OpenSnitch Security Review

By: Tanner Hermann

November 29th, 2018

Table of Contents

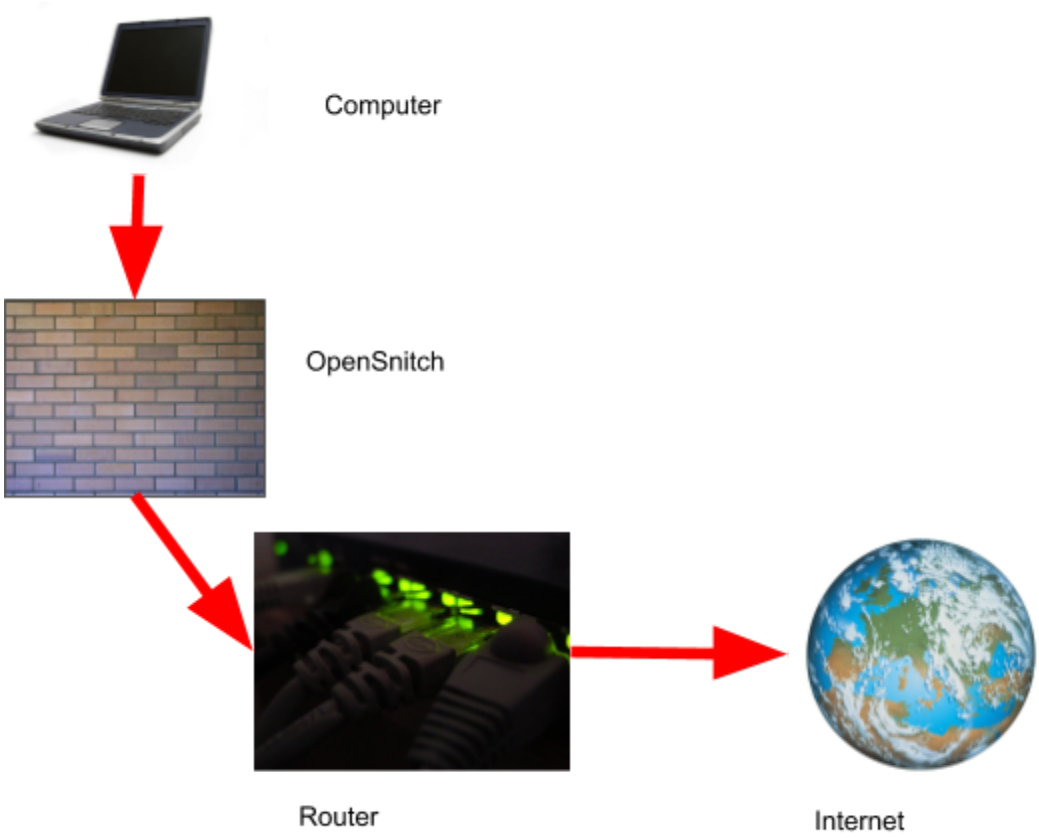
Introduction

Materials & Methods

This security audit focuses on examining the code and functionality of OpenSnitch to find security problems. The goal is to focus on potential security issues while also taking an in-depth look at the code. Investigation and analysis was performed on a Kali VM with the use of Wireshark to monitor network traffic.

About OpenSnitch

OpenSnitch is a linux port of the popular Little Snitch application designed for MAC/OSX. Designed by Simone 'evilsocket' Margaritelli, the application is a host-based, open-source firewall that allows users to track and manage the outgoing connections of applications. OpenSnitch uses NFQUEUE to intercept IP packets and display them to the user. The user has the ability to allow or drop IP packets passing through OpenSnitch. Users have the ability to permanently block an application from sending outgoing data by setting rules in JSON files. The application is written in Python and Go programming languages.



The figure above shows an outline of the path that a packet takes on a system with OpenSnitch. The packet would start on an application on the computer and be caught on OpenSnitch. If the user chose to deny the permissions of the application, the packet would be dropped and not continue. If the permissions were granted by the user, then the packet would continue as normal.

