Enhanced Attack Report

Goofy Guineapig

Generated on 2025-03-03

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 on the system.
- The system has a physical memory size exceeding 2GB.
- The system has a disk size exceeding 1GB.
- The system has a debugger running.
- The system has other analysis tools running.
- The system has a sandbox detection mechanism implemented.
- The system has a virtual machine (VM) detection mechanism implemented.
- The Goopdate.dll DLL is present on the system.
- The Goopdate.dll DLL is UPX packed.
- The Goopdate.dll DLL is bundled in a NSIS installer.
- The NSIS installer is a trojanised Firefox installer.
- The system has the necessary permissions to create or modify a Windows service.
- Windows operating system
- Physical memory size exceeding 2GB
- Disk size exceeding 1GB
- Debugger
- Analysis tools
- Sandbox detection mechanism
- Virtual machine (VM) detection mechanism
- Goopdate.dll DLL
- NSIS installer
- Firefox installer
- UPX packing tool
- NSIS installer creation tool
- Windows service creation and modification permissions
- System administrator privileges
- Access to the system's registry and file system

Post-Conditions:

- Persistence mechanism is established through a Windows service.
- Malicious files are present in the ProgramData directory.
- Initial directory contains only intended files, with malicious files hidden in ProgramData.
- Malware can continue to execute without a persistence mechanism.
- Basic anti-sandbox/anti-VM techniques are implemented.
- Physical memory size and disk size checks are performed.
- Malware can communicate over HTTP, HTTPS, and UDP using KCP protocol.
- Malware can collect adapter information and host name using Windows APIs.
- MD5 hash of concatenated adapter information and host name is taken.
- Malware can determine communication type based on embedded configuration string.
- Windows service created in the Services control panel.
- Malicious files in the ProgramData directory.
- Log entries in the Windows Event Viewer.
- Network connections to static.tcplog.com:4443.
- Files created in the initial directory.
- Registry entries modified to enable persistence.

- Log entries in the Windows Application Event Log.
- Files created in the C:\ProgramData\GoogleUpdate directory.
- Network connections to GoogleUpdate.exe.
- Log entries in the Windows System Event Log.

Attack Step 1.1

==========
Name: Create or Modify System Process: Windows Service as used by Goofy Guineapig
Description: Goofy Guineapig achieves persistence by installing itself as a Windows service. Here's how this is likely performed: Service Creation: The malware uses functions within the Windows API to create a new service. This service will have a name (possibly disguised as a legitimate service), a display name, and a start type (likely set to "Automatic" for automatic startup). Service Binary: The malware's own executable file is registered as the service binary. This means when the service starts, it will execute Goofy Guineapig's code. Configuration: The service configuration, including startup parameters and other settings, is embedded within the malware's binary. This configuration might specify the service's behavior, communication protocols, and other crucial details. Service Registration: The malware registers the newly created service with the Windows service manager. This makes the service a permanent part of the system, ensuring it starts automatically on system boot. Key Points: Stealth: By masquerading as a legitimate Windows service, Goofy Guineapig can avoid detection by security software that might focus on suspicious processes rather than services. Automatic Startup: The "Auto start" setting ensures that the malware automatically starts every time the system boots, guaranteeing its persistence. Embedded Configuration: Storing the configuration within the binary itself makes it harder for analysts to modify or understand the malware's behavior without directly analyzing the code. 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 Windows operating system is installed.
- The malware has access to the system.
- The malware has the necessary permissions to create or modify system processes.
- The malware has the necessary information to masquerade as a legitimate process.
- The malware has the necessary tools to create a legitimate-looking process.
- 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.
- The system has a process with a legitimate name or location that can be matched.
- The system has a process with a legitimate window text that can be matched.
- Windows operating system
- Malware with masquerading capabilities
- System with physical memory size exceeding 2GB
- System with disk size exceeding 1GB
- System not running in a sandbox or virtual machine environment
- Access to system processes and permissions to create or modify them
- Tools to create a legitimate-looking process
- Information about legitimate processes and their locations
- Access to system APIs to collect information about the system and its processes
- A legitimate process or location to match the malware's masquerading capabilities
- The system time is registered.
- The time register is checked twice.
- A delay of more than 100 milliseconds has not elapsed.
- A system with a time register.
- A system with a delay of more than 100 milliseconds.
- The ability to check the system time.
- The ability to check the time register twice.
- A system with a logical processor count exceeding 2.
- The system has a disk.
- The system has physical memory.
- The system has a logical processor.
- The system's disk size exceeds 1GB.
- The system's physical memory size exceeds 2GB.
- The system has more than 2 logical processors.
- A Windows environment.
- Access to the system's disk size, physical memory size, and number of logical processors.
- The ability to check the system's disk size, physical memory size, and number of logical processors.
- The system's disk size, physical memory size, and number of logical processors must be available for checking.
- The system must be running a 64-bit operating system (inferred from the requirement for physical memory size exceeding 2GB).
- The system must have a 64-bit processor (inferred from the requirement for physical memory size exceeding 2GB).
- The system is running.
- The system has a disk.

