A Zero-day Attach Exploiting a Yahoo Messenger Vulnerability

Monther Aldwairi
Jordan University of Science and Technology
munzer@just.edu.jo

Haitham Noman
New York Institute of Technology
haitham.online@yahoo.com

Abstract— In computers security terms, vulnerability is a flaw in the computer system due to a bug or weakness in software, security policy and/or overall system configuration. Vulnerabilities are recognized if they are exploited by attackers using a tool to allow system violation. Unfortunately, there is no one standard for vulnerability reporting to date, and the debate continues between supporters of full discloser, non-discloser and responsible disclosure. We follow the responsible disclosure definition outlined by Shepherd, by reporting the issue to the vendor first and give a month to the vendor to establish a meaningful connection or provide a suitable fix [1]. Otherwise, go public with full disclosure. In this paper we discuss techniques to exploit a weakness in Yahoo messenger client. We successfully build a Trojan, called Caruso, which basically allows the attacker to gain access to the victim's Yahoo account without the need to crack the password.

Index Terms—*Vulnerability disclosure, vulnerabilities, exploits, Trojans, Yahoo messenger.*

1 INTRODUCTION

With the advent of information technology and the integration of computer systems into all aspects of our lives, comes a great challenge to secure information. The insecurity of computer systems stems from the lack of good design, careful implementation, proper testing, accountability and quick response to detected flaws. The lack of timely response raises a very important question about who is responsible for finding vulnerabilities and what are the proper methods for reporting them. Vulnerability disclosure becomes more and more important and the lack of a unified standard puts critical information at risk.

Voelker et al. stated that over 100 vulnerabilities were reported per week and over 7,400 new vulnerabilities disclosed in 2008 [2]. While the debate is heated on the best way for vulnerability disclosure, system security researchers are lost in their aim to protect users between the opposing arguments and law suits. Full and public disclosure (FD), in one hand is to publically expose the vulnerability with the exploit without alerting the vendor first. The rationale here is to announce the vulnerabilities as soon as they are discovered to allow the users to protect themselves by removing the software or disabling certain features before a wide spread of an attack or virus. To further argue the case of FD, the enthusiasts claim that this helps influence the vendors to develop patches faster. Finally, the researcher gets immediate credit for discovering the vulnerability. The argument against full discloser is mainly that, it allows the risk of wide spread attacks before the vendor have the appropriate time to patch the flaws. In addition, it is argued that public disclosure may not inspire vendors to patch their applications. Limited disclosure, on other hand does not include exploit code but this does not hinder the expert hackers from developing one [1].

Under the non-disclosure (ND) policy pushed by some vendors, the security researcher must keep the discovered vulnerability secret. This is based on the assumption that a good portion of vulnerabilities remain undiscovered and the fact that this allows the vendor more time to find vulnerabilities and update their software. The argument against it is that, it leaves the system exposed and encourages malicious intent where vulnerabilities are exploited by the discoverers over and again [3].

Responsible disclosure (RD) on the other hand, tries to minimize the risk to the end users by keeping the reporting to a trusted group of individuals until the fix is released. In this case, it is hard to define the trusted individuals, and what is a reasonable length of time. It is up to the vendor to decide what and who is reasonable and they are less motivated to patch the flawed software. In addition, the original researchers who discovered the vulnerability lose their credits in case of rediscovery as well as any possible financial compensation. Immediate and instant disclosure is being favored by discoverers as the best way to speedup patching by vendors assuming hackers might already know and use those undisclosed vulnerabilities [4][5].

The Computer Emergency Response Team / Coordination Center (CERT / CC) works as an independent mediator between security researchers and vendors [6]. CERT discloses the vulnerability after 45 days whether a patch was developed or not. Moreover, a new consensus, called Coordinated Vulnerability Disclosure (CVD), is growing among discoverers and vendors. CVD is an extension of RD where the finders or discoverers report directly to the vendor and national CERT. The vendors diagnose, develop patches and coordinate closely with the finder. After patches are developed the vendor recognizes the finder in later advisories. If an attack went wild before the patches are developed, both the vendor and the finder coordinate to provide the best public disclosure of that vulnerability

[7].

