MalOrbot
An Android malware to bridge mixes in Tor

Evangelos Mitakidis*, Dimitrios Taketzis*, Alexandros Fakis*, Georgios Kambourakis*
*Information and Communication Systems Department
University of the Aegean, Greece

***Abstract*—Abstract goes here**

I. Introduction

Internet was not created and designed with anonymity as its main idea. On the contrary, if two parties wanted to communicate it was mandatory to provide their real source addresses to exchange messages. Even TCP/IP protocol, which is the foundation of modern internet connections, requires the true source and destination addresses to be able to transfer packets between hosts. This was not much of an issue until internet population grew exponentially. Only then, we discovered the need for security, privacy and anonymity.

One of the most widely used anonymous communication systems is Tor network. It constists of a group of volunteer-operated servers. It's main purpose is to allow people to improve their anonymity, privacy and security over the Internet. Tor's users employ this network by connecting through a series of virtual tunnels and not by making a direct connection. This implementation allows the exchange of information over public networks with privacy and anonymity. In addition, Tor can be used as a censorship circumvention tool. It does that, by allowing its users to connect to otherwise blocked destinations or have access to otherwise blocked content. Journalists trying to communicate safely with whistleblowers or dissidents, activist groups that endorse mainting civil liberties online and people living in countries that restrict freedom of information and access to websites take the main benefit of its use. And that has to be protected at all costs. On the other hand, like every other tool, anonymity through Tor is also being used for malicious purposes. Drugs, guns, and pedophile material is exchanged everyday between Tor users at special websites only accessible through Tor, hosted on servers which offer Tor "hidden services". This has led goverments and security researchers to try to find ways to deanonymize Tor traffic.

Many researchers are trying various attacks on Tor protocol to discover ways to deanonymize its users.

In this paper, we show a new stealthy approach to attack Tor users, which enables attackers to deanonymize users instantly. In this approach, the attacker is in control of a specially configured exit node. In addition, using social engineering techniques, a user-side android malware is planted on the tor user's smart device. Its function, among others, is to inject user identifying information on every request that goes through the Tor network. When the malicious exit node receives the request with the injected information, there will be instant user deanonymization.

II. Theoretical Part

Here we will describe in theory how Tor works and our approach

III. Practical Part

Here we will describe exactly how each component works in our implementation

A. *MalOrbot - user side malware*

Malorbot is the user side Android malware app we developed, which is installed on the victims smart device with specific properties. Our app has three main functions. The first function is modifying the settings of the apps that are used to access Tor network (Orbot and Orweb). The second is obtaining the required information to deanonymize the victim and the last serves as a MitM HTTP proxy in order to inject personal information into the victims requested URLs.

The first property that we set for the malware app was to start running initially after installation and after every boot of the smart device. After installation, malorbot app masquerades as Adobe Flash v2.0, which is a very widely known software and will not attract the attention of the victim. It, also, deletes Superuser and SuperSU app to bypass permission request dialogs. These dialogs are shown to the user by the use of toast messages, every time an application requests root permissions. This kind of toast messages can attract victims attention, cause suspicion that could lead to the uninstallation of the app. Additionally, we focused on the stealthy operation of the app to avoid the detection of antimalware software. To accomplish this, we minimized the number of connections outside the Tor network. More specifically, there is only one request to a public web service to discover the public IP, while the rest of the communication remains within the Tor network.

The first function of our malware app is focusing on modifying the environment settings to accomplish the attack. In particular, it modifies Orbot configuration settings to always use the attackers exit node, during the creation of Tor circuit. After that, it prohibits any further modifications. This forces Orbot to use our specially configured exit node. Additionally, it changes the proxy settings of the Orweb app, which is the most commonly used mobile browser to access Tor. The forged settings force the victims traffic to be proxied through our Malorbot HTTP proxy app.

This allows us to perform a Man-in-the-Middle attack during HTTP proxy operation for outgoing and incoming communications.

The second function gathers all required information to deanonymize Tor users. To accomplish this, the malware app makes a connection to a public service (e.g. www.myip.com) to get the public IP of the victim. Since this is the only outside-of-tor connection, stealth operation is guaranteed.

The last function of the malware is that of an internal HTTP proxy. It acts as MitM between the Orweb and Orbot app, in order to perform the intended injection attack on HTTP connections. This injection adds the public IP of the victim, obtained from the previous function, into the victims requested URL. E.g., assuming the URL is http://www.in.gr before injection, it will be modified to http://www.in.gr/&sourceip=65.65.65.65 and will be forwarded to the Orbot app. When HTTPS connections are requested we first perform a downgrade attack and then the previously described injection attack.

B. *Exit node - configuration*

On the exit node side, we installed Ubuntu Linux and then the following procedures took place:

A. Installation and configuration of the Tor software.

B. Implementation of an HTTP Proxy to act as a MitM between the exit node and the target server.

C. Implementation of a logging system for the sniffed traffic with special tagging for the infected clients

D. Redirection of the traffic coming from the Tor exit node software to the HTTP Proxy.

The first procedure involved the installation of the official Tor software to our Ubuntu server. We then proceeded with a basic configuration of the torrc file, so we can operate as a trusted exit node. In particular, we configured torrc allow our exit node to route HTTP and HTTPS traffic (ExitPolicy accept *:80 and ExitPolicy accept *:443). We also adjusted bandwidth that we will make available to Tor and finally provided a name for our exit node.

The second procedure involved writing a basic HTTP Proxy in Python that listens on a different port on the same box as Tor. Its goal is to sniff HTTP traffic while, at the same time, try to match a specific pattern in the sniffed HTTP request headers.

The third procedure involved proper handling of the HTTP request inside the proxy. In particular, If there is a match then a log is created with a special tag along with the request information properly formatted. Then, the pattern is removed from the HTTP request header and the request is forwarded to the target web server. The http response from the target webserver is forwarded to the Tor network with no further manipulation.

The final procedure involved the use of an iptables command to redirect all traffic that has port 80 destination to our local MitM HTTP Proxy so we can start sniffing and logging/tagging.

Finally, we coded a very simple bash script tool to filter the sniffed traffic and show the tagged-infected client information on screen.

IV. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to $\LaTeX$*, 3rd ed. Harlow, England: Addison-Wesley, 1999.