- The system has physical memory.
- The system has logical processors.
- The system has a network connection.
- The system has a user logged in.
- The system has processes running.
- The system has system time set.
- Environment: Windows operating system.
- Tools: None explicitly mentioned, but the system must have the necessary APIs and libraries to interact with the system.
- Resources:
- The malware Goofy Guineapig is present in the system.
- The system has a legitimate NSIS installer.
- The system has a UPX packing tool.
- The system has a legitimate executable file.
- The system has a physical memory size exceeding 2GB.
- The system has a disk size exceeding 1GB.
- The system has a debugger or analysis tools running.
- The system has a sandbox or virtual machine environment.
- Environment: Windows system.
- Tools: UPX packing tool, NSIS installer, debugger or analysis tools.
- Resources: Physical memory size exceeding 2GB, disk size exceeding 1GB.
- Availability of legitimate NSIS installer and executable file.
- Presence of Goofy Guineapig malware.
- Access to the system's physical memory and disk.
- The Goofy Guineapig malware is present in the environment.
- The malware contains stack-based strings obfuscated with single byte XOR or subtraction.
- The obfuscated strings are stored in the malware's binary.
- The environment has a mechanism to access and analyze the malware's binary.
- The environment has a mechanism to compare and analyze the obfuscated strings.
- The environment has a mechanism to determine the type of obfuscation used (single byte XOR or subtraction).
- The environment has a mechanism to identify the key used for XOR obfuscation.
- The environment has a mechanism to decode the obfuscated strings.
- Access to the Goofy Guineapig malware's binary.
- A tool or mechanism to analyze and compare the obfuscated strings.
- A tool or mechanism to determine the type of obfuscation used (single byte XOR or subtraction).
- A tool or mechanism to identify the key used for XOR obfuscation.
- A tool or mechanism to decode the obfuscated strings.
- A stable and controlled environment to perform the analysis.
- A sufficient amount of physical memory (more than 2GB) to run the analysis.
- A sufficient amount of disk space (more than 1GB) to run the analysis.
- A tool or mechanism to detect and prevent sandbox or virtual machine detection.
- A tool or mechanism to detect and prevent debugger detection.
- The malware is loaded on the infected machine.
- The malware has the functionality to perform process hollowing on dllhost.exe.
- The process hollowing on dllhost.exe is performed successfully.
- The physical memory size of the machine exceeds 2GB.
- The disk is more than 1GB in size.
- A debugger is not running on the machine.
- The machine is not in a sandbox or virtual machine environment.
- The malware has the capability to create a named pipe.
- The malware has the capability to hash the computer name twice using MD5.
- The malware has the capability to store the handle of the spawned process in a global structure.

- Windows environment.
- Availability of dllhost.exe.
- Availability of a debugger (to be excluded).
- Availability of a sandbox or virtual machine detection tool (to be excluded).
- Availability of a tool to hash strings using MD5.
- Availability of a tool to create named pipes.
- Availability of a tool to store handles in a global structure.
- Availability of a tool to perform process hollowing.
- Availability of a tool to check physical memory size and disk size.
- Availability of a tool to check for sandbox or virtual machine environments.
- The malware is initially run in the location to which it is downloaded.
- The files are moved to a legitimate looking directory.
- The files are deleted from the initial download location.
- The persistence mechanism has been installed.
- The malicious files are copied to the ProgramData directory.
- The initial directory to which the files were downloaded contains only the files the recipient likely intended to download.
- The malicious files are present in the ProgramData directory.
- The ProgramData directory is a hidden directory by default.
- The batch script is executed.
- The physical memory size of the machine exceeds 2GB.
- The disk is more than 1GB in size.
- Windows environment.
- GoogleUpdate.exe and Goopdate.dll files.
- Batch script.
- ProgramData directory.
- Hidden directory capability.
- File deletion capability.
- File copying capability.
- Persistence mechanism.
- Debugger or analysis tools are not running.
- Sandbox detection capabilities are not present.
- 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 malicious DLL is sideloaded by the legitimate executable.
- The legitimate executable is dropped alongside the malicious DLL.
- The malicious DLL is bundled in a UPX packed NSIS installer.
- The NSIS installer is a trojanised Firefox installer.
- The legitimate executable is GoogleUpdate.exe.
- The malicious DLL is Goopdate.dll.
- The environment is a Windows system.
- Windows environment.
- NSIS installer (trojanised Firefox installer).
- UPX packer.
- GoogleUpdate.exe (legitimate signed executable).
- Goopdate.dll (malicious DLL).
- DLL side-loading capability.
- Ability to install legitimate executable alongside malicious DLL.
- Ability to sideload malicious DLL by legitimate executable.
- Ability to bundle malicious DLL in UPX packed NSIS installer.
- Ability to trojanise Firefox installer.
- The system has a legitimate dllhost.exe binary.

