Enhanced Attack Report

Goofy Guineapig

Generated on 2025-03-04

Disclaimer

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

Definitions

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

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

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

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

STIX

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

Milestone 1

Pre-Conditions:

- The malware Goofy Guineapig is running on a Windows system.
- The malware Goofy Guineapig has the necessary permissions to create or modify a Windows service.
- The malware Goofy Guineapig has access to the Windows service management API.
- The malware Goofy Guineapig has the necessary configuration or settings to create or modify a Windows service.
- The system has a valid Windows service configuration.
- The system has a valid Windows service management API.
- The malware Goofy Guineapig has the necessary resources to create or modify a Windows service.
- The system has a valid Windows operating system.
- Windows operating system.
- Windows service management API.
- Necessary permissions to create or modify a Windows service.
- Valid Windows service configuration.
- Valid Windows service management API.
- Necessary resources to create or modify a Windows service.
- Connectivity to the Windows service management API.
- Availability of the Goofy Guineapig malware.
- Availability of the necessary configuration or settings to create or modify a Windows service.
- Availability of the necessary tools to create or modify a Windows service.

Post-Conditions:

- Persistence mechanism is established.
- Malicious DLL is loaded by a legitimate executable.
- Windows service is created for persistence.
- Malware collects system information.
- Malware checks for automated analysis environment.
- Malware checks for debugging tools.
- Malware masquerades as a Firefox installer and a Google updater.
- Windows service created for persistence.
- Malicious DLL loaded by a legitimate executable.
- System information collected by malware.
- Automated analysis environment detection logs.
- Debugging tool detection logs.
- Firefox installer and Google updater masquerade logs.
- Malware execution logs.
- MD5 hash of concatenated adapter information and host name.
- Suspended dllhost.exe process.
- Created dllhost.exe process in memory.

Attack Step 1.1

==========

Name: Create or Modify System Process: Windows Service as used by Goofy Guineapig
Description: Goofy Guineapig achieves persistence by creating a Windows service. Here's a breakdown of how this likely works: Service Creation: Goofy Guineapig uses a function or API call to create a new Windows service. This service will have a unique name chosen by the malware. Service Configuration: The malware configures the service with specific settings: Service Name: A seemingly innocuous name to avoid detection. Service Description: A fabricated description to further disguise its purpose. Startup Type: Likely set to "Automatic" or "Automatic (Delayed Start)" to ensure it starts with the system. Service Binary: The path to Goofy Guineapig's executable file. Service Registration: The malware registers the newly created service with the Windows service manager. This makes the service a permanent part of the system. Persistence: When the system boots up or restarts, the Windows service manager automatically starts the Goofy Guineapig service, loading the malware into memory and allowing it to continue its malicious activities. Important Note: The exact implementation details of how Goofy Guineapig creates and configures the service are not provided in the context. Let me know if you have any other questions.

MITRE Technique

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

Indicators

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

Milestone 2

Pre-Conditions:

- The Goofy Guineapig malware is present on the system.
- The system has a legitimate Firefox installer and a Google updater.
- The system has a Windows operating system.
- The system has a ProgramData directory.
- The system has a GoogleUpdate directory.
- The system has a Windows service management system.
- The system has a process management system.
- The system has a system time checking system.
- The system has a properties checking system.
- The system has a legitimate Mozilla website.
- Environment: Windows operating system.
- Tools:
- Connectivity:
- Resources:
- No specific user account or privileges are required.
- The system time register is checked twice.
- The system time register is checked for a delay of more than 100 milliseconds.
- The system time register is checked for a delay of less than or equal to 100 milliseconds.
- The system time register is checked for a delay of less than 100 milliseconds.
- The system time register is checked for a delay of more than 100 milliseconds and less than or equal to 200 milliseconds.
- The system time register is checked for a delay of more than 100 milliseconds and less than 200 milliseconds.
- The system time register is checked for a delay of more than 100 milliseconds and less than or equal to 500 milliseconds.
- The system time register is checked for a delay of more than 100 milliseconds and less than 500 milliseconds.
- The system time register is checked for a delay of more than 100 milliseconds and less than or equal to 1000 milliseconds.
- The system time register is checked for a delay of more than 100 milliseconds and less than 1000 milliseconds.
- A system with a time register.
- A system with a time register that can be checked.
- A system with a time register that can be checked for a delay.
- A system with a time register that can be checked for a delay of more than 100 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than or equal to 200 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than 200 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than or equal to 500 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than 500 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than or equal to 1000 milliseconds.

- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than 1000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than or equal to 5000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than 5000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than or equal to 10000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than 10000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than or equal to 20000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than 20000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than or equal to 50000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than 50000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than or equal to 100000 milliseconds.
- A system with a time register that can be checked for a delay of more than 100 milliseconds and less than 100000 milliseconds.
- A system with a time register.
- The ability to check the time register.
- The ability to check the time register for a delay of more than 100 milliseconds.
- The system has a disk.
- The system has physical memory.
- The system has a logical processor.
- The system has a disk size greater than 1GB.
- The system has a physical memory size greater than 2GB.
- The system has a number of logical processors greater than 2.
- A system with a disk.
- A system with physical memory.
- A system with a logical processor.
- A system with a disk size greater than 1GB.
- A system with a physical memory size greater than 2GB.
- A system with a number of logical processors greater than 2.
- The system must be running a 32-bit or 64-bit operating system (inferred from the context).
- The system must have the necessary resources to run the malware (inferred from the context).
- The system must have a network connection (inferred from the context).
- The system must have the necessary permissions to execute the malware (inferred from the context).
- The malware must be present on the system (inferred from the context).
- The malware must be able to execute on the system (inferred from the context).
- A system is running.
- The system has processes running.
- The system has a physical memory size exceeding 2GB.
- The system has a disk size exceeding 1GB.
- The system is not being reverse engineered.
- The system is not being debugged.
- The system does not have a debugger running.
- The system does not have other analysis tools running.

