AUTOMATED TESTING OF PRIVILEGE ESCALATION IN WEB APPLICATIONS

ORY SEGAL, DIRECTOR OF SECURITY RESEARCH
WATCHFIRE

A whitepaper from Watchfire

TABLE OF CONTENTS

PREFACE **1**
PRIVILEGE ESCALATION IN WEB APPLICATIONS **1**
The challenge of automating Privilege Escalation testing *3*
PRIVILEGE ESCALATION TESTING WITH WATCHFIRE APPSCAN 7.0 ..... **3**
CONCLUSION **5**
REFERENCES **6**
ABOUT WATCHFIRE **6**

PREFACE

"Then Isaac said to Jacob, 'Come near so I can touch you, my son, to know whether you really are my son Esau or not.' Jacob went close to his father Isaac, who touched him and said, 'The voice is the voice of Jacob, but the hands are the hands of Esau.' He did not recognize him, for his hands were hairy like those of his brother Esau; so he blessed him. 'Are you really my son Esau?' he asked. 'I am,' he replied." (Genesis 27:21-24)

This quote from the Bible documents what is quite possibly the earliest ever attempt at "Privilege Escalation." In more contemporary times, Privilege Escalation means exploiting a bug in an application in order to procure resources which are intended to be inaccessible, or accessible only to users with higher access privileges.

Privilege Escalation vulnerabilities exist in modern times too, and in this article we will take a closer look at Privilege Escalation in web applications, and how to automate the process of testing for them.

PRIVILEGE ESCALATION IN WEB APPLICATIONS

There are two types of Privilege Escalation:

- **Horizontal Privilege Escalation** occurs when a malicious user attempts to access resources and functions that belong to peer users, who have similar access permissions.

- **Vertical Privilege Escalation** occurs when a malicious user attempts to access resources and functions that belong to a user with higher privileges, such as application or site administrators.

Figure 1: The Two Types of Privilege Escalation

Let's take a closer look at some examples of both types of Privilege Escalation.

Example 1: Data Mining at MySpace.com

This vulnerability was published in the [Full Disclosure] mailing list on June 30th 2006[1]. MySpace.com, an online social networking web site, offers its members the ability to send news bulletins to other MySpace members. When you submit your bulletin a URL is sent to your friends that look similar to this:

http://bulletin.myspace.com/index.cfm?fuseaction=bulletin.read&messageID=[BID]

Where [BID] is an automatically generated numeric bulletin ID.

By simply changing the bulletin ID number, users were able to access the news bulletins of other MySpace members which they had *not* received notification about, and read the contents.

This example is a simple one, yet it demonstrates clearly how users can employ horizontal privilege escalation to access resources that are not intended for them.

Example 2: Gaining administrative privileges on a mailing list administration web interface

This example was taken from a real world web application assessment of an airline web site. The majority of leading airlines today encourage their customers to enroll in a frequent flyer club, which will grant them airline mileage, member perks and bonuses. These frequent flyer clubs usually have a mailing list, which frequent flyer club members can subscribe to in order to receive newsletters and notification about special fares.

During the web application assessment, we noticed the entry page to the frequent flyer mailing list. This part of the application allowed web users to register to the mailing list and manage their mailing list accounts.

After logging-in to the mailing list web interface, users were presented with an HTML page that offered them various account management options. This web page included HTML Forms similar to the following:

```
<FORM METHOD="POST" ACTION="http://www.some.site/scripts/mailing_list.pl?sub=11111&id=123456789">
        <INPUT TYPE="hidden" NAME="check_code" VALUE="55ABCDEFGHIJ55">
        <INPUT TYPE="hidden" NAME="text_mode" VALUE="1">
        <INPUT TYPE="hidden" NAME="login_name" VALUE="someuser@some.site">
        <INPUT TYPE="hidden" NAME="site_desc" VALUE="AIRLINE SITE">
        <INPUT TYPE="hidden" NAME="site" VALUE="AIRLINE">
        <INPUT TYPE="hidden" NAME="list" VALUE="FREQUENT_FLYER">
        <INPUT TYPE="hidden" NAME="login_id" VALUE="12345">
        <INPUT TYPE="hidden" NAME="posting" VALUE="F">
        <INPUT TYPE="hidden" NAME="list_admin" VALUE="F">
        <INPUT TYPE="hidden" NAME="member_type" VALUE="normal">
        <INPUT TYPE="hidden" NAME="list_from" VALUE="mail-list@some.site">
        <INPUT TYPE="hidden" NAME="list_no_search" VALUE="T">
        <INPUT TYPE="hidden" NAME="list_no_archive" VALUE="F">
</FORM>
```

