Analysis of the Sophos Update Mechanism

Marco Gonzalez, Jeffrey Herron, Zita Hiu Kit Wong, Ge (Grace) Yu
marcog@ece.ubc.ca, jherron@interchange.ubc.ca, zitawong@interchange.ubc.ca, graceyyy116@yahoo.com

***Abstract*-The Sophos Anti-Virus updating mechanism has been found to be insecure due to a lack of authentication leading to a client side weakness to various spoofing attacks. In this paper, we demonstrate that it is vulnerable to three attacks, a man-in-the-middle attack, a packet injection attack, and a DNS poisoning attack. Through these attacks we have found that it is possible to fake up-to-date status possibly leaving end systems unknowingly more vulnerable to various types of malware. Each of these attacks succeeded in denying updates and forging up-to-date status on both virtualized and physical machines.**

I. INTRODUCTION

In today's networking environment, a PC's security relies heavily on consistent updates from trusted software vendors. As such, ensuring that these update mechanisms are secure is just as important as the security that the updates provide once installed. However, despite this fact, many software companies tend to treat distribution of their updates as a secondary task.

Securing updates is an important question that is being addressed by a number of different companies in the software market. For instance, Microsoft Windows Update uses SSL (Secure Socket Layer) encryption and hashes single updates and certificates for each update. SSL is used to encrypt the information that is transferred, to prevent hackers from tampering with information that is being transferred, and to ensure that the Windows Update agent is transferring data from an authorized Microsoft server [1]. Another major player in the anti-virus market is McAfee Anti-Virus, which uses authenticated binaries, but it does not authenticate the connection [2] so it too may be vulnerable to many of the same attacks described in this report.

Upon examining Sophos Anti-Virus, the anti-virus software provided for free to UBC students by the UBC IT Department, we have identified a severe flaw in the update mechanism. When a user requests an update, the Sophos client uses non-authenticated HTTP packets to check if there is an update. Because of this, it is a simple matter to deny updates to clients through various attacks. Through these attacks, we also have found that it is possible to fake up-to-date status on the client. This would deceive users into believing that his/her PC is safe from malware. Another critical implication is that it may be possible to send corrupted data, which would leave end systems even more vulnerable to malware. While we did not have the time to pursue this further and actually corrupt the data, it is certainly an area for further research. However, denying service updates for Sophos Anti-Virus leads to less secure systems on the network, and thus compromises the network as a whole. This paper discusses in detail the three attacks that we have made, the man-in-the-middle attack, a packet injection attack, and a DNS poisoning attack, to prove the vulnerabilities of the Sophos Anti-Virus updating mechanism.

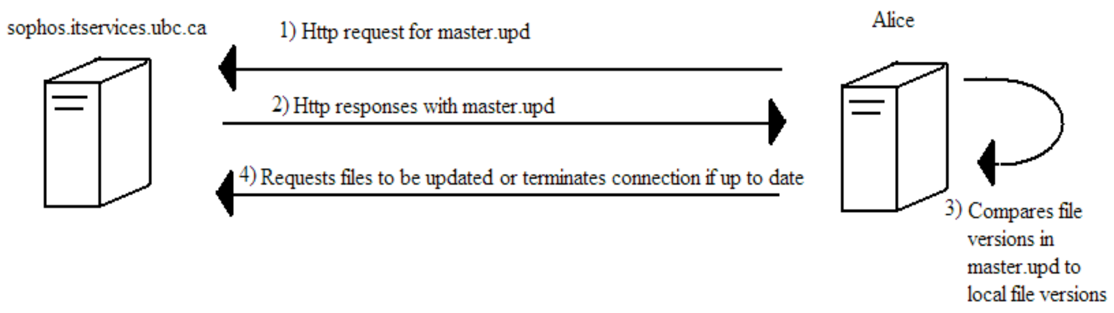
sophos.itservices.ubc.ca
Alice
1) Http request for master.upd
2) Http responses with master.upd
4) Requests files to be updated or terminates connection if up to date
3) Compares file versions in master.upd to local file versions

Figure 1: Sophos Update Process

II. ANALYSIS

A. Sophos Update Mechanism

First we needed to establish how Sophos Anti-Virus actually updates the client. We did so by using the Wireshark packet sniffer to examine the entire update process. We took samples over a week period from several different machines. From this sample set we determined that the update mechanism is done in four steps as shown in Figure 1. First the Sophos client sends a HTTP GET request for master.upd to the UBC Sophos update server (in this case, sophos.itservices.ubc.ca). The Sophos update server responds with master.upd which contains the current file versions hosted by the server.
The client then checks the file versions hosted by the server against it's own file versions and then will request files as necessary. If no files need to be updated, the client will simply terminate the connection.