- Environment: A Windows system with a physical memory size exceeding 2GB and a disk size exceeding 1GB.
- Tools: None explicitly stated, but the system must have processes running, and the malware must have the capability to check for specific process names.
- Connectivity: None explicitly stated, but the system must be able to communicate with the malware.
- Resources: None explicitly stated, but the system must have sufficient resources to run the malware and perform the checks.
- Availability of the malware: The Goofy Guineapig malware must be present on the system.
- Availability of the necessary plugins: The malware must have the necessary plugins to perform the checks.
- Network connectivity to tcplog.com: The system must be able to communicate with tcplog.com over HTTPS using GET and POST requests.
- The malware Goofy Guineapig is present.
- The malware Goofy Guineapig has been executed.
- The malware Goofy Guineapig has access to the system's memory.
- The system's memory size exceeds 2GB.
- The system's disk size exceeds 1GB.
- The system has a legitimate NSIS installer.
- The system has a legitimate FireFox installation.
- The system has a legitimate GoogleUpdate.exe executable.
- Environment: Windows operating system.
- Tools: UPX packer, NSIS installer, FireFox installation, GoogleUpdate.exe executable.
- Connectivity: None.
- Resources: System memory, disk space, CPU resources.
- Availability of legitimate NSIS installer, FireFox installation, and GoogleUpdate.exe executable.
- Presence of a legitimate FireFox NSIS installation package.
- Presence of a legitimate GoogleUpdate.exe executable.
- Availability of system memory and disk space to pack and unpack the malware.
- The binary of Goofy Guineapig is available.
- The binary of Goofy Guineapig is UPX packed.
- The binary of Goofy Guineapig is packaged in with a legitimate NSIS installer.
- The binary of Goofy Guineapig contains stack-based strings.
- The binary of Goofy Guineapig is obfuscated with single byte XOR or subtraction.
- The binary of Goofy Guineapig is RC4 encrypted.
- The key for RC4 encryption is known.
- The URL string in the backdoor of Goofy Guineapig is under one-byte XOR obfuscation.
- The key for one-byte XOR obfuscation is known.
- A computer with a stable operating system.
- A tool capable of analyzing and deobfuscating binary files.
- A tool capable of decrypting RC4 encrypted files.
- Access to the binary file of Goofy Guineapig.
- Access to the key for RC4 encryption.
- Access to the key for one-byte XOR obfuscation.
- A stable internet connection for research purposes.
- A text editor or a code analysis tool to view and analyze the deobfuscated code.
- A system with sufficient resources (CPU, memory, and disk space) to handle the deobfuscation process.
- The Goofy Guineapig malware is present in the system.
- The system has a legitimate dllhost.exe process.
- The Goofy Guineapig malware has the capability to perform process hollowing.
- The system has a hidden directory, ProgramData, by default.
- The system has a legitimate GoogleUpdate.exe executable.
- The system has a legitimate NSIS installation package.

- The system has a legitimate FireFox installation.
- The system has a legitimate FireFox NSIS installation package.
- The system has a legitimate url.dll binary.
- The system has a legitimate rundll32.exe binary.
- Environment: Windows operating system.
- Tools: None explicitly stated, but the presence of legitimate binaries and processes implies that the system has the necessary tools and libraries.
- Connectivity: None explicitly stated, but the presence of a legitimate C2 server (tcplog.com) implies that the system has internet connectivity.
- Resources: None explicitly stated, but the presence of legitimate binaries and processes implies that the system has sufficient resources (CPU, memory, etc.) to run the malware.
- Availability of legitimate binaries and processes (dllhost.exe, GoogleUpdate.exe, NSIS installation package, FireFox installation, url.dll, rundll32.exe).
- Presence of a hidden directory, ProgramData, by default.
- Ability to perform process hollowing.
- Ability to read and write to process memory.
- Ability to change thread context.
- Ability to execute shellcode.
- The Goofy Guineapig malware is initially downloaded to a specific location.
- The malware is moved to a legitimate-looking directory.
- The initial download location is accessible.
- The ProgramData directory is accessible.
- The directory containing the extracted Firefox files is accessible.
- The batch script is executed.
- The GoogleUpdate.exe and Goopdate.dll files are present in the original file path.
- The UPX packed NSIS installer is available.
- The Goopdate.dll DLL is executed.
- The C:\\ProgramData\\GoogleUpdate directory is accessible.
- Environment: Windows operating system.
- Tools: Batch script, UPX packed NSIS installer, Goopdate.dll DLL.
- Connectivity: None specified.
- Resources:
- The Goofy Guineapig malware is present in the system.
- A legitimate executable is installed by the Goofy Guineapig loader.
- A malicious DLL is loaded by the legitimate executable.
- The legitimate executable is installed alongside the malicious DLL.
- The malicious DLL is sideloaded by a legitimate, signed, executable.
- The legitimate executable is signed.
- The malicious DLL is available.
- The Goofy Guineapig loader is available.
- The legitimate executable is installed.
- The malicious DLL is loaded.
- The legitimate executable is running.
- Environment: Windows operating system.
- Tools: Goofy Guineapig loader, legitimate executable, malicious DLL.
- Connectivity: None specified.
- Resources: None specified.
- Availability of legitimate executable and malicious DLL.
- Availability of Goofy Guineapig loader.
- Signed legitimate executable.
- Internet connectivity to download the malicious DLL (inferred from the context).
- Presence of a legitimate Firefox NSIS installation package (inferred from the context).
- Presence of a legitimate FireFox executable (inferred from the context).