- The system has a legitimate rundll32.exe binary.
- The system has a legitimate url.dll binary.
- The system has a legitimate GoogleUpdate.exe binary.
- The system has a legitimate FireFox NSIS installation package.
- The system has a legitimate FireFox executable.
- The system has a legitimate GoogleUpdate.exe executable.
- The system has a legitimate url.dll DLL.
- The system has a legitimate FireFox NSIS installation package executable.
- The system has a legitimate FireFox executable with a legitimate FireFox NSIS installation package.
- The system has a legitimate GoogleUpdate.exe executable with a legitimate url.dll DLL.
- The system has a legitimate FireFox NSIS installation package executable with a legitimate FireFox executable.
- The system has a legitimate GoogleUpdate.exe executable with a legitimate FireFox NSIS installation package executable.
- The system has a legitimate FireFox executable with a legitimate GoogleUpdate.exe executable.
- The system has a legitimate FireFox NSIS installation package executable with a legitimate GoogleUpdate.exe executable.
- The system has a legitimate GoogleUpdate.exe executable with a legitimate FireFox NSIS installation package executable.
- The system has a legitimate FireFox executable with a legitimate FireFox NSIS installation package executable.
- The system has a legitimate GoogleUpdate.exe executable with a legitimate FireFox executable.
- The system has a legitimate FireFox NSIS installation package executable with a legitimate FireFox executable.
- The system has a legitimate GoogleUpdate.exe executable with a legitimate FireFox NSIS installation package executable.
- Windows operating system.
- dllhost.exe binary.
- rundll32.exe binary.
- url.dll binary.
- GoogleUpdate.exe binary.
- FireFox NSIS installation package.
- FireFox executable.
- GoogleUpdate.exe executable.
- url.dll DLL.
- FireFox NSIS installation package executable.
- Named pipe creation capability.
- Process hollowing capability.
- Process injection capability.
- CPU timestamp counter capability.
- MD5 hashing capability.
- Computer name access capability.
- Global structure access capability.
- Process termination capability.
- Time-based evasion capability.
- Malicious DLL sideloading capability.
- Legitimate process masquerading capability.
- Legitimate FireFox NSIS installation package executable with a legitimate FireFox executable.
- Legitimate GoogleUpdate.exe executable with a legitimate url.dll DLL.
- Legitimate FireFox NSIS installation package executable with a legitimate GoogleUpdate.exe executable.
- Legitimate FireFox executable with a legitimate GoogleUpdate.exe executable.

- Legitimate GoogleUpdate.exe executable with a legitimate FireFox NSIS installation package executable.
- Legitimate FireFox NSIS installation package executable with a legitimate FireFox executable.
- Legitimate GoogleUpdate.exe executable with a legitimate FireFox executable.
- Legitimate FireFox executable with a legitimate FireFox NSIS installation package executable.
- Legitimate GoogleUpdate.exe executable with a legitimate FireFox NSIS installation package executable.
- A Windows operating system is present.
- The GoogleUpdate.exe binary is available.
- The url.dll binary is available.
- The rundll32.exe binary is available.
- The malicious DLL is present.
- The legitimate Firefox NSIS installation package is present.
- The malicious DLL is sideloaded by the legitimate GoogleUpdate.exe executable.
- The initial Goopdate.dll execution has occurred.
- A hidden process has been created.
- A batch file named tmp.bat is present.
- Windows operating system
- GoogleUpdate.exe binary
- url.dll binary
- rundll32.exe binary
- Malicious DLL
- Legitimate Firefox NSIS installation package
- Batch file editor (e.g., Notepad)
- Command line interface (CLI)
- Hidden process creation capability
- Ability to execute batch files
- Ability to sideload DLLs
- Signed GoogleUpdate.exe executable

Post-Conditions:

- Persistence as a Windows service.
- Malicious files present in the ProgramData directory.
- Malware masquerading as a FireFox installer and a Google updater.
- Malware checks for running processes containing 'dbg', 'debug', or 'ida' strings.
- Malware checks for debugger presence.
- Malware implements anti-sandbox/anti-VM techniques.
- Malware collects host information using Windows APIs.
- Malware collects adapter information and host name.
- Malware takes an MD5 hash of the concatenated adapter information and host name.
- Windows service logs.
- Malicious files in the ProgramData directory.
- FireFox installer and Google updater executables.
- Process logs showing 'dbg', 'debug', or 'ida' string checks.
- Debugger presence detection logs.
- Anti-sandbox/anti-VM technique logs.
- Host information collection logs.
- Adapter information and host name logs.
- MD5 hash logs.
- Network connections to command and control servers (HTTPS).
- HTTP and KCP network connections.