This paper sheds the light on previously unknown vulnerability in Yahoo messenger and develops a new Trojan called "Caruso" which exploits the weakness to allow the attacker full access to the victim's account. The vulnerability was disclosed to Yahoo under RD on January of 2012 and due to the lack of response this paper acts as a responsible discloser to protect customers. The rest of this paper is organized as follows. Section 2 explains the Yahoo authentication process in details and presents the vulnerability. Section 3 introduces Caruso Trojan and explains it functionality. Section 4 briefly analyzes Caruso and presents its weaknesses and finally, we conclude in Section 5 by presenting proposed remedies.

2 The Yahoo messenger authentication vulnerability

Older versions of Yahoo messenger up until 6.x stored the encrypted password as special token in the registry under HKEY_CURRENT_USER\Software\Yahoo\Pager with value called: "EOptions string" value. Hackers figured out how to extract the encrypted password from the registry and subsequently to decrypt it.

Starting with version 7.x, Yahoo changed the located of the encrypted password to: HKEY_CURRENT_USER\Software\Yahoo\Pager\ETS as shown in Figure 1. In addition Yahoo Corp encrypted a token using a key "MBCS sucks + USERNAME" and coded the results with yahoo64 secret Yahoo encoding. Luckily no one discovered the new encryption technique and none are able to decrypt the token. Attackers desperately resorted to key loggers in order to hack their victim's Yahoo accounts.

"SlicK" provided an in depth analysis of the Yahoo authentication in his white paper [8]. First, the Yahoo messenger user enters his username and password and a request is sent for a token. If the user has checked the box labeled "Remember my ID & password", the messenger will retrieve the token from the registry ETS key. Once the connection is verified with the Yahoo login servers the authentication process begins by sending the username. The server sends a challenge string for encrypting the username and password. Now we have the valid token the client makes a request for the value of the cookie and a "crumb". The client encrypts the username and password, and sends back two 24-byte special string with the crumb and challenge. If everything checks out the server sends back a cookie to enable the user to access the email and friends lists. The cookie expires after logout or if it exceeds a predetermined time duration.

The above authentication scheme seems bullet proof, but unfortunately, if someone gets a hold of the encrypted token and manages to access and extract the encrypted password from the users ETS register key and embed into his registry, he will be able to sign into Yahoo messenger with no need to decrypt the password. Simply, as if the attacker have logged in before and chose to save and remember the username and password. In the next section we demonstrate a simple but effective Trojan that has been implemented to extract the token and send it to the attackers email.

3 Caruso Trojan

Caruso Trojan is grabs the encrypted password from the registry and sends it to the hacker without the consent of the victim. Afterward, the hacker then can inject the encrypted string into his registry to enable him to sign in into the victim's account without decrypting the password. Caruso can be delivered through emailing an infected application to the victim. Figure 2 shows the flowchart for Caruso Trojan which works as follows. Once the victim runs the Trojan it copies itself to the folder path C:\Windows\Updator.exe and runs automatically. Next, Caruso will check if the victim is connected to the Internet. If connected to the Internet the Trojan searches the registry for Yahoo messenger's username and password. If found the Trojan sends the username, encrypted password and IP address via email to the attacker as shown in Figure 3. If the user has never clicked "Remember my ID and password" the Trojan calls a windows API which checks the box on behalf of the user. The box is made invisible to insure the user does not pay attention and unchecks the box. If the victim is not connected to the Internet, the Trojan disables the previous check box, waits for a connection to be established and the user to login before it searches the registry. Once the attacker receives the email with the encrypted password string, he embeds it into his registry and signs up as if he has saved his credentials earlier.
Programs and windows in MS Windows are organized into a hierarchy and modifying the Yahoo messenger client is a difficult task. Using MS Spy++ it is very easy to get the ID for parents and child windows. Several ready-made tools such as Windows tools pro enables the attacker to shows and hide windows components as shown in Figure 4. The Figure shows that the "Remember my ID and password" check box is hidden.