- The dllhost.exe binary is present on the system.
- The Goofy Guineapig malware is installed on the system.
- The system has a legitimate dllhost.exe process running.
- The system has a legitimate dllhost.exe process that can be suspended.
- The system has a legitimate dllhost.exe process that can be hollowed.
- The system has a legitimate dllhost.exe process that can be injected with content.
- The system has a legitimate dllhost.exe process that can be resumed after injection.
- The system has a legitimate dllhost.exe process that can be executed.
- The system has a legitimate dllhost.exe process that can load the malicious DLL.
- The system has a legitimate dllhost.exe process that can be used for process hollowing.
- Environment: Windows operating system.
- Tools: dllhost.exe, rundll32.exe, url.dll, Goofy Guineapig malware.
- Connectivity: None.
- Resources: System memory, system processes, system files.
- Availability of legitimate dllhost.exe process.
- Availability of legitimate rundll32.exe process.
- Availability of legitimate url.dll process.
- Availability of system memory to hollow the dllhost.exe process.
- Availability of system processes to inject content into the dllhost.exe process.
- Availability of system files to load the malicious DLL.
- A legitimate signed executable is installed.
- A malicious DLL is loaded by the legitimate executable.
- The malicious DLL is sideloaded by the legitimate executable.
- The legitimate executable is able to execute the malicious DLL.
- The malicious DLL is able to communicate with the C2 server.
- The C2 server is accessible over the non-standard HTTPS port 4443.
- The Windows service is created and running.
- The Goofy Guineapig loader is executed.
- The Goofy Guineapig loader is able to download content from the C2 server.
- The Goofy Guineapig loader is able to inject content into the dllhost.exe binary.
- Environment: Windows operating system.
- Tools: Rundll32.exe, url.dll, dllhost.exe, Goofy Guineapig loader, malicious DLL.
- Connectivity: Access to the C2 server over the non-standard HTTPS port 4443.
- Resources: Available disk space to store the malicious DLL and the Goofy Guineapig loader.
- Network connectivity: Ability to communicate with the C2 server.
- User privileges: Administrator privileges to create and run the Windows service.
- System configuration: Ability to execute the legitimate executable and load the malicious DLL.
- Software: Firefox NSIS installation package, GoogleUpdate.exe, and the Goofy Guineapig loader.

Post-Conditions:

- Persistence as a Windows service
- Malware installed on the system
- Backdoor established for remote access
- Potential for additional plugins to be loaded
- System process modified to evade detection
- Legitimate system files masqueraded as malware
- Network connections established for C2 communications
- Log files modified to evade detection
- System configuration modified for persistence
- Malware indicators left in system memory
- Windows service logs

- System process logs
- Network connection logs
- Log files modified to evade detection
- System configuration files modified
- Malware files left on the system
- Network traffic logs
- System memory dumps
- Registry keys modified for persistence
- System files modified to masquerade as malware
- The malware will not continue execution if the disk size is less than 1GB.
- The malware will not continue execution if the physical memory size is less than 2GB.
- The malware will not continue execution if the number of logical processors is less than expected.
- The malware will not continue execution if the system time is not as expected.
- The malware will not continue execution if a debugger or other analysis tools are running.
- The malware will not continue execution if the machine is being reverse engineered or debugged.
- The malware will not continue execution if the process is running in an automated analysis environment.
- The malware will not continue execution if a process containing the string 'dbg', 'debug', or 'ida' is running.
- Logs of system time checks.
- Logs of physical memory size checks.
- Logs of disk size checks.
- Logs of logical processor checks.
- Logs of process checks for 'dbg', 'debug', or 'ida' strings.
- Logs of automated analysis environment checks.
- Logs of reverse engineering or debugging checks.
- Network connections for HTTP, HTTPS, and KCP communications.
- Files containing configuration for the malware.
- Files containing embedded configuration for the binary.
- Logs of sandbox detection checks.
- Logs of virtual machine detection checks.
- Files created by the malware during execution.
- Modified system files or registry entries created by the malware.
- Network connections for command and control communications over HTTPS.
- The malware will exit if any of the sandbox detection checks fail.
- 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 the disk size, physical memory size, or number of logical processors are not sufficient.
- The malware will not continue execution if the system is being reverse engineered or debugged.
- The malware will not continue execution if the system time indicates the process is running in an automated analysis environment.
- The malware will not continue execution if the properties of the infected machine indicate it is running in an automated analysis environment.
- The malware will not continue execution if the running processes indicate the system is being reverse engineered or debugged.
- The malware will not continue execution if the configuration for the binary analyzed results in command and control communications occurring over an unsupported protocol.
- The malware will not continue execution if the configuration for the binary analyzed results in command and control communications occurring over an unsupported method.
- Logs of the malware's sandbox detection checks.
- Files created by the malware, such as the dllhost.exe process.
- Network connections to the command and control server over HTTPS.

- Network connections to the command and control server over HTTP.
- Network connections to the command and control server over KCP.
- Files created by the malware's process hollowing technique.
- Files created by the malware's plugin execution.
- Logs of the malware's system time checks.
- Logs of the malware's properties checks.
- Logs of the malware's running processes checks.
- Files created by the malware's defence evasion techniques.
- Network connections to the malware's configuration server.
- Files created by the malware's configuration download.
- Logs of the malware's command and control communications.
- Files created by the malware's plugin execution in the current process.
- The malware will not continue execution if any process containing the string 'dbg', 'debug', or 'ida' is running.
- The malware will not continue execution if any of the following checks fail: disk size, physical memory size, or number of logical processors.
- The malware will exit if the number of logical processors does not exceed 2.
- The malware will impersonate the user associated with the session ID.
- An instance of the legitimate dllhost.exe process will be created in the suspended state.
- Command data will be written into the process memory.
- The thread context will be changed to point at the new data and the thread will be resumed.
- The malware will check for processes running on a system which indicate that it is being reverse engineered or debugged.
- The malware will check the properties of the infected machine, as well as the running processes and system time.
- The malware will collect information about the infected machine or run additional plugins.
- Logs of system checks (disk size, physical memory size, number of logical processors).
- Logs of process checks (running processes containing 'dbg', 'debug', or 'ida').
- Logs of user impersonation.
- Creation of a legitimate dllhost.exe process in the suspended state.
- Writing of command data into the process memory.
- Changes to the thread context.
- Network connections for HTTP, HTTPS, and KCP communications.
- Files created by the malware (configuration files, plugin files).
- Logs of sandbox detection checks.
- Logs of plugin execution.
- Malware infection
- Data exfiltration
- Command and control communications over HTTPS
- Defence evasion techniques implemented
- Malicious DLL sideloaded by legitimate executable
- Malware masquerading as legitimate processes
- Malicious files dropped alongside legitimate files
- Malware checking for automated analysis environment
- Malware checking for physical memory size and disk size
- Malware implementing anti-sandbox/anti-VM techniques
- Malicious DLL file created in system directory
- Malicious executable file created in system directory
- Log entries in Windows Event Viewer indicating malware activity
- Network connections to command and control server
- HTTPS traffic to command and control server
- RC4 encrypted data transmitted over HTTPS
- MD5 hash of concatenated adapter information and host name