- Logs of system process creation or modification.
- Logs of system process deletion.
- Logs of system process execution.
- Malicious files are present in the ProgramData directory.
- The initial directory to which the files were downloaded will only contain the files the recipient likely intended to download.
- The malicious files will be deleted after execution.
- The infected machine's system time checks will be altered.
- The infected machine's running processes will be altered.
- The infected machine's properties will be altered.
- The malware will exit if any sandbox detection checks fail.
- The malware will not continue execution if more time has elapsed.
- The malware will not continue execution if it is being reverse engineered or debugged.
- The malware will not continue execution if the number of logical processors exceeds 2.
- Logs of system time checks.
- Logs of running processes checks.
- Logs of system properties checks.
- Logs of sandbox detection checks.
- Files in the ProgramData directory.
- Files in the initial download directory.
- Network connections to command and control servers.
- Files created by the UPX packed NSIS installer.
- Files created by the Goopdate.dll DLL.
- Registry entries created by the malware for persistence.
- Logs of the malware's execution.
- Files created by the shellcode.
- Logs of the malware's defence evasion techniques.
- Malicious files are present in the ProgramData directory.
- The initial directory to which the files were downloaded will only contain the files the recipient likely intended to download.
- The malware will exit if any of the sandbox detection checks fail.
- The malware will not continue execution if the physical memory size of the machine is less than 2GB or the disk is less than 1GB in size.
- The malware will not continue execution if the number of logical processors is less than or equal to 2.
- The malware will not continue execution if it detects processes running on a system that indicate it is being reverse engineered or debugged.
- Logs of the malware's execution and sandbox detection checks.
- Files in the ProgramData directory.
- Network connections to the C2 server.
- Files in the initial directory to which the files were downloaded.
- Registry entries or other system configuration changes made by the malware.
- Logs of the disk size, physical memory size, and number of logical processors checks.
- Logs of the process checks for reverse engineering or debugging.
- Files created by the malware, such as the batch script.
- Network connections to the RDP server if the protocol type is RDP.
- Malicious files are present in the ProgramData directory.
- The initial directory to which the files were downloaded will only contain the files the recipient likely intended to download.
- The malicious files will be deleted after the final command in the batch script is executed.
- The infected machine's disk size, physical memory size, and number of logical processors are altered.
- The infected machine's system time is altered.

- The infected machine's processes are altered.
- The infected machine's properties are altered.
- The infected machine's logon sessions are enumerated.
- The infected machine's username and client protocol type (RDP or LOCAL) associated with the enumerated sessions are queried.
- Logs of the enumerated logon sessions.
- Files in the ProgramData directory.
- Altered system time logs.
- Altered process logs.
- Altered system property logs.
- Network connections for the enumerated logon sessions.
- Files created by the batch script.
- Logs of the system checks performed by the malware.
- Logs of the time-based evasion checks performed by the malware.
- Logs of the virtualization/sandbox evasion checks performed by the malware.
- Malicious files are present in the ProgramData directory.
- Malware is loaded by a legitimate executable file.
- Malware checks for running processes containing specific strings.
- Malware will not continue execution if a debugger or IDA is running.
- Malware starts a service for persistence.
- Malware is UPX packed and packaged in a legitimate NSIS installer.
- Malware contains stack-based strings which are obfuscated with single byte XOR or subtraction.
- Malware is obfuscated with software packing.
- Malware is bundled in a trojanised Firefox installer.
- Malware has a flawed secondary check.
- Files in the ProgramData directory.
- Logs of malware execution.
- Network connections to C2 servers.
- Files in the %APPDATA% directory.
- Files in the %TEMP% directory.
- Registry keys for malware persistence.
- Logs of process creation and termination.
- Files in the %PROGRAMFILES% directory.
- Files in the %WINDIR% directory.
- Network connections to the NSIS installer.
- Malicious files are present in the ProgramData directory.
- The initial directory to which the files were downloaded will only contain the files the recipient likely intended to download.
- The malware communicates over non-standard HTTPS port 4443.
- The malware uses fallback channels such as UDP and KCP protocol.
- The malware implements defence evasion techniques.
- 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 checks for physical memory size and disk size to detect sandbox or virtual machine.
- Files in the ProgramData directory.
- Logs of network connections over non-standard HTTPS port 4443.
- Configuration string hardcoded in the binary.
- Embedded configuration string in the binary.
- Files in the initial directory to which the files were downloaded.
- Logs of UDP and KCP protocol communications.
- UPX packed and NSIS installer files.
- Stack-based strings obfuscated with single byte XOR or subtraction in the binary.