B. Man-in-the-Middle Attack

We decided to first implement a man-in-the-middle attack due to the level of control the attacker has on the communication between the two parties. The basis of the attack is illustrated below in Figure 2. However, one of the difficulties in a man-in-the-middle attack is determining how exactly the attacker (Eve) will become the man-in-the-middle (MitM). Due to the fact that this is a project on Sophos update security and not how to setup a MitM attack, we chose to forgo the details and simply manually set up Eve as a proxy for all traffic from the client (Alice). In order to accomplish this, we configured Webscarab running on Eve to act as the proxy server and had it listening on port 8008. From here, we set the proxy settings on Sophos update to use Eve's IP port 8008 as the proxy server. We then set out to collect samples of the master.upd files transferred between Alice and the Sophos update server. After saving the initial conversation between the client and update server, we then modified any response to a later request to return instead the original master.upd from the first conversation. In this manner we successfully utilized a MitM attack to delay updates to client for over a week, however it could have continued indefinitely.

C. Packet Injection Attack

Another vulnerability of the update mechanism in Sophos is that any entity can listen (once again, Eve) to the packets between the server. Once a

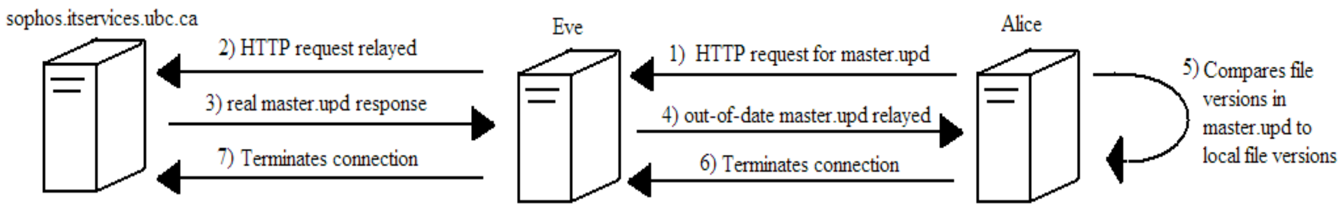
sophos.itservices.ubc.ca
Eve
Alice
2) HTTP request relayed
3) real master.upd response
7) Terminates connection
1) HTTP request for master.upd
4) out-of-date master.upd relayed
6) Terminates connection
5) Compares file versions in master.upd to local file versions

Figure 2: Man-in-the-Middle Attack

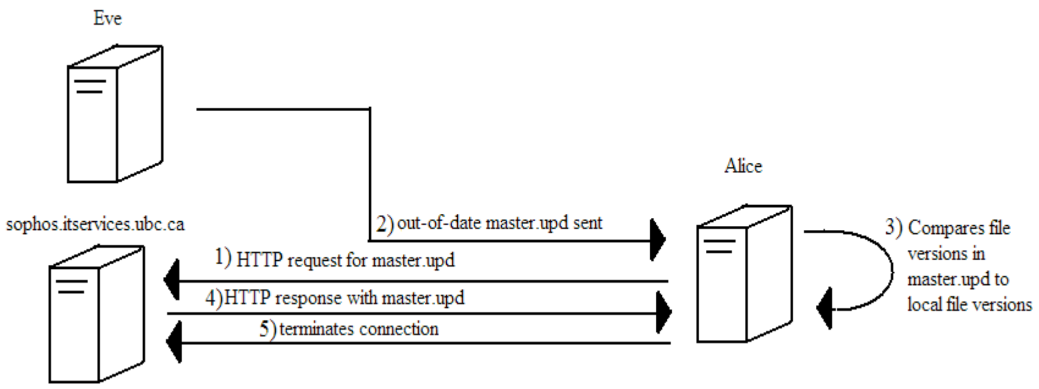
Eve
sophos.itservices.ubc.ca
Alice
2) out-of-date master.upd sent
1) HTTP request for master.upd
4)HTTP response with master.upd
5) terminates connection
3) Compares file versions in master.upd to local file versions