- Malicious files dropped in user's Firefox directory
- Sideloaded malicious DLL in GoogleUpdate.exe process
- Log entries in NSIS installer indicating malware installation
- Malware configuration embedded in binary file
- UPX packed loader file created in system directory
- Obfuscated files or information in malware binary
- Stack-based strings obfuscated with single byte XOR or subtraction
- Malware checking for process window text and window title
- The backdoor is established on the system.
- The backdoor supports multiple communications methods, including HTTP, HTTPS, and KCP.
- The configuration is embedded in the binary.
- Command and control communications occur over HTTPS.
- Many defence evasion techniques are implemented throughout execution.
- The process name is changed to evade detection.
- The window text is modified to evade detection.
- The system is vulnerable to process exit due to flawed logic.
- The system is vulnerable to single-byte XOR obfuscation.
- The system is vulnerable to RC4 encryption.
- Logs of HTTP, HTTPS, and KCP communications.
- Embedded configuration string in the binary.
- Files created by the backdoor, such as logs or communication data.
- Network connections to the C2 server.
- Files created by the RC4 encryption, such as encrypted communication data.
- Logs of process name and window text modifications.
- Files created by the single-byte XOR obfuscation, such as obfuscated communication data.
- Logs of defence evasion techniques, such as ROR and XOR.
- Files created by the loader binary, such as obfuscated strings.
- Logs of system vulnerabilities, such as process exit due to flawed logic.
- Process hollowing on dllhost.exe
- Hidden Window
- Creation of a legitimate looking directory
- Deletion of files from the initial download location
- Persistence through a service
- Decoding and loading of the config.dat file
- Modification of system time checks
- Checking of properties of the infected machine
- Checking of running processes
- Masquerading as legitimate processes
- Trojanisation of a legitimate FireFox NSIS installation package
- Sideloaded malicious DLL by a legitimate, signed, executable
- Bundling of malicious files in a UPX packed NSIS installer
- Creation of a service for persistence
- Modification of the GoogleUpdate.exe executable
- Logs of process hollowing on dllhost.exe
- Hidden Window process
- Creation of a legitimate looking directory
- Deleted files from the initial download location
- Service logs for persistence
- Decoded config.dat file
- Modified system time checks
- Logs of properties of the infected machine
- Logs of running processes
- Logs of masquerading as legitimate processes

- Trojanised FireFox NSIS installation package
- Sideloaded malicious DLL by a legitimate, signed, executable
- Logs of bundling of malicious files in a UPX packed NSIS installer
- Service logs for persistence
- Modified GoogleUpdate.exe executable
- Network connections to C&C; domains
- Logs of KEYPLUG samples
- Files moved to a legitimate looking directory
- Logs of file deletion
- Logs of system time checks modification
- Malicious files are present in the ProgramData directory.
- Initial download directory only contains intended files.
- Malware persistence mechanism is installed.
- Malicious files are deleted from initial download location.
- Batch script deletes itself.
- GoogleUpdate.exe process is restarted from the ProgramData directory.
- Malware checks for sandbox detection and exits if checks fail.
- Logs of malware execution and persistence mechanism installation.
- Files in the ProgramData directory.
- Deleted files in the initial download location.
- Batch script logs.
- Network connections to command and control servers over HTTPS.
- Logs of system time checks and process checks.
- Files in the 'Functionality (Defence Evasion)' section of the report.
- Logs of logical processor checks.
- Network connections to HTTP and KCP servers.
- Logs of malware configuration and command and control communications.
- Persistence of a malicious DLL in the system.
- Creation of a Windows service for persistence.
- Hijacking of legitimate executable processes.
- Loading of malicious DLL by legitimate executable.
- Masquerading as legitimate processes.
- Trojanisation of a legitimate Firefox NSIS installation package.
- Sideloaded malicious DLL by legitimate, signed, executable.
- Communication over non-standard HTTPS port 4443.
- Process hollowing on dllhost.exe binary.
- Execution of malicious DLL through rundll32.exe and url.dll.
- Malicious DLL files in system directories.
- Windows service created for persistence.
- Logs of hijacked legitimate executable processes.
- Files created by legitimate executable with malicious DLL loaded.
- Network connections to static.tcplog.com.
- Network connections to moziila.net and hxxp://\x00refox.com.cn.
- Files created by trojanised Firefox NSIS installation package.
- Logs of sideloaded malicious DLL by legitimate executable.
- Network connections to HTTPS port 4443.
- Logs of process hollowing on dllhost.exe binary.
- Files created by rundll32.exe and url.dll execution.
- Malicious certificate belonging to KgLKgLKgLKgLKgLKgLKgLKgLKgLKgLKgL (Founder Technology Group Corporation).
- The legitimate executable is installed by the Goofy Guineapig loader, alongside a malicious DLL which will be loaded by the legitimate executable.
- The legitimate executable is hollowed, allowing the malicious DLL to be injected into the process.