- Logs of physical memory size and disk size checks.
- Network connections to static.tcplog.com.
- Persistence of malware on the system
- Creation of a hidden directory in ProgramData
- Presence of malicious files in the ProgramData directory
- Modification of the registry to auto-start the malware
- Creation of a named pipe with the computer name MD5 hashed twice
- Termination of previously spawned processes
- Implementation of anti-sandbox and anti-VM techniques
- Potential for process hollowing on dllhost.exe
- Logs of registry modifications
- Presence of malicious files in the ProgramData directory
- Creation of a named pipe with the computer name MD5 hashed twice
- Network connections to command and control servers over HTTPS
- Logs of process creation and termination
- Presence of hidden directories and files
- Registry entries for auto-starting the malware
- Logs of anti-sandbox and anti-VM checks
- Files created by process hollowing on dllhost.exe
- Logs of process hollowing on dllhost.exe
- Malicious files are present in the ProgramData directory.
- Initial download directory only contains intended files.
- Persistence mechanism is installed.
- Malicious files are copied to the ProgramData directory.
- Files are deleted from the original download location.
- GoogleUpdate.exe process is restarted from the ProgramData directory.
- Batch script deletes itself.
- Physical memory size check is implemented.
- Disk size check is implemented.
- Sandbox detection techniques are implemented.
- Logs of persistence mechanism installation.
- Files in the ProgramData directory.
- Deleted files in the original download location.
- Network connections for command and control communications.
- Logs of command and control communications.
- Files in the GoogleUpdate directory.
- Batch script execution logs.
- Logs of physical memory size check.
- Logs of disk size check.
- Logs of sandbox detection techniques.
- Modified registry entries for persistence mechanism.
- Network connections for HTTPS communications.
- Files in the C:\windows\system32 directory.
- Logs of rundll32.exe and url.dll execution.
- Modified system files for persistence mechanism.
- Persistence of malicious DLL (Goopdate.dll) in the system.
- Creation of a Windows service for persistence.
- Malicious files present in the ProgramData directory.
- Hijacking of legitimate executable (GoogleUpdate.exe) for malicious purposes.
- Loading of malicious DLL by legitimate executable (GoogleUpdate.exe).
- Process hollowing on dllhost.exe binary.
- Injection of content downloaded by C2 into dllhost.exe.
- Execution of malicious DLL through rundll32.exe and url.dll.

- Malicious communications over non-standard HTTPS port 4443.
- Malicious files deleted from initial download directory.
- Presence of malicious DLL (Goopdate.dll) in the system.
- Windows service created for persistence.
- Malicious files in the ProgramData directory.
- Logs of legitimate executable (GoogleUpdate.exe) hijacking.
- Network connections to C2 over non-standard HTTPS port 4443.
- Files created by process hollowing on dllhost.exe binary.
- Logs of process hollowing on dllhost.exe binary.
- Files created by injection of content downloaded by C2 into dllhost.exe.
- Logs of injection of content downloaded by C2 into dllhost.exe.
- Registry entries created for Windows service.
- Presence of UPX packed NSIS installer in the system.
- Logs of execution of malicious DLL through rundll32.exe and url.dll.
- Network connections to C2 from rundll32.exe and url.dll.
- Files created by malicious communications over non-standard HTTPS port 4443.
- Logs of malicious communications over non-standard HTTPS port 4443.
- Persistence of malicious files in the ProgramData directory.
- Malicious DLL loaded by legitimate GoogleUpdate.exe.
- Process hollowing on dllhost.exe binary.
- Creation of a named pipe with a computer name MD5 hashed twice.
- Malicious files deleted from the initial download directory.
- Malicious process masquerading as legitimate processes.
- Malicious DLL sideloaded by legitimate GoogleUpdate.exe.
- Malicious files dropped alongside legitimate FireFox files.
- Logs of process hollowing on dllhost.exe binary.
- Network connections to C2 server for command and control communications.
- Files in the ProgramData directory.
- Named pipe with a computer name MD5 hashed twice.
- Logs of malicious DLL loaded by legitimate GoogleUpdate.exe.
- Logs of process masquerading as legitimate processes.
- Logs of malicious DLL sideloaded by legitimate GoogleUpdate.exe.
- Files in the initial download directory after deletion.
- Logs of malicious process creation and termination.
- System binary proxy execution logs.
- Persistence of malicious DLL in the system through the use of rundll32.exe and url.dll.
- Creation of a hidden process that calls a batch file.
- Modification of the GoogleUpdate.exe process to load the malicious DLL.
- Deletion of malicious files from the original file path.
- Presence of malicious files in the ProgramData directory.
- Creation of a batch file that contains malicious commands.
- Error in the choice command due to the format string '%d' not being replaced.
- Execution of malicious shellcode.
- Creation of a hollowed process through process hollowing.
- Injection of content downloaded by the C2 into the dllhost.exe binary.
- Logs of rundll32.exe and url.dll execution.
- Presence of the malicious DLL in the system.
- Batch file created in the ProgramData directory.
- Hidden process created in the system.
- Network connections to the C2 server.
- Files created in the ProgramData directory.
- Modification of the GoogleUpdate.exe process.
- Logs of process hollowing and injection.

- Presence of the malicious shellcode in the system.
- Logs of the choice command error.
- Files deleted from the original file path.
- Presence of the malicious files in the ProgramData directory.

Attack Step 2.1

==========
Name: Masquerading: Match Legitimate Name or Location as used by Goofy Guineapig
Description: Goofy Guineapig masquerades as a legitimate program to evade detection. How it's performed: File Packaging: The malware is packaged within a legitimate NSIS installer, which is commonly used for software installations. This disguises the malicious nature of the file. Mimicking Legitimate Names: The malware uses names like "Firefox installer" and "Google updater" to blend in with expected software updates or installations. This makes it less likely to raise suspicion during analysis or execution. Exploiting Trust: By leveraging the trust associated with well-known software like Firefox and Google Update, the malware tricks users into running it without realizing the true intent. This tactic relies on exploiting the user's trust in familiar software names and packaging to bypass security measures and gain a foothold on the 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. Evaluating the delay: If the elapsed time exceeds 100 milliseconds, it interprets this as a sign of being in a virtualized or sandboxed environment where time manipulation might occur. Aborting execution: If the delay exceeds the threshold, Goofy Guineapig terminates its own process, preventing further analysis or interaction. This technique aims to detect environments where time is artificially accelerated or manipulated, which is common in security testing and analysis tools.