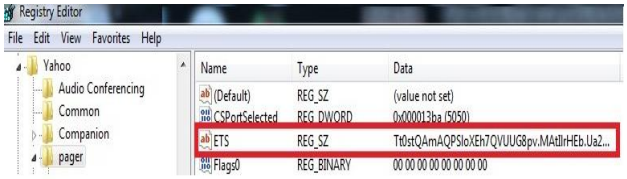

Figure 1. Registry key for Yahoo password

4 Security analysis

Caruso is a proof of concept Trojan which cannot be detected by firewalls because it sends the email through the commonly open HTTP port 80. Second, it is almost undetectable by Antivirus software because it has not been released in the wild, yet. Finally, the code has been packaged with UPX, which is an open source obfuscator that compresses the code, to protect from reverse engineering.

Weaknesses of the Trojan include permissions to write the code to the Windows folder. The attackers email is susceptible

to sniffing by expert security administrators. To demonstrate that sniffing of email traffic by the system admin, we run Wireshark on an infected machine. Figure 5 shows clearly the email of the attacker and message contents.

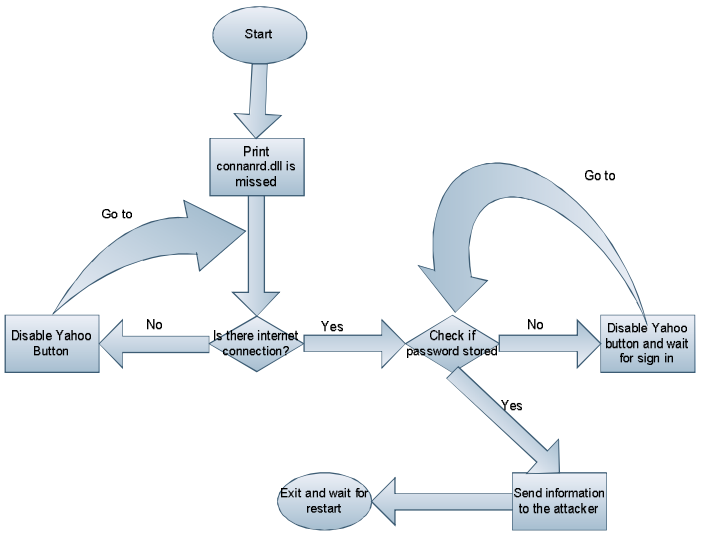

Figure 2. Caruso Trojan flowchart

Figure 3. Password email

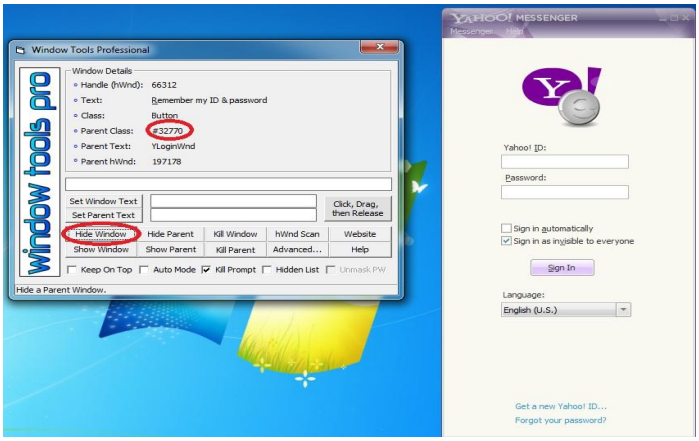

Figure 4. Hiding the check box

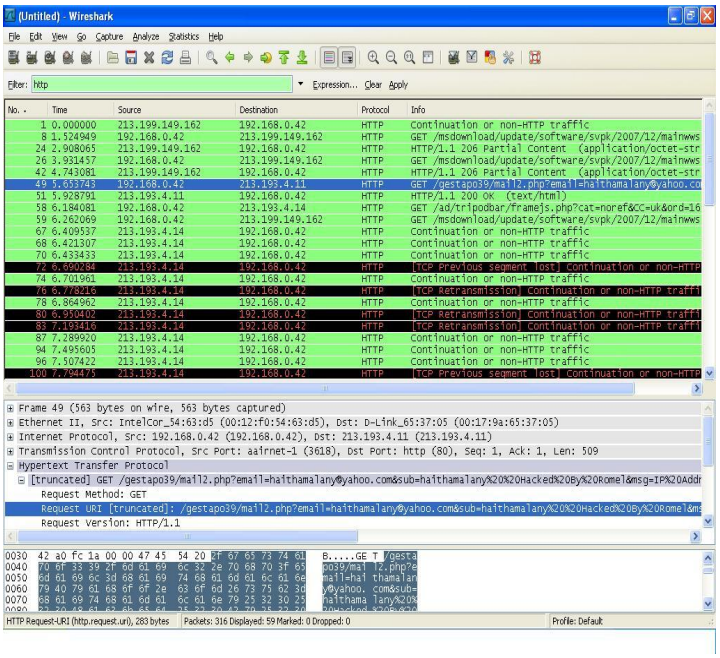

Figure 5. Sniffing attackers email using Wireshark

5 Conclusions and Remedies

In this paper we have developed a zero-day Trojan to exploit previously unknown Yahoo messenger vulnerability. The zero-day attach allowed the hacker unauthorized access to victims Yahoo messenger. We agree that a strong encryption algorithm is essential to protect credentials but the authentication protocol itself could be of the greater importance. Because of a weakness in the Yahoo authentication scheme, it is not necessary to decrypt the password to gain access to the user account. In addition, allowing the modification of running processes in windows opened the door for the Trojan to enable password saving without the users consent.