- The malicious DLL is sideloaded by the legitimate, signed, executable GoogleUpdate.exe.
- The malicious DLL is loaded by the legitimate dllhost.exe process.
- The process is created hidden.
- The malicious loader is reused, indicating a connection to APT41.
- The malicious loader is hosted on hxxps://download-ssl.moziila.net.
- The actor is attempting to masquerade as a legitimate Firefox service.
- The malicious loader is signed using a legitimate certificate belonging to KgLKgLKgLKgLKgLKgLKgLKgLKgLKgLKgL (Founder Technology Group Corporation).
- A legitimate executable is installed by the Goofy Guineapig loader.
- A malicious DLL is created and loaded by the legitimate executable.
- The legitimate dllhost.exe process is hollowed and the malicious DLL is injected into the process.
- The malicious DLL is sideloaded by the legitimate GoogleUpdate.exe executable.
- The process is created hidden, leaving no visible process in the task manager.
- Logs of the malicious loader being reused, indicating a connection to APT41.
- Network connections to hxxps://download-ssl.moziila.net.
- Files created by the malicious loader, including the malicious DLL and the legitimate executable.
- A legitimate certificate is used to sign the malicious loader, belonging to KgLKgLKgLKgLKgLKgLKgLKgLKgLKgLKgL (Founder Technology Group Corporation).
- A hidden window is created by the Goofy Guineapig process.
- Persistence of malicious DLL through Windows service.
- Creation of hidden processes (dllhost.exe).
- Process hollowing on dllhost.exe.
- Injection of content downloaded by C2 into dllhost.exe.
- Execution of malicious DLL through legitimate executable (GoogleUpdate.exe).
- Sideloaded malicious DLL through legitimate executable (GoogleUpdate.exe).
- Creation of non-standard HTTPS connection on port 4443.
- Configuration embedded in the binary for command and control communications over HTTPS.
- Implementation of multiple defence evasion techniques.
- Masquerading as legitimate processes (Firefox NSIS installation package).
- Trojanisation of legitimate FireFox NSIS installation package.
- Creation of malicious Windows service.
- Logs of hidden processes (dllhost.exe).
- Files created by process hollowing on dllhost.exe.
- Network connections on non-standard HTTPS port 4443.
- Configuration files embedded in the binary for command and control communications over HTTPS.
- Logs of defence evasion techniques.
- Files created by masquerading as legitimate processes (Firefox NSIS installation package).
- Modified FireFox NSIS installation package.
- Sideloaded malicious DLL through legitimate executable (GoogleUpdate.exe).
- Logs of execution of malicious DLL through legitimate executable (GoogleUpdate.exe).
- Windows service created for persistence of malicious DLL.
- Network connections to command and control server over HTTPS.
- Logs of command and control communications over HTTPS.

Attack Step 2.1

==========

Name: Masquerading: Match Legitimate Name or Location as used by Goofy Guineapig
Description: Goofy Guineapig masquerades as a Firefox installer and a Google update component through the following actions: File Naming: The malware is initially packaged as a trojanized NSIS1 Firefox installer file. This implies it uses a filename that mimics legitimate Firefox installer files, tricking users into believing it's a genuine update. File Content: While the file might initially appear as a Firefox

installer, its actual content is malicious code. This deception relies on the user not examining the file's contents closely. Execution Behavior: Upon execution, Goofy Guineapig might display fake update interfaces or messages, further reinforcing the illusion of being a legitimate Google update component. This could involve mimicking the visual style and language of official Google update prompts. Essentially, Goofy Guineapig leverages social engineering by exploiting the user's trust in familiar software names and update processes.

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 implements Virtualization/Sandbox Evasion: Time Based Evasion by performing the following actions: Time Register Check: The malware checks the system's time register at a specific point in its execution. Delay Measurement: It introduces a deliberate delay of more than 100 milliseconds. Second Time Register Check: After the delay, Goofy Guineapig checks the time register again. Execution Halt: If the elapsed time exceeds the expected delay (more than 100 milliseconds), the malware terminates its execution. This technique aims to detect if the environment is running in a sandbox or virtualized environment, where time manipulation might be more common. By observing the time difference between the two checks, Goofy Guineapig can infer if the system's time is being artificially manipulated, indicating a potential sandbox or analysis environment. Let me know if you have any other questions about Goofy Guineapig's tactics or techniques!

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 system checks by utilizing system APIs or functions to retrieve information about the disk size, physical memory size, and the number of logical processors available on the system. It then compares these retrieved values against predefined thresholds or expected values. If any of the checks fail to meet these criteria, indicating a potential virtualization or sandbox environment, Goofy Guineapig will terminate its execution, preventing further malicious activity. Let me know if you'd like more details on specific APIs or functions that might be involved!

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 to evade sandbox environments by monitoring running processes for indicators of reverse engineering or debugging. Here's how it works: Process Name Scanning: Goofy Guineapig examines the names of all running processes on the infected system. Keyword Detection: It specifically looks for process names containing the strings "dbg", "debug", or "ida". These strings are commonly associated with debugging tools like IDA Pro, debuggers, and similar software used by security researchers. Execution Halt: If Goofy Guineapig detects any process with these keywords, it interprets this as a sign that it is being analyzed in a controlled environment and will immediately terminate its own execution. This technique aims to prevent the malware from being analyzed and understood by security researchers who might be using these tools. Let me know if you have any other questions about Goofy Guineapig or malware analysis techniques!

MITRE Technique

ID: T1497.002
Name: Virtualization/sandbox evasion: user activity based checks
Description: Adversaries may employ various user activity checks to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the

presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads. Adversaries may use the information learned from Virtualization/Sandbox Evasion during automated discovery to shape follow-on behaviors.
More info: https://attack.mitre.org/techniques/T1497/002

Indicators

- The file 'tmp.bat' is located in the directory 'C:\ProgramData\GoogleUpdate\GoogleUpdate'.
- A HTTP request with the User-Agent header 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36' was made.

Attack Step 2.5

==========
Name: Obfuscated Files or Information: Software Packing as used by Goofy Guineapig
Description: Goofy Guineapig is obfuscated by being UPX packed. This means its code is compressed and encrypted, making it harder to analyze and understand. Furthermore, it is packaged within a legitimate NSIS installer. This disguises the malicious nature of the payload, making it more likely to be accepted by users who might otherwise avoid downloading suspicious files. Let me know if you have any other actions you'd like explained!

