimate process.
- The ability to check the disk size.
- The system has a functioning clock.
- The time difference between the two registrations is less than 100 milliseconds.
- The ability to check the system time.
- The ability to implement a delay of more than 100 milliseconds.
- The system time is registered twice.
- The system has a functioning operating system.
- A system with a physical memory size exceeding 2GB.
- The system has a functioning memory.
- The ability to check the physical memory size.
- The system has logical processors.
- A system with a sizeable disk (more than 1GB).
- The system has a disk.
- The physical memory size is available.
- A system with a number of logical processors greater than 2.
- The system has the necessary environment to run the malware.
- The disk size is available.
- The number of logical processors is available.
- The system has physical memory.
- The system has the necessary tools to run the malware's checks (e.g. GetLocalTime API).
- A system with a legitimate browser (for the Jolly Jellyfish malware).
- The system has processes running.
- The system has a user activity.
- The system has a system time.
- A system with a debugger (for the 'dbg', 'debug', or 'ida' string check).
- A system with a network connection (for the C2 communication).
- A system with a UPX packed NSIS installer (for the trojanised Firefox installer).
- A system with a legitimate signed executable file (for the GoogleUpdate.exe).
- The system has a physical memory size.
- A system with a name for each running process.
- The system has a disk size.
- A system with a legitimate executable file (for the Goopdate.dll).

- A system with a sandbox or virtual machine detection tool.
- A system with a legitimate operating system.
- The malware is UPX packed.
- The malware is available in the environment.
- The UPX tool is available in the environment.
- The malware is packaged in with a legitimate NSIS installer.
- The NSIS installer is a trojanised Firefox installer.
- The NSIS installer is available in the environment.
- The malware is a binary file.
- The environment has a physical memory size exceeding 2GB.
- Tools: UPX, NSIS, and a debugger (to analyze the malware).
- Connectivity: None.
- The environment has a trojanised Firefox installer available.
- Environment: A Windows environment with a physical memory size exceeding 2GB and a disk size exceeding 1GB.
- The binary file is executable.
- The binary file contains stack-based strings.
- Resources: None.
- The malware checks the physical memory size of the machine and the disk size.
- Availability of the malware sample.
- The malware uses HTTPS and RC4 encrypted C2 communications with a key.
- Resources: A system with sufficient resources to run the malware and the tools required to deobfuscate and decode the files or information.
- Availability of the tools required to deobfuscate and decode the files or information.
- Knowledge of the malware's behavior and the specific obfuscation techniques used.
- The malware contains stack-based strings which are obfuscated with single byte XOR or subtraction throughout the binary.
- Environment: A system with a malware sample containing the obfuscated files or information.
- The malware checks for the presence of a debugger or other analysis tools.
- The malware checks the name of each running process on the machine.
- Connectivity: None.
- The malware embeds a binary in the shellcode which is RC4 encrypted with a key.
- The malware has a hardcoded configuration string under a single byte XOR with a key.
- Tools: A disassembler, a debugger, and a tool capable of performing single byte XOR or subtraction decryption.
- The malware is UPX packed and packaged in with a legitimate NSIS installer.
- Connectivity: None required.
- The system has a debugger or other analysis tools running.
- The system has a legitimate GoogleUpdate.exe executable.
- The system has a legitimate binary that can be used for process hollowing.
- The system has a dllhost.exe process.
- The system has a url.dll process.
- Environment: A Windows system with a legitimate FireFox installation and a legitimate GoogleUpdate.exe executable.
- Resources:
- None of the requirements are directly stated, but they can be inferred from the context.
- Tools: A debugger or other analysis tools to detect the presence of the malware.
- The system has a rundll32.exe process.
- The malware is running on a system.
- The system has a legitimate FireFox installation.
- The system has a physical memory size exceeding 2GB.
- The malware has been downloaded to a location on the host.
- The capability to delete the batch script.