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 disk size and compares it to a predefined threshold. If the disk size falls outside the expected range, it assumes the environment is a sandbox or virtual machine and terminates execution. Checking physical memory size: Similar to disk size, it probes the system's physical memory size and compares it to a predefined threshold. An unexpected memory size indicates a potential sandbox or virtualized environment, leading to termination. Checking the number of logical processors: It determines the number of logical processors available on the system and compares it to a predefined threshold. An abnormal number of logical processors could suggest a virtualized environment, triggering termination. These checks are designed to identify common characteristics of sandboxed or virtualized environments, allowing Goofy Guineapig to avoid detection and execution within those controlled settings.

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 User Activity Based Checks for Virtualization/Sandbox Evasion by monitoring the running processes on the infected system. It specifically looks for processes whose names contain strings like "dbg", "debug", or "ida", which are often associated with debugging tools and reverse engineering environments. If any of these processes are found running, Goofy Guineapig will terminate its own execution, effectively avoiding analysis in a sandbox or debugging environment. Let me know if you'd like more details on any specific aspect of this evasion technique!

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: The action "Obfuscated Files or Information: Software Packing" is performed by Goofy Guineapig using the UPX (Ultimate Packer for eXecutables) packing technique. This means the malware's executable code is compressed and encrypted within a larger, legitimate NSIS installer package. This makes it harder for security software to detect and analyze the malicious code. Let me know if you'd like more details on UPX or other malware packing 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 byte in the string is XORed with a specific value (0x6D in this case). This effectively scrambles the characters, making them unreadable without applying the inverse XOR operation. Single-Byte Subtraction: Similar to XOR, each byte in the string is subtracted by a specific value (0x73). This also results in a scrambled representation of the original string. These techniques are employed throughout the binary, hiding the true nature of the strings until they are decoded by the malware itself. Let me know if you'd like more details on any specific aspect of the deobfuscation process!

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 system’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 malicious activities by using process hollowing on the legitimate dllhost.exe process. Here's how it works: Target Selection: Goofy Guineapig identifies dllhost.exe as its target for process hollowing. Process Hollowing: Goofy Guineapig takes control of an existing dllhost.exe process and replaces its original code with its own malicious code. This effectively makes dllhost.exe a vessel for Goofy Guineapig's operations. Hidden Execution: Crucially, Goofy Guineapig configures the hollowed dllhost.exe process to run hidden. This means the process won't have a visible window, making it harder to detect through traditional monitoring methods. Malicious Activity: The hollowed dllhost.exe now executes Goofy Guineapig's commands, such as collecting information or running additional plugins, all while remaining hidden from the user's view. This technique allows Goofy Guineapig to operate stealthily, evading detection by security tools that rely on visible process windows or traditional signature-based detection methods.

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 moves the malicious files to a directory named "ProgramData". This directory is typically hidden by default, making it less likely to be noticed. Deletion from Download Location: Finally, Goofy Guineapig deletes the original files from the initial download location. This action removes the direct evidence of the malware's presence, making it harder to detect. This technique aims to conceal its tracks and avoid detection by security tools that might be monitoring the initial 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 trusted by the system and is likely already present on the target machine. Malicious DLL: Alongside the legitimate executable, Goofy Guineapig also drops a malicious DLL (the actual backdoor payload). 2. The Trick: Side-Loading: When GoogleUpdate.exe runs, it loads any DLLs found in its directory. Goofy Guineapig's malicious DLL is placed in this same directory, ensuring it gets loaded alongside the legitimate executable. 3. The Hijack: Function Execution: Once loaded, the malicious DLL's functions become accessible to GoogleUpdate.exe. Goofy Guineapig can then leverage these functions to: Execute its own code: The backdoor can hijack the legitimate executable's processes, effectively running its own malicious code under the guise of a trusted application. Steal data: It can access sensitive information handled by GoogleUpdate.exe or other system processes. Establish communication: It can use the legitimate executable's network connections to communicate with its command-and-control server. Why This Works: Trust: The system trusts GoogleUpdate.exe, so it doesn't question the legitimacy of the loaded DLL. Persistence: The malicious DLL is loaded every time GoogleUpdate.exe runs, ensuring the backdoor remains active. Let me know if you'd like more details on any specific aspect of this process!

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 hollowing on the dllhost.exe binary as follows: Download: The malware first downloads malicious content from its Command and Control (C2) server. Hollowing: It then uses process hollowing techniques to replace the legitimate code within the dllhost.exe process with the downloaded malicious content. This effectively injects the malicious code into a legitimate process, making it harder to detect. Execution: The hollowed dllhost.exe process now executes the injected malicious code, allowing Goofy Guineapig to carry out its malicious activities under the guise of a legitimate system process. Let me know if you'd like more details on any specific aspect of process hollowing.