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 '2b8aab068ef15cb05789da320b7099932a0a4166'.
- The file 'Firefox-latest.exe' has a SHA-256 hash of '19cef7f32e42cc674f7c76be3a5c691c543f4e018486c29153e7dde1a48af34c'.
- The file 'setup-stub.exe' has a MD5 hash of '180e0bb4b570c215bfe7abdf209402aa'.
- The file 'setup-stub.exe' has a SHA-1 hash of '6f5c07c50ce4976ddb3879ce65d3b2f96693dc4c'.
- The file 'setup-stub.exe' has a SHA-256 hash of '97f66bcd7d73917a8b59d9a1dcac21a58936bcafa91e757a9dfb8e5c320af40f56'.
- The file 'Goopdate.dll' has a MD5 hash of 'f98537517212068d0c57968876fc8204'.
- The file 'Goopdate.dll' has a SHA-1 hash of '7961930d13cb8d5056db64b6749356915fb4c272'.
- The file 'Goopdate.dll' has a SHA-256 hash of '12a29373c1f493f7757b755099bd e4770c310af3fde376176b6d792cd1c5e150'.
- The file 'config.dat' has a MD5 hash of '3dc1096e73db4886fb66ed9413ca994c'.
- The file 'config.dat' has a SHA-1 hash of '628ce6721b97fa12590356712fbffc5ae030781ce'.

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

Attack Step 2.6

==========
Name: Deobfuscate/Decode Files or Information as used by Goofy Guineapig
Description: Goofy Guineapig obfuscates its stack-based strings using two primary methods: Single-byte XOR: Each character in the string is XORed with a single byte key. This effectively scrambles the characters, making them unreadable without knowing the key. Subtraction: Similar to XOR, subtraction is used to shift the characters in the string. A specific value is subtracted from each character's ASCII code, again rendering the string unreadable without knowing the offset. To deobfuscate the strings, an analyst would need to identify the XOR key or the subtraction value used. This often involves analyzing the binary code for patterns or clues about the obfuscation technique.

MITRE Technique

ID: T1027.008
Name: Obfuscated files or information: stripped payloads
Description: Adversaries may attempt to make a payload difficult to analyze by removing symbols, strings, and other human readable information. Scripts and executables may contain variables names and other strings that help developers document code functionality. Symbols are often created by an operating 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 achieves "Hide Artifacts: Hidden Window" by employing a technique called process hollowing. Here's how it works: Target Selection: Goofy Guineapig targets the legitimate

dllhost.exe process. Process Hollowing: Goofy Guineapig injects its own malicious code into the memory space of the dllhost.exe process. Code Execution: The malicious code then overwrites the original dllhost.exe code, effectively replacing it with its own. Hidden Process: Since the malicious code is now running within the legitimate dllhost.exe process, it inherits the process's identity and privileges. This makes the malicious activity appear as if it's coming from a legitimate system process, effectively hiding it from detection. The result is a hidden process running Goofy Guineapig's malicious code, making it harder to detect and analyze.

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 that appears legitimate. Deletion: Finally, Goofy Guineapig deletes the original files from the initial download location, effectively removing the indicators of its presence. This technique aims to make it harder for security researchers and analysts to detect the malware's presence on a compromised system.

MITRE Technique

ID: T1070
Name: Indicator removal: relocate malware
Description: Adversaries may delete or modify artifacts generated within systems to remove evidence of their presence or hinder defenses. Various artifacts may be created by an adversary or something that can be attributed to an adversary’s actions. Typically these artifacts are used as defensive indicators related to monitored events, such as strings from downloaded files, logs that are generated from user actions, and other data analyzed by defenders. Location, format, and type of artifact (such as command or login history) are often specific to each platform.
More info: https://attack.mitre.org/techniques/T1070/

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 Installation: The Goofy Guineapig loader installs a legitimate executable (in this case, GoogleUpdate.exe) as part of a seemingly normal installation process. This executable is often signed, making it appear trustworthy. Malicious DLL Placement: Simultaneously, the loader installs a malicious DLL (the "Goofy Guineapig" DLL) in a location that is accessible to the legitimate executable. 2. The Hijack: Execution Flow: When the legitimate executable (e.g., GoogleUpdate.exe) is run, the operating system loads any DLLs specified in its dependencies. Side-Loading: The malicious DLL is included in the legitimate executable's dependencies, either directly or through a clever manipulation of the executable's structure. DLL Injection: As the legitimate executable loads, the malicious DLL is injected into its memory space. This effectively gives the malicious code control over the legitimate process. 3. The Consequences: Command and Control: The malicious DLL can now communicate with a command-and-control (C2) server, allowing the attacker to remotely control the infected system. Data Exfiltration: The malicious DLL can steal sensitive information from the infected system and send it back to the attacker. Further Malicious Activity: The attacker can use the compromised process to install additional malware, launch attacks, or perform other malicious actions. Key Points: Stealth: DLL side-loading is a stealthy technique because the malicious code is hidden within a legitimate process. Trust: The attacker exploits the trust users have in legitimate software to gain a foothold on their systems. Persistence: Goofy Guineapig's use of a Windows service ensures that the malicious DLL remains loaded even after the user logs off or restarts their computer. Let me know if you have any other questions.

MITRE Technique

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

Indicators

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

Attack Step 2.10

==========
Name: Process Injection: Process Hollowing as used by Goofy Guineapig
Description: Goofy Guineapig performs process hollowing on the dllhost.exe binary by: Downloading malicious content: Goofy Guineapig fetches malicious code from its Command and Control (C2) server. Allocating memory: It allocates a new memory region within the dllhost.exe process's address space, large enough to hold the downloaded malicious code. Overwriting existing code: Goofy Guineapig overwrites the original code of dllhost.exe with the downloaded malicious code. This effectively replaces the legitimate functionality of dllhost.exe with the malicious payload. Redirecting execution: It modifies the entry point of dllhost.exe to point to the beginning of the injected malicious code. When

dllhost.exe is executed, it now runs the malicious code instead of its original functionality. This technique allows Goofy Guineapig to inject its malicious payload into a legitimate process, making it harder to detect and analyze.

MITRE Technique

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

Indicators

- 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 rundll32.exe utility and the url.dll library to achieve persistence. Here's how it works: Legitimate Binary: Goofy Guineapig relies on a legitimate Windows executable. Rundll32 Invocation: It uses rundll32.exe to execute a function within url.dll. URL.dll Function: The specific function within url.dll is designed to load and execute the legitimate binary. Malicious DLL Loading: When the legitimate binary starts, it loads the malicious Goofy Guineapig DLL as a dependency. This effectively injects the malicious code into the legitimate process. Essentially, Goofy Guineapig disguises its malicious DLL loading as a normal system process by hijacking the functionality of rundll32.exe and url.dll. This makes it harder to detect as malicious activity.