Saving passwords is never a safe option and future Yahoo messenger versions should use a different technique to store and transmit passwords. Furthermore, antivirus software protects only against known attacks that have been reported and analyzed. Anomaly based virus scanners that sniff packets and reconstruct sessions will gain higher importance in the near future. Finally, there is a dire need for a standard mechanism for vulnerability disclosure to better protect the user's data.

References

[1] S. Shepherd, Vulnerability Disclosure: How do we define Responsible Disclosure?, SANS Institute, (2003).
[2] F. Massacci, S. Neuhaus and V. H. Nguyen, After-Life Vulnerabilities: A Study on Firefox Evolution, Its Vulnerabilities, and Fixes, Lecture Notes in Computer Science: Engineering Secure Software and Systems, Springer Berlin, Heidelberg, (2011) Vol. 6542, pp.195-208.
[3] S. Frei, D. Schatzmann, B. Plattner and B. Trammell. Mod-

eling the Security Ecosystem–The Dynamics of (In)Security. Proceedings of the Workshop on the Economics of Information Security, (2009), 24-25 June, University College London, England.
[4] B. Schneier. The nonsecurity of secrecy. Communications of the ACM, 47, 10 (2004).
[5] J. T. Chambers and J. W. Thompson. Niac vulnerability disclosure framework. Department of Homeland Security, (2004).
[6] US Computer Emergency Readiness Team. [last access 5/12/2012]. http://www.us-cert.gov/
[6] Microsoft response security center. Coordinated Vulnerability Disclosure. [last access 5/12/2012]. http://www.microsoft.com/security/msrc/report/disclosure.aspx
[7] SlicK, In-Depth Analysis of Yahoo! Authentication Schemes, RSTzone.org. [last access 5/12/2012].
http://www.xssed.com/article/14/Paper_In-Depth_Analysis_of_Yahoo_Authentication_Schemes/