All work was done by Tanner Hermann between November 17th, 2018 and November 27th, 2018.

Firewall Effectiveness

OpenSnitch claims to allow or deny all outgoing network packets at the discretion of the user. In order to test the truth of this, Wireshark was ran at the same time as OpenSnitch. While the system was idle, Wireshark packet capture did not catch any outgoing packets that OpenSnitch did not notify the user of. After letting the system idle, multiple applications were ran with permission from OpenSnitch to send outgoing packets. After examining Wireshark packet data, OpenSnitch appeared to have notified the user of all outgoing network traffic.

OpenSnitch was then tested by denying applications permission to send outgoing packets. For the first trial, Mozilla Firefox was denied permission to connect. After being unable to connect to bing and google, the Wireshark packet capture was analyzed. Wireshark did not show any packets being sent and OpenSnitch showed the packets being denied and dropped. After repeated trials using Firefox, Wireshark and OpenSnitch yielded the same results. The results from Wireshark show that the OpenSnitch firewall is secure and does not allow outbound packets to be sent from an application if the user has not given specific permission.

Packages

The source code of OpenSnitch provided the packages used by the application. The packages found in the source code that are not standard Go libraries include:

- **fsnotify**
- **Gopacket**
- **FTrace**
- **gRPC**
- **Context**

All of these packages are used for the Go programming language. Fsnotify is a package that allows for file-system notifications. It provides functionality for watching the OS for events. Gopacket allows for packet interpretation. Gopacket decodes packet data so that the information can be accessed and processed within Go. FTrace allows for tracking kernel events and system calls within Go. FTrace gives the functionality that allows OpenSnitch to track which process is creating a connection. gRPC is an open source remote procedure call system that allows multiple features such as authentication and flow control. The Context package defines a context data type that carries data about deadlines and cancellation signals. Go 1.7 and later includes Context in the standard library.

None of these packages provide code enabling network activity. Thus, they do not appear to provide a channel for user-data to be obtained. Therefore none of these packages appear to weaken the security of OpenSnitch.

Make Rules Python Script

OpenSnitch comes with a Python script that generates JSON rule files based on multiple lists containing malicious, advertisement, or tracking sites. The script is named make_ads_rules.py and is not run until the user decides to execute it themselves. Basic functionality includes iterating through each site and creating a JSON rule file to prevent connections to the destination host. The links to the lists are:

- https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts
- https://mirror1.malwaredomains.com/files/justdomains
- http://sysctl.org/cameleon/hosts
- https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist
- https://s3.amazonaws.com/lists.disconnect.me/simple_tracking.txt
- https://s3.amazonaws.com/lists.disconnect.me/simple_ad.txt
- https://hosts-file.net/ad_servers.txt

In order for the script to generate the rules file a connection to the sites must be established. The sites do not contain any malicious content or security issues at the current time. While it can be assumed that the developer trusts these sites, it is possible for the site to change in the future. A site could potentially change into a malicious or harmful connection at a later date. For this reason this script must be marked as a minor security flaw.

Code Documentation

After reviewing the code of OpenSnitch, it is apparent that it is poorly documented and large parts are difficult to understand. The developer of OpenSnitch provided hardly any documentation at all. The following is an attempt at an in-depth code documentation to provide clarity as to the purpose of important functions and files in the daemon.

daemon/conman/connection.go

This file contains functions that detect and analyze outgoing connections. There are parse functions that store packet data to variables. For each new connection that is made information about protocol, source IP, source port, destination IP, destination port, destination host, and packet type are stored.

daemon/dns/track.go

The track.go file uses functions to obtain the packet information of connections. Packets are imported into the functions in the file and DNS information is collected.

daemon/log/log.go

This file has functions for keeping track of all errors that OpenSnitch encounters. Error information is exported with color coding and time of error.