Figure 3: Injection attack

update request is sent, Eve can inject a spoofed response from the server containing an out of date master.upd. This attack is further outlined in Figure 3. In order to implement this attack we used a custom packet generation program, Wireshark and Webscarab.

We programmed our packet generator in Java using the Jpcap library. Our program would create TCP packets with the valid MAC address of Eve in order to be properly routed, but with the source IP address and port number of the Sophos server. The payload contained the byte-stream corresponding to a HTTP packet header from the Sophos server and the out of date master.upd, both of which were captured from previous exchanges between the client and server. However, three fields need to be set on a per-connection basis. These consist of the client side TCP port, the sequence number (SEQ) and the acknowledge number (ACK). The SEQ and ACK numbers increment throughout the packet exchange to ensure packet order and check for packet loss, whereas the port number is semi-randomly assigned when the TCP connection is initialized.

Webscarab was used as a proxy in order to introduce a delay between the client and the server. The difference between this attack and the MitM attack is that we did not use Webscarab to alter any portion of the packets. We needed this delay in order to compensate for manually altering the fields in the packets that we generate so that they will be accepted as valid packets. The fields we needed to alter on a per-connection basis were the client-side port number and the SEQ/ACK numbers. By listening in on the previous packets of the exchange using Wireshark, we were able to assign the appropriate values to those fields to have our program create a valid spoofed packet. As long as this spoofed packet arrives before the response from the Sophos server, it is accepted as the actual response.

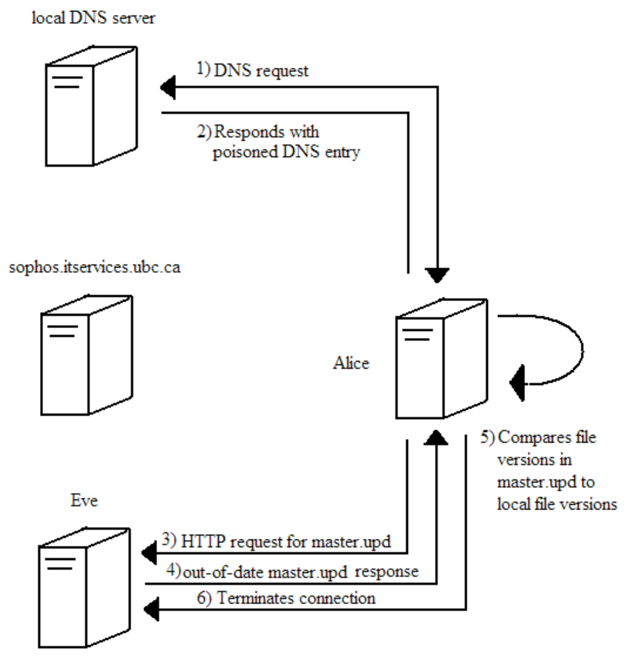
local DNS server
1) DNS request
2) Responds with poisoned DNS entry
sophos.itservices.ubc.ca
Alice
5) Compares file versions in master.upd to local file versions
Eve
3) HTTP request for master.upd
4) out-of-date master.upd response
6) Terminates connection

Figure 4: DNS Poisoning Attack

D. DNS Poisoning Attack

DNS poisoning consists of modifying or providing false data for entries in a DNS server or cache, so that a DNS query for a domain returns a false IP address [3]. This false IP address could be that of an attacker's server.

We configured three different machines. The first of these was Alice who is a regular Sophos client. Eve is a fake Sophos server hosting an out of date master.upd. Lastly, we used the host machine to act as the local DNS server for Alice. This topology and details of the attack are shown in Figure 4 for further clarification. Notice how the authentic Sophos server is not actually involved in the entire update process.