- The malware has the capability to modify the batch script.
- The presence of the GetLocalTime API.
- The malware has established persistence on the host.
- The malware has access to the directory containing the extracted Firefox files.
- The malware has access to the ProgramData directory.
- A Windows environment with a 64-bit architecture.
- The presence of the batch script.
- The capability to copy files on the host.
- The capability to write to the ProgramData directory.
- The malware has the capability to restart the GoogleUpdate.exe process.
- The presence of the GoogleUpdate.exe and Goopdate.dll files.
- The malware has the capability to delete files on the host.
- The capability to write to the directory containing the extracted Firefox files.
- A system with a physical memory size exceeding 2GB.
- The legitimate executable is dropped alongside the malicious DLL.
- The legitimate executable is signed.
- Availability of the CPU timestamp counter.
- The Goofy Guineapig loader is present.
- Availability of the DLLHost.exe binary.
- The Goofy Guineapig loader has trojanised a legitimate FireFox NSIS installation package.
- Windows operating system.
- Presence of the legitimate executable (GoogleUpdate.exe).
- Availability of the GetLocalTime API.
- A legitimate executable is installed.
- Connectivity to the internet (for downloading content).
- The legitimate executable is installed by the Goofy Guineapig loader.
- Availability of the Rundll32.exe binary.
- Availability of the URL.dll binary.
- Availability of the Windows service mechanism.
- Resources: The machine has the necessary resources to perform process hollowing, including memory and CPU resources.
- Tools: The malware has the necessary tools to perform process hollowing, including the ability to inject content into the dllhost.exe binary.
- The machine has a physical memory size exceeding 2GB.
- The machine is not running a process containing the string 'dbg', 'debug', or 'ida'.
- Environment: A Windows machine with a physical memory size exceeding 2GB and a disk size exceeding 1GB.
- The machine has the necessary permissions to download content from the C2 server.
- The malware is running on a machine.
- The rundll32.exe binary is present on the machine.
- The url.dll binary is present on the machine.
- The machine has the necessary permissions to inject content into the dllhost.exe binary.
- The machine is not running a debugger.
- The malware has been sideloaded by the legitimate, signed, executable GoogleUpdate.exe.
- Capability to inject content into the dllhost.exe binary.
- The malware has maintained persistence using a Windows service.
- The malware has the capability to inject content downloaded by the C2 into the dllhost.exe binary.
- Availability of the legitimate FireFox NSIS installation package and GoogleUpdate.exe.
- The malware has been successfully trojanised a legitimate FireFox NSIS installation package.
- Capability to maintain persistence using a Windows service.
- The malware has the capability to perform process hollowing on the dllhost.exe binary.
- Windows operating system.
- Rundll32.exe and url.dll tools.

- The malware has been loaded by a legitimate signed executable.
- Capability to execute the legitimate binary which will load the malicious DLL.
- Availability of the dllhost.exe binary.
- Connectivity to the C2 server for downloading content.

Post-Conditions:

- Network connections to the C2 infrastructure (static.tcplog.com:4443).
- Creation of a legitimate FireFox installation package with the malware bundled inside.
- Windows service logs with a legitimate name or location.
- Sideload of the malicious DLL by the legitimate GoogleUpdate.exe executable.
- Modification of the system to use the KCP protocol for UDP communications.
- Persistence of the Goofy Guineapig malware as a Windows service.
- Configuration string hardcoded in the binary with a single byte XOR.
- Files created by the malware with MD5 hashes (MD5(MD5(MD5(MD5(ComputerName))))).
- Indicators of compromise in the form of a hardcoded configuration string.
- Creation of a Windows service with a legitimate name or location.
- Deployment of the malware through social engineering.
- Logs of the Windows service creation and modification.
- Malware checks for processes indicating reverse engineering or debugging.
- System registry modifications (e.g., adding malware configuration keys).
- Malware implements anti-sandbox and anti-VM techniques.
- Malware process logs in the system event logs.
- MD5 hash of concatenated adapter information and host name in the system logs.
- Malware returns gathered information to a C2 server in a pipe-separated string.
- Malicious DLL (Goopdate.dll) is loaded by a legitimate executable (GoogleUpdate.exe).
- Malware will not continue execution if more time has elapsed or if checks fail.
- Malware configuration files (e.g., registry keys, configuration files).
- NSIS installer with a trojanized Firefox installer.
- Pipe-separated string logs in the system logs.
- Malware executable files (e.g., Goopdate.dll).
- Loaded DLL (Goopdate.dll) in the process list.
- System information collection logs in the Windows Event Viewer.
- Malware concatenates adapter information and host name, then takes an MD5 hash of the result.
- Network traffic logs (e.g., firewall logs, network packet captures).
- UPX packed files.
- Network connections to the C2 server.
- Malware collects system information using Windows APIs.
- Malware checks the time register twice for a delay of more than 100 milliseconds.
- Files created by UPX packed NSIS installer.
- Log entries indicating malware execution and behavior.
- Process creation of Goofy Guineapig malware.
- Malware (Goofy Guineapig) checks for sandbox environments and exits if detected.
- Log entries indicating sandbox detection and evasion techniques.
- Process creation of GoogleUpdate.exe.
- Malware (Goofy Guineapig) returns gathered information to a C2 server.
- Loaded DLL (Goopdate.dll) in memory.
- Malware (Goofy Guineapig) checks for logical processor count and exits if it is below a certain threshold.
- Network traffic from Goofy Guineapig malware.
- Network connection to C2 server.