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 leverages the legitimate Windows binaries rundll32.exe and url.dll for persistence. Here's how it works: Batch Script Execution: The malware's persistence mechanism starts with a batch script that executes rundll32.exe. URL Protocol Handler: The rundll32.exe command is structured to use the url.dll library's FileProtocolHandler function. This function is typically used to handle file associations and open files using associated applications. Loading the Malicious DLL: By cleverly manipulating the arguments passed to rundll32.exe, Goofy Guineapig tricks the system into loading a malicious DLL. This DLL contains the actual backdoor functionality. Persistence: Because rundll32.exe is a legitimate system process, its execution is often less scrutinized by security software. This allows Goofy Guineapig to establish persistent access to the infected system. Essentially, Goofy Guineapig exploits the standard functionality of rundll32.exe and url.dll to disguise its malicious DLL loading as a legitimate file association operation.

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 installed on the victim machine.
- The malware has the necessary permissions to access the victim machine's information.
- The victim machine is running a Windows operating system.
- The necessary Windows APIs are available.
- The malware has the necessary configuration to collect system information.
- The malware has the necessary functionality to collect system information.
- The malware has the necessary communication method to send the collected information.
- The malware has the necessary embedded configuration string.
- The malware has the necessary functionality to communicate using UDP, KCP protocol, or direct socket communications.
- The malware has the necessary functionality to collect adapter information and host name.
- Windows operating system.
- Goofy Guineapig malware.
- Necessary Windows APIs.
- Necessary configuration to collect system information.
- Necessary communication method (HTTPS).
- Embedded configuration string.
- Functionality to collect adapter information and host name.
- Functionality to communicate using UDP, KCP protocol, or direct socket communications.
- Functionality to collect system information.
- Necessary permissions to access the victim machine's information.
- A way to send the collected information (HTTPS).
- A way to collect and process the information (Windows APIs).

Post-Conditions:

- The backdoor is installed on the victim machine.
- The backdoor supports multiple communications methods, including HTTP, HTTPS, and KCP.
- The configuration for the binary is embedded in the binary itself.
- Command and control communications occur over HTTPS.
- Many defence evasion techniques are implemented throughout execution.
- The backdoor contains the functionality to communicate using UDP and the KCP protocol, or direct socket communications.
- The backdoor communicates over the non-standard HTTPS port 4443.
- The backdoor encodes shellcode using a specific XOR algorithm.
- The backdoor collects information about the victim machine, including operating system caption, antivirus product display name, adapters information, host and host name, and computer name.
- The backdoor sends information about the infected machine in each C2 packet, as an obfuscated ‘Authorization’ string in the HTTP header.
- The backdoor uses HTTPS for its C2 communications.
- The backdoor creates a unique identifier for the victim machine by concatenating adapter information and host name, and taking an MD5 hash of the result.
- Logs of HTTP, HTTPS, and KCP communications.
- Embedded configuration in the binary.
- Defence evasion techniques logs.
- UDP and KCP protocol communications logs.

- Direct socket communications logs.
- Non-standard HTTPS port 4443 communications logs.
- Encoded shellcode files.
- Collected information about the victim machine (operating system caption, antivirus product display name, adapters information, host and host name, and computer name).
- Obfuscated ‘Authorization’ strings in HTTP headers.
- HTTPS communications logs.
- Unique identifier files for the victim machine.
- MD5 hash files of concatenated adapter information and host name.
- Logs of failed information collection attempts.
- Pipe-delimited list files containing host information.
- Prepend bytes of MD5 hash files.

Attack Step 3.1

==========
Name: System Information Discovery as used by Goofy Guineapig
Description: Goofy Guineapig performs System Information Discovery by embedding information about the infected machine within each Command and Control (C2) communication packet. Here's how it's done: Data Collection: Goofy Guineapig gathers various system details, including: Operating system caption Antivirus product display name Adapters information Host and host name Computer name Obfuscation: This collected data is obfuscated and encoded into a string. HTTP Header Embedding: The obfuscated system information is then inserted into the "Authorization" header of each HTTPS request sent to the C2 server. Transmission: The C2 server receives these requests containing the hidden system information. This method allows Goofy Guineapig to discreetly exfiltrate sensitive system details to its controllers without raising immediate suspicion.