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 Goofy Guineapig is loaded on the infected machine.
- The infected machine has a Windows operating system.
- The infected machine has antivirus software installed.
- The infected machine has network adapters.
- The infected machine has a host name.
- The infected machine has a computer name.
- The infected machine has a COM interface.
- The infected machine has WMI information available.
- The infected machine has relevant Windows APIs available.
- The infected machine has the ability to access the internet.
- A Windows operating system.
- Antivirus software.
- Network adapters.
- A host name.
- A computer name.
- A COM interface.
- WMI information.
- Relevant Windows APIs.
- Internet connectivity.
- The ability to access the internet.
- The malware Goofy Guineapig.
- The ability to execute plugins or process hollowing dllhost.exe.
- HTTPS connectivity to static.tcplog.com.
- The ability to collect and send information about the infected machine.

Post-Conditions:

- The infected machine's operating system caption is sent to the C2 server.
- The infected machine's antivirus product display name is sent to the C2 server.
- The infected machine's adapters information is sent to the C2 server.
- The infected machine's host and host name are sent to the C2 server.
- The infected machine's computer name is sent to the C2 server.
- The C2 server receives the collected information from the infected machine.
- The C2 server stores the collected information.
- The infected machine's system time is checked for indication of automated analysis environment.
- The infected machine's running processes are checked for indication of automated analysis environment.
- The infected machine's properties are checked for indication of automated analysis environment.
- The infected machine's C2 communications occur over HTTPS.
- The infected machine's C2 communications occur over HTTP (S).
- The infected machine's C2 communications occur over UDP.
- The infected machine's C2 communications occur over direct socket communications.
- The infected machine's defence evasion techniques are implemented.
- Logs of C2 communications on the C2 server.
- Files containing the collected information on the C2 server.
- Network connections between the infected machine and the C2 server.

- Files containing the malware's configuration on the infected machine.
- Files containing the malware's plugins on the infected machine.
- Logs of the malware's execution on the infected machine.
- Files containing the malware's defence evasion techniques on the infected machine.
- Network connections between the infected machine and the malware's C2 server.
- Files containing the infected machine's system time on the C2 server.
- Files containing the infected machine's running processes on the C2 server.
- Files containing the infected machine's properties on the C2 server.
- Files containing the infected machine's adapters information on the C2 server.
- Files containing the infected machine's host and host name on the C2 server.
- Files containing the infected machine's computer name on the C2 server.
- MD5 hash of the concatenated adapter information and host name on the C2 server.

Attack Step 3.1

==========

Name: System Information Discovery as used by Goofy Guineapig
Description: Goofy Guineapig performs System Information Discovery by collecting various pieces of information about the infected machine and sending them to its Command and Control (C2) server. Here's how it's done: Data Collection: Goofy Guineapig gathers data about the infected system using a combination of methods: COM and WMI: It leverages COM (Component Object Model) to access Windows Management Instrumentation (WMI), which provides information about the operating system and installed software. This is how it obtains the "Operating system caption" and "Antivirus product display name". Windows APIs: It utilizes various Windows APIs to collect other system details like adapter information, host and hostname, and computer name. Obfuscation: The collected information is then obfuscated and encoded into an "Authorization" string within the HTTP header of each communication sent to the C2 server. This makes it harder to detect and analyze the transmitted data. Transmission: Goofy Guineapig transmits this obfuscated "Authorization" string with each C2 communication, effectively sending the system information to its controllers. Let me know if you have any other questions about Goofy Guineapig's tactics or techniques.

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 malware Goofy Guineapig is present on the infected machine.
- The infected machine has a physical memory size exceeding 2GB.
- The infected machine has a disk size exceeding 1GB.
- The infected machine has a network connection.
- The infected machine has a web browser or a web server.
- The infected machine has a TLS (HTTPS) protocol implementation.
- The infected machine has a non-standard TLS port (4443) available.
- Environment: A Windows environment (inferred from the context).
- Tools: None explicitly stated, but a web browser or a web server is required (inferred from the context).
- Connectivity: A network connection is required (inferred from the context).
- Resources: A physical memory size exceeding 2GB and a disk size exceeding 1GB are required (inferred from the context).
- Availability of the malware Goofy Guineapig on the infected machine.
- Availability of the non-standard TLS port (4443) on the infected machine.
- The infected machine must be able to communicate over HTTPS using the TLS protocol.
- An embedded configuration string is present in the binary.
- The embedded configuration string contains the string 'UDP' or 'udp'.
- The embedded configuration string does not contain the string 'HTTP' or 'http'.
- The binary is running on a machine with a network connection.
- The binary has the capability to use the KCP protocol.
- Environment: A Windows machine with a network connection.
- Tools: None explicitly stated, but the binary must have the capability to use the KCP protocol.
- Connectivity: A network connection is required for the binary to communicate over UDP.
- Resources: The binary must have sufficient resources (e.g., memory, CPU) to execute the fallback channel functionality.
- Availability of the KCP protocol: The KCP protocol must be available on the machine for the binary to use it.
- Presence of the embedded configuration string: The binary must have the embedded configuration string present in order to determine the communication type.
- Capability to split the configuration string: The binary must have the capability to split the configuration string using the pipe character.
- The embedded configuration string contains a non-standard port number.
- The embedded configuration string contains the string 'HTTPS' or 'http'.
- The embedded configuration string contains the string 'UDP' or 'udp'.
- The physical memory size of the machine exceeds 2GB.
- The disk is more than 1GB in size.
- A Windows service is present.
- A legitimate signed executable is loaded.
- The malware is running on a machine with a physical memory size exceeding 2GB and a disk size exceeding 1GB.
- Environment: Windows operating system.
- Tools: None explicitly stated, but likely requires a debugger or analysis tool to inspect the malware's configuration.
- Connectivity: None explicitly stated, but likely requires network connectivity to communicate over the non-standard port.