- Malware (Goofy Guineapig) checks for physical memory size and disk size, and exits if they are below certain thresholds.
- Malicious DLL (Goopdate.dll) is loaded by a legitimate executable (GoogleUpdate.exe).
- Registry entries modified by malware.
- Malware (Goofy Guineapig) queries client address and concatenates information into a pipe-separated string.
- The malware checks that the physical memory size of the machine exceeds 2GB and that the disk is more than 1GB in size.
- Registry entries created by the malicious DLL Goopdate.dll.
- Files created by the UPX packed NSIS installer.
- The malware will not continue execution if any of the sandbox detection checks fail.
- Network connections made by the trojanised Firefox installer.
- Logs of the malware's sandbox detection checks.
- Process creation logs of the malware.
- The malware checks the properties of the infected machine, running processes, and system time for any indication the process is running in an automated analysis environment.
- The malware checks the name of each running process on the machine and will not continue execution if any process containing the string 'dbg', 'debug', or 'ida' is determined to be running.
- The malware is bundled in a UPX packed NSIS installer which is a trojanised Firefox installer.
- Disk space usage logs showing the malware's disk space requirements.
- System time and date modified by the malware.
- Malicious DLL Goopdate.dll is loaded by the legitimate signed executable file GoogleUpdate.exe.
- The malware implements basic anti-sandbox / anti-virtual machine (VM) techniques.
- Malware creates an MD5 hash of the concatenated adapter information and host name.
- Process exit logs due to window text checks.
- Files created by the UPX packed NSIS installer.
- Loaded DLL (Goopdate.dll) in the process list.
- Malware implements anti-sandbox and anti-VM techniques.
- Logs of anti-sandbox and anti-VM checks.
- Malware checks physical memory size and disk size to evade detection.
- Malware is UPX packed and packaged in a legitimate NSIS installer.
- Malware communicates with a hardcoded URL (HTTPS://static.tcplog.com:4443) using HTTP or UDP protocol.
- Network connections to HTTPS://static.tcplog.com:4443.
- Malicious DLL (Goopdate.dll) is loaded by a legitimate executable (GoogleUpdate.exe).
- Malware contains stack-based strings obfuscated with single byte XOR or subtraction.
- Malware collects host information using Windows APIs.
- Logs of disk and physical memory size checks.
- The malware Goofy Guineapig embeds a binary in the shellcode which is RC4 encrypted with the key: 2UFdRF06kYvIXWOW.
- The malware Goofy Guineapig implements basic anti-sandbox / anti-virtual machine (VM) techniques.
- The malware Goofy Guineapig communicates with C2 servers over HTTPS and RC4 encrypted with the key: uirWmX3fSBhplR2sj.
- The malware Goofy Guineapig uses the KCP protocol for UDP communications.
- Network connections to C2 servers over UDP using the KCP protocol.
- Configuration string hardcoded in the binary under single byte XOR with the key 0x59.
- The malware Goofy Guineapig contains stack-based strings which are obfuscated with single byte XOR or subtraction throughout the binary.
- Network connections to C2 servers over HTTPS and RC4 encrypted with the key: uirWmX3fSBhplR2sj.
- RC4 encrypted binary embedded in the shellcode.
- Obfuscated files or information in the malware binary.