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 Goofy Guineapig malware is present on the system.
- The system has a physical memory size exceeding 2GB.
- The system has a disk size exceeding 1GB.
- The system has a network connection.
- The HTTPS protocol is available.
- The RC4 encryption algorithm is available.
- The key 'uirWmX3fSBhplR2sj' is available.
- The system has a process with a window containing the relevant strings.
- Environment: Windows system
- Tools: Debugger, analysis tools
- Resources: Physical memory, disk space, network connection
- Availability of HTTPS protocol
- Availability of RC4 encryption algorithm
- Key 'uirWmX3fSBhplR2sj'
- The embedded configuration string is present.
- The configuration string contains UDP rather than HTTP (S).
- The physical memory size of the machine exceeds 2GB.
- The disk is more than 1GB in size.
- The machine is not running a debugger.
- The machine is not running other analysis tools.
- The machine has a non-standard HTTPS port 4443 available.
- The KCP protocol is available.
- Direct socket communications are available.
- The embedded configuration string contains the necessary information for UDP communication.
- Environment: Windows operating system.
- Tools:
- Resources:
- Availability of UDP protocol.
- Availability of direct socket communications.
- The Goofy Guineapig malware is present in the environment.
- The environment is a Windows system.
- The malware has been loaded by a legitimate signed executable.
- The malware has maintained persistence using a Windows service.
- The environment has a physical memory size exceeding 2GB.
- The environment has a disk size exceeding 1GB.
- The environment is not running in a sandbox or virtual machine.
- Environment: Windows system with a physical memory size exceeding 2GB and a disk size exceeding 1GB.
- Tools: None explicitly mentioned, but the presence of a debugger and analysis tools is inferred.
- Resources:
- Availability:

Post-Conditions:

- Malicious files are present in the ProgramData directory.

- The initial directory to which the files were downloaded will only contain the files the recipient likely intended to download.
- The malware implements some basic anti-sandbox / anti-VM techniques.
- The C2 communications are HTTPS and RC4 encrypted with the key: uirWmX3fSBhplR2sj.
- The loader for the Goofy Guineapig malware is UPX packed.
- The malware uses non-standard HTTPS port 4443.
- The malware communicates using UDP and the KCP protocol.
- The malware sends information about the infected machine in each C2 packet.
- The malware deletes itself after the final command in the batch script.
- Logs of C2 communications.
- Files in the ProgramData directory.
- Network connections to non-standard HTTPS port 4443.
- Network connections using UDP and the KCP protocol.
- Encrypted C2 communications logs.
- UPX packed loader files.
- HTTP headers with obfuscated 'Authorization' string.
- HTTPS communications logs.
- RC4 encryption keys.
- Files in the initial directory to which the files were downloaded.
- Malware infection
- Data exfiltration
- Command and control communications over HTTPS port 4443
- Implementation of anti-sandbox/anti-VM techniques
- Session enumeration
- Logon session searching
- Querying for username and client protocol type
- Basic defence evasion techniques
- Embedded configuration string hardcoded in the binary
- RC4 key stack string setup
- Log entries of command and control communications
- Network connections to HTTPS port 4443
- Files created by the malware
- Registry entries modified by the malware
- System calls related to session enumeration
- Logon session records
- Files with encoded shellcode
- Embedded configuration string in the binary
- RC4 key stack string setup in memory
- Logs of anti-sandbox/anti-VM techniques
- Network connections to the proxy server (if used)
- Files with XORed data
- Logs of MD5 hash calculations
- Network connections to the static.tcplog.com server
- Files with encoded configuration strings
- Persistence of malware through a Windows service.
- Malicious files present in the ProgramData directory.
- Non-standard HTTPS port 4443 used for communication.
- Basic anti-sandbox/anti-VM techniques implemented.
- Malware maintains persistence even after initial directory cleanup.
- Malicious DLL loaded by a legitimate signed executable.
- Malware uses flawed logic for secondary process check.
- Malware uses XOR encryption with key 0x59.
- Malware uses MD5 hashing with multiple iterations.

- Malware uses KCP protocol for UDP communication.
- Logs of non-standard HTTPS port 4443 usage.
- Files in the ProgramData directory.
- Network connections to static.tcplog.com:4443.
- Logs of XOR encryption with key 0x59.
- Logs of MD5 hashing with multiple iterations.
- Logs of KCP protocol usage.
- Logs of basic anti-sandbox/anti-VM techniques.
- Windows service logs indicating malware persistence.
- Malicious DLL files in the system directory.
- Logs of flawed logic used in secondary process check.

Attack Step 4.1

==========
Name: Application Layer Protocol: Web Protocols as used by Goofy Guineapig
Description: Goofy Guineapig utilizes the HTTPS protocol for its Command and Control (C2) communications. This means that the malware establishes encrypted connections over port 4443 (a non-standard port) to communicate with its command server. The encrypted nature of HTTPS helps the malware evade detection by security tools that might otherwise monitor traffic on standard ports.

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 determines its communication method based on an embedded configuration string. This string dictates whether it will use: UDP: A connectionless protocol that offers speed but lacks reliability. KCP: A fast and reliable protocol designed for low-latency, high-throughput connections often used in gaming and other real-time applications. Direct Socket Communications: A more traditional method of establishing a direct connection between two machines. The specific protocol used is chosen at runtime based on the value stored in the embedded configuration string.

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 utilizes the non-standard HTTPS port 4443 for its communication with the command and control server. This means that instead of using the standard HTTPS port 443, the malware establishes connections on port 4443 to avoid detection by security tools that might be monitoring common HTTPS traffic.

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.