First, we ran a Windows DNS server program called Simple DNS Plus on the host machine to provide the DNS service to Alice. In the program we created a simulated poisoned entry which specified the IP address for sophos.itservices.ubc.ca as being Eve's IP address. So when Alice requests an update from Sophos server, a DNS query will first be sent to the DNS server to resolve the Sophos server hostname to an IP address. The local DNS server will then respond with the poisoned DNS entry by giving the IP address of Eve instead of the IP address for the Sophos server.

At this point, Alice will again proceed to make her HTTP request for master.upd. In order to mimic the Sophos server, we had the Cherokee Web Server running on Eve to respond to HTTP requests. We then hosted an out of date master.upd in the appropriate directory (which is known to be ESXP/master.upd by packet sniffing with Wireshark). When Alice requests master.upd, Eve will simply respond through a standard HTTP response with the out of date file which appears to be valid. As such, Alice has once again been spoofed into thinking she is up to date.

III. DISCUSSION

A. Feasibility of the Attacks

In order to prove that Sophos update is insecure, we must also discuss the feasibility of such attacks being used in a non-controlled environment.

MitM attacks have been a consistent problem for the security field. With the current Sophos update implementation, the attack itself is trivial. Furthermore, there are many circumstances in which a MitM attack would be quite straightforward. One example of this is attackers setting up their own wireless access points. From there it is a simple matter to alter and listen to the packets as desired.

The packet injection attack is perhaps more worrisome because it can be implemented by any member of the network that can sniff the packets between the clients and the server. In order for the attack to be successful, the only requirement is that the delay between the client and attacker must be smaller than the delay between the client and server. Furthermore, if a denial of service attack was mounted against the Sophos update server, the chance of success would rise significantly.

The vulnerability of clients to a DNS poisoning attack is twofold. Firstly, the DNS server cache is only as secure as the the software running on the server, so on a less secure DNS server it may be possible to directly alter the server's DNS records. This would produce results similar to the DNS attack carried out in the analysis portion of this report. The second vulnerability lies in the DNS protocol itself. Like the packet injection attack, Eve could listen for the DNS requests for the IP address of the Sophos server, and then spoof DNS responses. This is possible because DNS uses UDP, uses predictable algorithms and has no authentication. In this case, only the local DNS caches on the hosts would be poisoned, but the end result would be the same.

B. Implications

The fact that Sophos update is so insecure could potentially have drastic effects on the networks that utilize them as the major anti-virus tool. Without access to updates, end systems may have vulnerabilities to malware. Because of this, new malware could spread and flourish in a network where updates were unavailable, and could remain undetected for a much longer amount of time. At that point the assets at risk is the usability and data stored on any end system blocked from receiving updates. This vulnerability is probably due to a the Sophos update designers not properly questioning their assumptions of the system. If they didn't consider that there may be computers on the network that actively want to deny updates, then there is no reason to have the system run any differently than it does today.

None of these attacks would be possible if Sophos update server set up a properly authenticated and encrypted channel. As it stands however, the systems protected by Sophos are vulnerable to being denied updates.

C. Future Work

There are several different directions for future work on this topic. The first of these is to work on a scheme to secure this connection from the previously mentioned attacks. This could possibly include server authentication, time-stamping, and/or channel encryption. This would essentially increase the overall amount of defense in depth as per the principles of designing secure systems. However, depending on the solution it may require additional infrastructure and code, such as key distribution methods, which would slow down the overall updating process.

The second direction for future work could focus on providing corrupted data during the Sophos update process. This may be a trivial matter in the DNS poisoning attack and the MitM attacks, but may be considerably more difficult for the packet injection attack.

REFERENCES

[1] Microsoft, “Windows Update Explained,” September 2008. [Online] Available: http://www.scribd.com/doc/6331827/Windows-Update-Explained [Accessed Nov. 10, 2008].

[2] A. Bellissimo, J. Burgess, and K. Fu, “Secure Software Updates: Disappointments and New Challenges”, USENIX Hot Topics in Security Workshop, July 2006.

[3] T. Olzak, “DNS Cache Poisoning: Definition and Prevention,” unpublished.