- Logs of the malware implementing anti-sandbox / anti-VM techniques.
- UPX packed NSIS installer files.
- Logs of the malware using the KCP protocol for UDP communications.
- The malware Goofy Guineapig is UPX packed and packaged in with a legitimate NSIS installer.
- Logs of the malware checking the name of each running process on the machine.
- Malicious DLL Goopdate.dll is loaded by the legitimate signed executable file GoogleUpdate.exe.
- The malware Goofy Guineapig checks the name of each running process on the machine and will not continue execution if a process containing the string 'dbg', 'debug', or 'ida' is running.
- System calls made by the malware, including calls related to process hollowing and anti-sandbox techniques.
- The malware checks for sandbox detection and various anti-analysis techniques.
- The malware implements basic anti-sandbox / anti-VM techniques.
- The disk is more than 1GB in size.
- Log entries of the malware sideloading the malicious DLL.
- Registry entries modified by the malware, including entries related to process hollowing and anti-sandbox techniques.
- The physical memory size of the machine exceeds 2GB.
- The malware reads the CPU timestamp counter and saves the result.
- Network connections made by the malware, including connections to C2 servers.
- Files created by the malware, including the malicious DLL and any other files used for process hollowing.
- The malware checks for debugger presence.
- The malware performs process hollowing on the dllhost.exe process.
- Process listings showing the malware running under a legitimate process path and name.
- Memory dumps of the malware, including memory dumps of the malicious DLL and any other malware components.
- The malware checks for the presence of other analysis tools.
- Malicious DLL Goopdate.dll is loaded by the legitimate signed executable file GoogleUpdate.exe.
- Log entries of the malware checking for physical memory size and disk size.
- Malicious files are removed from the directory containing the extracted Firefox files.
- Persistence mechanism is installed.
- Logs of the sandbox detection checks.
- Network connections to the C2 server.
- Malicious files are copied to the ProgramData directory.
- Initial directory to which files were downloaded will only contain intended files.
- GoogleUpdate.exe process is restarted from the ProgramData directory.
- Network connections to the Firefox installer server.
- Batch script logs (if any).
- Deleted files in the initial download location.
- Malicious DLL (Goopdate.dll) is loaded by a legitimate executable (GoogleUpdate.exe).
- Log files of the malicious actions.
- Malicious files are deleted from the initial download location.
- Communication over a non-standard HTTPS port (4443).
- Modified FireFox NSIS installation package.
- Process hollowing on the dllhost.exe binary.
- Persistence of malicious DLL in the system.
- Hijacking of legitimate executable's execution flow.
- Logs of rundll32.exe and url.dll execution.
- Trojanisation of a legitimate FireFox NSIS installation package.
- Malicious DLL loaded by a legitimate signed executable.
- Malicious DLL (Goopdate.dll) file on the system.
- Creation of a Windows service for persistence.
- Malicious DLL injected into dllhost.exe binary.

- Network connections established by the malware, such as connections to the C2 server.
- The malware is able to collect information from the Windows APIs.
- The malware is able to create an MD5 hash of the concatenated adapter information and host name.
- Rundll32.exe logs in the Windows Event Viewer.
- The malware is able to evade detection by sandbox and anti-virtual machine techniques.
- Logs of the malware's checks for running processes containing the string 'dbg', 'debug', or 'ida'.
- The malware is able to check the name of each running process on the machine.
- Process hollowing logs in the Windows Task Manager.
- Logs of the malware's activity in the Windows Event Viewer.
- Registry keys modified by the malware, such as the addition of a new key for the persistence mechanism.
- URL.dll logs in the Windows Event Viewer.
- The malware is able to inject content into the dllhost.exe binary.
- Files created by the malware, such as the malicious DLL and the legitimate binary.
- The malware is able to execute the legitimate binary using rundll32.exe and url.dll.
- The malware is able to prevent execution if a process containing the string 'dbg', 'debug', or 'ida' is running.
- MD5 hash logs in the Windows Event Viewer.
- Trojanisation of legitimate Firefox NSIS installation package.
- Files created by process hollowing (e.g. dllhost.exe).
- Sideload of malicious DLL by legitimate GoogleUpdate.exe.
- Persistence of malicious DLL in the system.
- Hijacking of legitimate executable to load malicious DLL.
- Malicious DLL file (e.g. Goopdate.dll) in system directory.
- Network connections to C2 server over port 4443.
- Network connections to legitimate GoogleUpdate.exe server.
- Modified Firefox NSIS installation package.
- Registry entries for Windows service and DLL loading.
- Creation of a Windows service for persistence.
- Execution of malicious DLL through rundll32.exe and url.dll.
- Communication over non-standard HTTPS port 4443.
- Log entries in Windows Event Viewer for DLL loading and execution.
- Process hollowing on dllhost.exe binary.
- Injection of content downloaded by C2 into dllhost.exe.

Attack Step 2.1

==========

Name: Masquerading: Match Legitimate Name or Location as used by the malware
Description: The malware masquerades as a legitimate program by using the following techniques: File Naming: It disguises itself as a Firefox installer and a Google updater by using filenames that mimic these legitimate applications. Installer Packaging: It is packaged within a legitimate NSIS (Nullsoft Scriptable Install System) installer, further blending in with common software installation practices. This combination of tactics aims to deceive users into believing the malware is a harmless update or installation package, increasing the likelihood of successful execution.