Figure 2: Mailing List Web Interface HTML source

The first thing that caught our attention was the hidden "list_admin" parameter. By simply changing the value of this parameter, from "F" (False), to "T" (True) and submitting the form to the web application, a

user with ordinary privileges was able to gain access to an administrative page including resources and functions that should only be accessible to administrators.

Once we had accessed this administrative page we could harvest usernames, passwords, email addresses, and even send emails to list users as if they came from the frequent flyer club.

THE CHALLENGE OF AUTOMATING PRIVILEGE ESCALATION TESTING

Privilege Escalation issues, are considered[2] by many to belong in the category of "Logical Vulnerabilities"[3]. This category includes vulnerabilities that often cannot be reliably tested by automated web application security scanners. Unlike "Technical Vulnerabilities", which are easier to automate, Logical Vulnerabilities may require knowledge and abilities that only a human user can supply.

Let's take another look at Example 2, above: An automated scanner may easily detect that an HTML form includes a hidden parameter with the string "admin" in its name. The automated scanner can also detect that the value of the parameter could be toggled (e.g. False→True, No→Yes, etc), thus the automated scanner could attempt to toggle the parameter value and escalate the privileges under which it is currently browsing the web application.

So far so good, but how do we automatically *validate* the results of such a request to the web application? How can an automated scanner detect that it is no longer viewing a regular user's page, but that of an administrator?

Many factors contribute to making automated Privilege Escalation testing a significant challenge, but with the right technology, and some smart heuristics, automated scanners can certainly assist humans with the tedious job of locating certain types of Privilege Escalation issues in complex web applications.

PRIVILEGE ESCALATION TESTING WITH WATCHFIRE APPSCAN 7.0

Until recently, testing for Privilege Escalation issues in web applications was a tedious series of tasks performed manually by application auditors. One of the auditor's tasks was to conduct what is usually referred to as "Differential Analysis"[4].

In a Differential Analysis the auditor browses the web application with the credentials of User A, and records all the different URLs that can be accessed by that user. Next, the auditor browses the same application, but this time with the credentials of User B, who has higher user privileges. When both browsing processes are complete, the auditor manually sifts through the two resulting lists of URLs, locating the differences between them, and then attempts to access URLs belonging only to User B, with the credentials and session identifiers of User A.

Figure 3: Differential Analysis of URLs

With the launch of Watchfire AppScan 7.0 Watchfire introduced an automated process aimed at helping application auditors with the cumbersome process of manual Differential Analysis.

As an automated web application scanner, AppScan includes a web crawler which spiders web applications with great speed and accuracy. In addition, AppScan is also "session aware" and has features that keep track of application session tokens (cookies or parameters) to confirm that it is logged in. With its new Privilege Escalation testing capabilities, users can use AppScan to automatically perform differential analysis of web application URLs, and report flaws in the application's authorization and access control model.

Differential Analysis using AppScan is a simple task performed by loading multiple crawling sessions, each with different application credentials, and then asking AppScan to perform the automated analysis. The results are a list of resources (URLs) that can in practice be accessed by users with lower privileges.