- Resources: None explicitly stated, but likely requires sufficient system resources to run the malware and perform communication over the non-standard port.
- Availability of the malware binary with the embedded configuration string.
- Availability of a legitimate signed executable to load the malware.
- Presence of a Windows service to maintain persistence.
- Sufficient system resources to run the malware and perform communication over the non-standard port.

Post-Conditions:

- Infected machine information is sent to the C2 server.
- Malware is installed on the infected machine.
- Malware uses non-standard HTTPS port 4443 for communication.
- Malware uses KCP protocol for communication.
- Malware uses direct socket communications.
- Malware uses UDP protocol for communication.
- Malware implements anti-sandbox/anti-VM techniques.
- Malware checks physical memory size and disk size to evade detection.
- Malware collects system information and sends it to the C2 server.
- Malware uses RC4-encrypted communication.
- Logs of HTTP GET and POST requests.
- Logs of HTTPS communication on port 4443.
- Logs of KCP protocol communication.
- Logs of direct socket communications.
- Logs of UDP protocol communication.
- Files created by the malware.
- Network connections to the C2 server.
- Network connections to the KCP protocol server.
- Network connections to the direct socket server.
- Network connections to the UDP protocol server.
- Encrypted communication logs.
- Malware configuration files.
- Malware binary files.
- System information logs sent to the C2 server.
- RC4-encrypted communication logs.
- Command and control communications occur over HTTPS.
- Functionality to support UDP and direct socket communications is present.
- The backdoor supports multiple communications methods, including HTTP, HTTPS, and KCP.
- Command and control communications occur over HTTPS with no proxy.
- Communications occur over port 4443.
- The malware contains the functionality to communicate using UDP and the KCP protocol, or direct socket communications.
- Goofy Guineapig communicates over the non-standard HTTPS port 4443.
- C2 communications occur using HTTPS GET and POST requests.
- The binary also supports HTTP, UDP communications (using the KCP protocol), as well as raw TCP socket communications.
- Embedded configuration string containing UDP rather than HTTP (S).
- Temporary files created during the execution of the temp.bat script.
- Logs of HTTPS GET and POST requests.
- Network connections over port 4443.
- Files created during the execution of the malware, such as log files or configuration files.
- Process creation logs showing the running of the malware.

- System call logs showing the execution of the malware's commands.
- Network traffic logs showing the communication between the malware and its C2 server.
- Files created during the execution of the malware, such as temporary files or configuration files.
- Registry keys modified by the malware to enable its functionality.
- Malware persistence using a Windows service.
- Malware communication over non-standard HTTPS port 4443.
- Malware communication over UDP and KCP protocol.
- Malware communication over direct socket connections.
- Malware evasion techniques implemented.
- Malware anti-sandbox and anti-VM techniques implemented.
- Malware configuration string hardcoded in the binary.
- Malware communication with C2 domains.
- Windows service logs.
- Network connections to static.tcplog.com:4443.
- Network connections to hxxps://139.59.116.0:4443.
- Files created by the malware.
- Registry entries created by the malware.
- Logs of malware communication over UDP and KCP protocol.
- Logs of malware communication over direct socket connections.
- Logs of malware evasion techniques.
- Logs of malware anti-sandbox and anti-VM techniques.
- Configuration string hardcoded in the binary.

Attack Step 4.1

==========
Name: Application Layer Protocol: Web Protocols as used by Goofy Guineapig
Description: Goofy Guineapig's Action: Application Layer Protocol: Web Protocols Goofy Guineapig utilizes HTTPS as its Application Layer Protocol for Command and Control (C2) communications. Here's a breakdown of how this action is performed: HTTPS Protocol: Goofy Guineapig leverages the HTTPS protocol, which is a secure version of the HTTP protocol. HTTPS encrypts the communication between the infected machine and the C2 server, making it harder to intercept and analyze. Non-Standard Port: Goofy Guineapig deviates from the standard HTTPS port (443) and instead communicates over port 4443. This non-standard port helps the malware evade detection by security tools that might be primarily focused on monitoring traffic on the standard HTTPS port. Obscured Information: Within the HTTPS communication, Goofy Guineapig obfuscates system information and sends it to the C2 server as part of the "Authorization" string in the HTTP header. This further complicates analysis and tracking of the malware's activities. This combination of HTTPS, a non-standard port, and obfuscation techniques allows Goofy Guineapig to establish a secure and covert communication channel with its C2 server, enabling it to receive commands and exfiltrate stolen data.

MITRE Technique

ID: Unknown
Name: Publish/subscribe protocols
Description: Unknown
More info: https://attack.mitre.org/techniques/Unknown/

Indicators

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

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

Attack Step 4.2

==========
Name: Fallback Channels as used by Goofy Guineapig
Description: Goofy Guineapig utilizes several fallback channels for communication when its primary HTTPS connection over port 4443 is unavailable or blocked. Here's how it works: UDP: Goofy Guineapig can establish communication using the User Datagram Protocol (UDP). This protocol is connectionless and offers faster transmission speeds but lacks reliability and guaranteed delivery. KCP: Goofy Guineapig can also use the KCP (Kylix Crypto Protocol) for communication. KCP is a fast and reliable protocol designed for low-latency, high-throughput applications. It offers features like congestion control and packet loss recovery, making it suitable for scenarios where network conditions are unpredictable. Direct Socket Communications: Goofy Guineapig has the capability to perform direct socket communications. This allows it to establish a raw connection with a Command and Control (C2) server without relying on established protocols like UDP or KCP. The specific details of these direct socket communications, such as the ports used, are likely determined by the embedded configuration string. The choice of fallback channel is likely determined dynamically based on factors like network availability, latency, and the specific configuration of the malware. Let me know if you have any other questions about Goofy Guineapig or malware analysis in general.

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 establishes communication with its Command and Control (C2) server over the non-standard HTTPS port 4443. This deviates from the standard HTTPS port (443) and is likely done to evade detection by security tools that might be monitoring common ports.

MITRE Technique

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

More info: https://attack.mitre.org/techniques/T1571/

Indicators

No indicators found.