MITRE Technique

ID: T1036.005
Name: Masquerading: match legitimate name or location

Description: Adversaries may match or approximate the name or location of legitimate files or resources when naming/placing them. This is done for the sake of evading defenses and observation. This may be done by placing an executable in a commonly trusted directory (ex: under System32) or giving it the name of a legitimate, trusted program (ex: svchost.exe). In containerized environments, this may also be done by creating a resource in a namespace that matches the naming convention of a container pod or cluster. Alternatively, a file or container image name given may be a close approximation to legitimate programs/images or something innocuous.
More info: https://attack.mitre.org/techniques/T1036/005

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 malware implements a time-based evasion technique by checking the system time register twice, with a delay of more than 100 milliseconds between the checks. Here's how it works: First Time Check: The malware reads the current system time. Delay: The malware introduces a delay of over 100 milliseconds. This delay is crucial as it allows the malware to observe the system's behavior during this period. Second Time Check: After the delay, the malware reads the system time again. Comparison: The malware compares the two timestamps. If the difference between the timestamps is less than 100 milliseconds, it suggests that the system is running in a normal environment. However, if the difference is greater than 100 milliseconds, it indicates that the system might be in a sandbox or a virtualized environment where time manipulation is common. Decision: Based on the comparison, the malware decides whether to continue execution. If the time difference is significant, the malware will likely terminate itself, avoiding further analysis. This technique aims to detect environments where time is artificially manipulated, a common characteristic of sandboxes and virtual machines. Let me know if you have any other questions.

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, Goofy Guineapig, performs Virtualization/Sandbox Evasion: System Checks by: Retrieving System Information: It accesses the system's disk size, physical memory size, and the number of logical processors. Comparing to Expected Values: It compares these retrieved values to predefined thresholds or expected values. Conditional Execution: If any of the retrieved system information deviates from the expected values, the malware will terminate its execution. This behavior is designed to detect environments like sandboxes or virtual machines, which often have atypical system configurations. Essentially, Goofy Guineapig assumes that legitimate systems will have specific characteristics regarding their hardware resources. By checking these characteristics and aborting if they don't match, it attempts to avoid execution in controlled or monitored environments.

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 User Activity Based Checks to evade detection in sandboxes and during reverse engineering. Here's how it works: Process Monitoring: Goofy Guineapig actively monitors the processes running on the infected system. Keyword Detection: It specifically looks for process names or titles containing keywords like "dbg", "debug", or "ida". These keywords are commonly associated with debugging tools and reverse engineering environments. Execution Halt: If Goofy Guineapig detects any process with these suspicious keywords, it immediately terminates its own execution. This prevents further analysis and hinders the efforts of security researchers trying to understand its behavior. Essentially, Goofy Guineapig uses these user activity checks as a rudimentary form of self-protection, assuming that the presence of debugging tools signals an attempt to analyze or reverse engineer the malware.

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, employs UPX packing to obfuscate its code. This means its original binary is compressed and encrypted, making it harder to analyze and detect by security tools. Furthermore, it is distributed using a legitimate NSIS installer. This packaging technique helps the malware blend in with legitimate software and bypass initial security checks. Essentially, the malware hides its malicious nature by: Compressing and encrypting its core code using UPX packing. Bundling itself within a seemingly harmless NSIS installer. This dual approach makes it more difficult to identify and remove the threat.

MITRE Technique

ID: T1027.002
Name: Obfuscated files or information: software packing
Description: Adversaries may perform software packing or virtual machine software protection to conceal their code. Software packing is a method of compressing or encrypting an executable. Packing an executable changes the file signature in an attempt to avoid signature-based detection. Most decompression techniques decompress the executable code in memory. Virtual machine software protection translates an executable's original code into a special format that only a special virtual machine can run. A virtual machine is then called to run this code.
More info: https://attack.mitre.org/techniques/T1027/002

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 uses stack-based strings obfuscated with single-byte XOR or subtraction throughout its binary. Here's how this deobfuscation/decoding likely works: Stack-Based Strings: The malware stores strings not directly in the code but on the program's stack. This makes them harder to immediately analyze as they are not readily visible in the code's text. XOR or Subtraction Obfuscation: Each character in these stack-based strings is encrypted using either: XOR (exclusive OR): A bitwise operation where each character's ASCII value is XORed with a specific key. To decode, the same key is used in reverse. Subtraction: Each character's ASCII value is subtracted by a constant value. To decode, the same subtraction is applied in reverse. Decoding at Runtime: When the malware needs to use a string, it retrieves it from the stack and applies the XOR or subtraction operation to reveal the original characters. This decoding happens dynamically during program execution. Why this is effective: Hiding: The obfuscation makes it harder for static analysis tools to directly identify the strings and their purpose. Dynamic Nature: The decoding happens at runtime, making it more difficult to predict and analyze the strings in advance. Let me know if you'd like more details on XOR or subtraction encryption techniques!

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 a technique called process hollowing. Here's how it works: Target Selection: Goofy Guineapig targets the legitimate dllhost.exe process. Process Hollowing: The malware injects its own malicious code into an existing instance of dllhost.exe. This malicious code then takes over the control flow of the dllhost.exe process, effectively replacing its original functionality. Hidden Process: Because the malicious code is running within the context of dllhost.exe, it inherits the same security privileges and visibility. Since dllhost.exe is a legitimate system process, the hidden malicious process is less likely to be detected by security tools. Evasion: By hiding its malicious activity within a legitimate process, Goofy Guineapig attempts to evade detection and analysis. Let me know if you have any other questions about malware techniques!