daemon/netfilter/

The netfilter folder contains files that enable the functionality of queuing packets. When a new connection is attempted, the packet is placed at the end of the queue, assigned an ID, and data about the packet including packet size is stored.

daemon/netstat/

The netstat folder contains three files that allow information to be collected about each connection. The entry.go file contains a function to store information such as source IP and destination IP in a struct named Entry. The parse.go file contains numerous functions for conversion between binary, decimal, hexadecimal, and IP. It also contains the Parse() function that allows for appending new network statistic entries into a text file. The find.go file contains a FindEntry() function that returns an entry after searching based on information contained in the Entry struct.

daemon/procmon/

The procmon folder contains files that enable OpenSnitch to detect when an application attempts a new connection. A watcher scans for new application connections and information including the process path and PID are stored to variables.

daemon/rule/

The rule folder contains files with functions to check new connections with existing rules. JSON files from the rules path are checked against new connections in order to determine if a connection should be allowed.

daemon/statistics/

The statistics folder contains two files named event.go and stats.go. These two files work together to create the statistics that are displayed on the user interface. event.go creates a struct named Event that contains variables about a connections rules and time. A function named NewEvent() is used to populate the variables of an Event struct. The stats.go file works to display the list of connections that have been made as well as start time, packets dropped, packets accepted, and the number of connections. Functions such as OnIgnored() and OnDNSResponse() provide counters for the variables that keep track of the number

of connections, packets dropped, etc. Other functions such as serializeEvents() and Serialize() work to organize the variables in a user-friendly order.

daemon/ui/

Files in the UI folder allow for functionality of the user interface. Information from all other files in OpenSnitch are sent to the UI files that allow the user to see what OpenSnitch is doing. Most of the information comes from the daemon/statistics/ files.

daemon/firewall/rules.go:

This file contains 4 functions that act to setup the firewall rules of OpenSnitch. The four functions are RunRule(), QueueDNSResponses(), QueueConnections(), and DropMarked().

- RunRule():
    - This function sets rules in the firewall about what connections have been approved by the user. The rules are then implemented with IPtables.
- QueueDNSResponses():
    - This function calls RunRule() with the specified rules.
- QueueConnections():
    - This function queues DNS packets through the use of NFQUEUE. The function outputs this command: "-t mangle -m conntrack --ctstate NEW -j NFQUEUE --queue-num 0 --queue-bypass". The "--queue-bypass" flag allows for packets to automatically be accepted and skip the queue if a connection has already been approved by the user.
- DropMarked():

- If this function is called then a packet that has been rejected will be dropped and not sent through. This function outputs this command: “-m mark --mark 101285 -j DROP”

Conclusion & Recommendations

After conducting an in-depth code review and security audit of OpenSnitch it can be concluded that it is a secure application that can be trusted. During the audit the only security issue that arose was the make_ads_rules.py issue that was discussed. Overall, OpenSnitch is an application that can be recommended for users who wish to control outbound data from a Linux machine.

Recommended Improvements

A simple fix for the make_ads_rules.py security issue would be to collect all lists from the sites already included and store them into a text file. The script could then create JSON rule files locally without making a connection. However, this would require the text file to be updated regularly with updated harmful connection lists.

Otherwise, the only improvements that can be suggested for OpenSnitch include UI changes to create a more user-friendly application. While the UI is simple and straightforward, there are a few minor annoyances. When using OpenSnitch the application window stays on top at all times. It can be resized into a small bar but it will still be on top of all applications. Another suggested improvement is to allow for the rule editing within the application window. Creating new JSON files is not a very user friendly approach. The average user would not have the knowledge to set up their own rules.

Sources

https://github.com/evilsocket/opensnitch

https://www.opensnitch.io/

https://godoc.org/github.com/google/gopacket

https://godoc.org/google.golang.org/grpc

https://godoc.org/golang.org/x/net/context

https://godoc.org/github.com/fsnotify/fsnotify

https://fsnotify.org

https://github.com/fsnotify/fsnotify

https://github.com/evilsocket/ftrace

https://grpc.io