AppScan allows users to load *any* number of different crawl sessions, belonging to users with different permissions, on which it performs the following differential analysis:

Non-Authenticated Users → Current User Crawl Session
Current User Crawl Session → All types of higher privileged Users

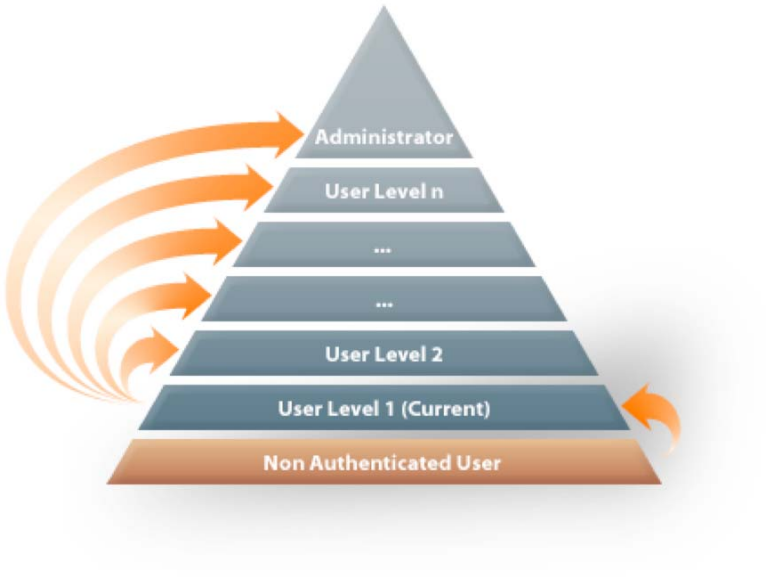

Figure 4: Differential Analysis in AppScan

AppScan first locates the unique (delta) URLs for each user level, and then it automatically attempts to retrieve these URLs with the credentials of a "regular" user. Resources that were successfully accessed without proper authorization are then compared to the same resources that were accessed with the proper authorization.

In addition, AppScan automatically attempts to access the "regular" user's resources *without* authenticating to the web application *at all*.

CONCLUSION

Privilege Escalation vulnerabilities in web applications have existed since the early days of web applications, yet since testing for them is such a complicated and tedious manual task, they are often overlooked in web application assessments.

Furthermore, since Privilege Escalation is considered a "Logical Vulnerability" that can only be assessed by humans, most assume that it is not possible to implement a successful automated process which will locate *all* such issues.

While there is truth in this assumption, things are not as bleak as they appear. Automated processes *can* help with making the testing process faster and more streamlined, and although it cannot cover *all* Privilege Escalation scenarios, it can certainly assist greatly with tasks such as Differential Analysis of URLs, given the proper technology and heuristics.

Watchfire AppScan 7.0 introduces an automated Differential Analysis, which saves auditors a substantial amount of time, allowing them to invest their precious resources in other aspects of web application vulnerability assessments.

REFERENCES

Relevant Documents:

[1] [Full Disclosure] "Data mining MySpace Bulletins" Security Advisory (http://archives.neohapsis.com/archives/fulldisclosure/2006-06/0896.html)

[2] "Challenges faced by automated web application security assessment tools", Robert Auger, CGISecurity.com (http://www.cgisecurity.com/articles/scannerchallenges.shtml)

[3] "Technology Alone Cannot Defeat Web Application Attacks: Understanding Technical vs. Logical Vulnerabilities", Jeremiah Grossman (http://www.whitehatsec.com/home/resources/articles/assets/WHArticleTechnical_vs_Logical_WHF.pdf)

[4] "Hacking Exposed Web Applications (Second Edition) – Chapter 5", Joel Scambray, Mike Shema, Caleb Sima (http://www.webhackingexposed.com/)

ABOUT WATCHFIRE

Watchfire provides software and services to help ensure the security and compliance of websites. More than 800 enterprises and government agencies, including AXA Financial, SunTrust, HSBC, Vodafone, Veterans Affairs and Dell, rely on Watchfire to audit and report on issues impacting their online business. IDC lists Watchfire as the worldwide market share leader in web application vulnerability assessment software, and the company was named a finalist in five categories of the 2007 *SC Magazine* Awards, including Best Security Company. Watchfire's partners include IBM Global Services, PricewaterhouseCoopers, Sapient, Fortify, Microsoft, Interwoven, WebTrends, EMC Documentum and Mercury. Watchfire is headquartered in Waltham, MA. For more information, please visit www.watchfire.com.