MITRE Technique

ID: T1564.003
Name: Hide artifacts: hidden window
Description: Adversaries may use hidden windows to conceal malicious activity from the plain sight of users. In some cases, windows that would typically be displayed when an application carries out an operation can be hidden. This may be utilized by system administrators to avoid disrupting user work environments when carrying out administrative tasks.
More info: https://attack.mitre.org/techniques/T1564/003

Indicators

No indicators found.

Attack Step 2.8

==========
Name: Indicator Removal on Host: File Deletion as used by the malware
Description: The malware, Goofy Guineapig, performs the "Indicator Removal on Host: File Deletion" action by following these steps: Initial Execution: The malware first runs from the directory where it was downloaded. File Relocation: It then copies itself and any other necessary files to a legitimate-looking directory on the system. This directory choice aims to blend in with normal system files and avoid detection. Deletion from Download Location: Finally, Goofy Guineapig deletes the original files from the initial download location. This removes the direct trace of the malware's arrival and makes it harder to track its initial infection point. This tactic helps the malware evade detection by security tools that might focus on monitoring newly downloaded files or suspicious activity in temporary directories.

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 Goofy Guineapig hijacks execution flow using DLL side-loading: 1. The Setup: Malicious Loader: Goofy Guineapig's operation begins with a loader component. This loader is responsible for several tasks: Dropping Files: It downloads and installs a legitimate executable (e.g., GoogleUpdate.exe) and a malicious DLL (the "Goofy Guineapig" DLL itself). Creating a Service: It sets up a Windows service to ensure persistence. This service will likely load the malicious DLL on system startup. The Trojanized Package: In some cases, Goofy Guineapig might disguise itself within a legitimate software package, like a Firefox installer. This helps it blend in and avoid detection. 2. The Side-Loading: Legitimate Executable: The legitimate executable (e.g., GoogleUpdate.exe) is designed to load and execute other DLLs. This is a common mechanism for software components to extend functionality. The Malicious DLL: The malicious DLL is strategically placed in a location where the legitimate executable will search for additional DLLs to load. Execution Hijack: When the legitimate executable runs, it follows its normal process of loading required DLLs. Because the malicious DLL is in the expected location, it gets loaded alongside the legitimate code. 3. The Payload: Code Injection: Once loaded, the malicious DLL injects its own code into the legitimate process's memory space. Command and Control (C2): The injected code establishes a connection to a remote server (the C2) to receive further instructions. Malicious Actions: The C2 server can then command the compromised process to perform various malicious actions, such as: Data Exfiltration: Steal sensitive information from the infected machine. Lateral Movement: Spread to other systems on the network. Ransomware: Encrypt files and demand payment for decryption. Key Points: Persistence: The Windows service ensures that Goofy Guineapig continues to run even after the user logs off or restarts their computer. Stealth: By leveraging a legitimate executable, Goofy Guineapig attempts to avoid detection by security software. Flexibility: The C2 server allows the malware operators to dynamically change the behavior of the infection. 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 performs Process Injection via Process Hollowing on the dllhost.exe binary. Here's a breakdown of how it's done: Target Selection: The malware identifies dllhost.exe as its target process. This process is often used by legitimate applications, making it a suitable candidate for hiding

malicious code. Process Hollowing: Memory Allocation: The malware allocates a new memory region within the dllhost.exe process's address space, large enough to hold the malicious code it intends to inject. Code Replacement: The malware then overwrites the original code of dllhost.exe within its allocated memory region with its own malicious payload. This effectively "hollows out" the original process. Redirection: The malware modifies the entry point of dllhost.exe to point to the beginning of its injected malicious code. Payload Execution: When dllhost.exe is executed, it now runs the malicious code injected by the malware. This allows the malware to operate within the context of a legitimate process, making it harder to detect. Downloaded Content: The malicious code injected into dllhost.exe is likely to download additional components or instructions from the C2 (Command and Control) server. This allows the malware to adapt and evolve its behavior over time. Let me know if you have any other questions.

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: Goofy Guineapig leverages the legitimate rundll32.exe and url.dll combination to achieve persistence. Here's how it works: Leveraging Rundll32: rundll32.exe is a Windows utility used to execute functions within DLL files. Using url.dll: The malware instructs rundll32.exe to load and execute functions from the url.dll file. Executing the Legitimate Binary: Within url.dll, the malware has likely embedded code that specifies the path to a legitimate Windows executable (e.g., a system service or a common application). Loading the Malicious DLL: When the legitimate binary is executed, Goofy Guineapig's malicious DLL is injected into its memory space. This happens through a technique like process hollowing, where the legitimate binary's contents are replaced with the malicious code. Persistence Achieved: Because the legitimate binary is often a system component or runs frequently, the injected malicious DLL remains loaded in memory, allowing Goofy Guineapig to maintain persistence on the infected system. 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 the necessary resources to store the collected information.
- Resources: Storage space to store the collected information, bandwidth to send the collected information.
- The malware has the necessary permissions to collect system information.
- The necessary Windows APIs are available.
- The malware is loaded.
- The malware has a valid HTTPS connection.
- Connectivity: HTTPS connection.
- Environment: Windows operating system.
- The COM interface to access WMI information is available.

Post-Conditions:

- Malware can execute plugins by process hollowing dllhost.exe.
- Malware can be tasked to collect information about the infected machine or run additional plugins.
- HTTPS connections to C2 servers.
- Loaded DLL (Goopdate.dll) in the process list.
- Malware (Goofy Guineapig) process in the process list.
- Files created by process hollowing (e.g., dllhost.exe).
- Malware concatenates adapter information and host name, takes an MD5 hash of the result.
- Malware fails to collect information, it replaces it with the string '(none)'.
- Malware sends collected information in an obfuscated 'Authorization' string in the HTTP header.
- Network connections to C2 servers.
- Malware (Goofy Guineapig) collects system information about the infected machine.
- Malware uses HTTPS for C2 communications.
- HTTP requests with obfuscated 'Authorization' string in the HTTP headers.
- Malicious DLL (Goopdate.dll) is loaded by a legitimate executable (GoogleUpdate.exe).
- Logs of system information collected by the malware.
- Files created by the malware (e.g., plugins, logs).

Attack Step 3.1

==========

Name: System Information Discovery as used by the malware
Description: The malware, Goofy Guineapig, performs System Information Discovery by embedding information about the infected machine within each communication packet sent to its Command and Control (C2) server. Specifically, it obfuscates this system information and includes it as part of an "Authorization" string within the HTTP header of each outgoing packet. Let me know if you'd like more details on any specific aspect of this process!

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:

- Connectivity: Internet connection.
- The malware has the necessary permissions to set internet options.
- Tools: ObtainUserAgentString Windows API.
- Environment: Windows operating system.
- Resources: Valid HTTPS connection, valid certificate for HTTPS communication, necessary permissions to set internet options, necessary permissions to retrieve the user agent string, necessary permissions to send HTTPS requests, necessary permissions to handle HTTPS certificate-related errors, necessary permissions to set security flags.
- The malware has access to the ObtainUserAgentString Windows API.
- The malware must have the necessary permissions to set security flags.
- The malware has access to the internet.
- The malware has the necessary permissions to set security flags.
- The malware has the necessary permissions to retrieve the user agent string.
- HTTPS is available.
- The malware must have the necessary permissions to handle certificate-related errors.
- The malware has already completed request 0x15.
- Connectivity: None explicitly stated, but likely requires internet connectivity to communicate with the C2 server.
- Availability of the malware's communication protocol (KCP) to use for UDP communication.
- The malware has already collected information about the infected machine.
- The malware has already enumerated the logon sessions on the infected machine.
- The disk is more than 1GB in size.
- Environment: A Windows machine with a physical memory size exceeding 2GB and a disk size exceeding 1GB.
- The malware has already implemented sandbox detection and anti-analysis techniques.
- The physical memory size of the machine exceeds 2GB.
- Availability of the infected machine's information to be sent in the C2 packet.
- Resources: None explicitly stated, but likely requires system resources to perform the fallback channel communication.
- The malware has already determined the communication type to be used (HTTP(S) or UDP).
- The embedded configuration string contains UDP rather than HTTP (S).
- Tools: None explicitly stated, but likely requires a debugger or analysis tools to detect sandboxing or virtualization.
- The embedded configuration string contains the string '\x00'.
- The embedded configuration string contains the string '12' or '5' or '1'.
- The embedded configuration string contains the string 'http' or 'HTTP'.
- The embedded configuration string contains the string 'HTTPS://static.tcplog.com:4443' or 'HTTPS://static.tcplog.com:4443'.
- Tools: None explicitly stated, but likely requires network analysis tools to detect non-standard port usage.
- Availability of the malware's binary and its embedded configuration string.
- Resources: Access to the malware's configuration string and the ability to analyze its communication protocols.
- The malware maintains persistence using a Windows service.
- Availability of the TCP log server at 'static.tcplog.com:4443'.

- The embedded configuration string is searched for in order to determine the communication type that should be utilised.
- Connectivity: Network connectivity to communicate over the non-standard port 4443.
- The malware is loaded by a legitimate signed executable.
- Environment: Windows operating system.
- The embedded configuration string contains UDP rather than HTTP (S).

Post-Conditions:

- Malware uses a hardcoded configuration string for C2 communications.
- Malware uses a hardcoded key for RC4 encryption.
- Malicious DLL (Goopdate.dll) is loaded by a legitimate executable (GoogleUpdate.exe).
- Malware communicates over non-standard HTTPS port 4443.
- HTTPS communication logs in the system event logs.
- NSIS installer with a trojanized Firefox installer.
- Malware implements anti-sandbox/anti-VM techniques.
- Loaded DLL (Goopdate.dll) in the process list.
- Malware checks for specific certificate-related errors in HTTPS requests.
- Malware binary with a single byte XOR with the key 0x59.
- Certificate-related error logs in the system event logs.
- UPX packed loader in the malware binary.
- Modified user agent string header in the HTTP requests.
- Malware sends system information about the infected machine in each C2 packet.
- Malware sets internet options to circumvent certificate-related errors.
- Network connections to non-standard HTTPS port 4443.
- Malware uses HTTPS for C2 communications.
- Network connections using UDP and KCP protocol.
- System information sent in each C2 packet.
- Infected machine sends information about itself in each C2 packet.
- Malware communicates over non-standard HTTPS port 4443.
- Logs of HTTP/S communications over port 4443.
- Files created by UPX packed NSIS installer.
- Malware collects a selection of information about the infected machine.
- Malware uses KCP protocol for UDP communications.
- Malware implements basic anti-sandbox / anti-VM techniques.
- Malware collects client address before concatenating gathered information.
- Network connections to UDP ports used by KCP protocol.
- Malicious DLL Goopdate.dll loaded by GoogleUpdate.exe.
- Malware uses HTTPS for C2 communications.
- Files created by Goofy Guineapig malware.
- Network connections to static.tcplog.com:4443.
- Registry entries created by Goopdate.dll.
- Malware sends obfuscated 'Authorization' string in HTTP header.
- Files created by Firefox installer.
- Logs of UDP communications using KCP protocol.
- Malware checks physical memory size of the machine exceeds 2GB.
- Loading of a malicious DLL (Goopdate.dll) by a legitimate executable (GoogleUpdate.exe).
- Implementation of sandbox detection and anti-analysis techniques.
- Presence of logs related to certificate-related errors and their circumvention.
- Creation of a UPX packed NSIS installer.
- Use of UDP and KCP protocol for communication.
- Persistence of a malicious Windows service.

- Trojanisation of a Firefox installer.
- Hardcoding of a configuration string in the binary.
- Presence of logs related to the ObtainUserAgentString Windows API.
- Communication over non-standard HTTPS port 4443.
- Modification of internet options to circumvent certificate-related errors.
- Presence of a dynamically retrieved HTTP user agent string header.

Attack Step 4.1

==========
Name: Application Layer Protocol: Web Protocols as used by the malware
Description: The malware utilizes the HTTPS protocol for its communication with the 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 the C2 server. Let me know if you have any other actions you'd like me to explain!

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

- 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. Here's how it works: Configuration-Driven: The specific communication method is determined by an embedded configuration string within the malware. Available Options: Goofy Guineapig supports three communication protocols: UDP: A connectionless protocol suitable for sending small packets of data. KCP: A fast and reliable protocol designed for low-latency and high-throughput communication, often used in gaming and other real-time applications. Direct Socket Communications: A more traditional method of establishing a direct connection between the malware and its Command and Control (C2) server. Dynamic Selection: Based on the configuration string, Goofy Guineapig will select the most appropriate communication method for establishing a connection with its C2 server. Let me know if you have any other questions about Goofy Guineapig's functionality.

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 action "Non-Standard Port" is performed by the malware Goofy Guineapig in the following way: Communication Protocol: Goofy Guineapig utilizes the HTTPS protocol for its communication with the Command and Control (C2) server. Port Number: Instead of using the standard HTTPS port 443, Goofy Guineapig deliberately employs the non-standard port 4443. This choice of a non-standard port serves several purposes for the malware: Evasion: By deviating from the common HTTPS port, Goofy Guineapig attempts to bypass security measures that might be specifically looking for traffic on port 443. Obfuscation: Using a less common port makes it harder for analysts to identify and track the malware's communication. Stealth: The non-standard port can help Goofy Guineapig operate more discreetly, as it is less likely to raise suspicion on network traffic analysis. Let me know if you have any other questions about Goofy Guineapig or its tactics.

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.