Automated Detection of Session Fixation Vulnerabilities

Yusuke Takamatsu† yusuke@sslab.ics.keio.ac.jp
Yuji Kosuga† yuji@sslab.ics.keio.ac.jp
Kenji Kono†‡ kono@ics.keio.ac.jp

†Department of Information and Computer Science
Keio University
‡Core Research for Evolutional Science and Technology
Japan Science and Technology Agency

ABSTRACT

Session fixation is a technique for obtaining the visitor's session identifier (SID) by forcing the visitor to use the SID supplied by the attacker. The attacker who obtains the victim's SID can masquerade as the visitor. In this paper, we propose a technique to automatically detect session fixation vulnerabilities in web applications. Our technique uses attack simulator that executes a real session fixation attack and check whether it is successful or not. In the experiment, our system successfully detected vulnerabilities in our original test cases and in a real world web application.

Categories and Subject Descriptors

D.2.5 [**Software**]: Software Engineering—*Testing and Debugging*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security

Keywords

session fixation, web application security

1. INTRODUCTION

Many of recent web applications employ session management to keep track of a visitor's activity over the inherently stateless protocol such as HTTP. A session identifier (SID) is usually a hash value uniquely assigned by the web application for the purpose of session management and is usually granted to the visitor on his first visit to the web application. The SID is an attractive target for attackers because attackers can masquerade as the visitor if they can obtain the visitor's SID. Session fixation [1] is a technique for obtaining the visitor's SID by forcing the visitor to use the SID supplied by the attacker. Although session fixation vulnerability can be eliminated in the development phase of web applications, it is laborious to setup a test environment and imposes the intimate knowledge of the attack on the people who conduct the test. According to a report from WhiteHat [3], 12% of websites are vulnerable to session fixation and it takes 106 days on average to fix an vulnerability.

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

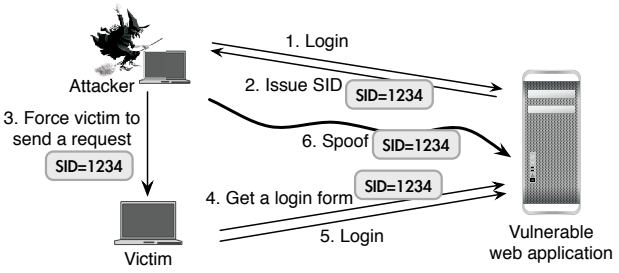

Figure 1: Session Fixation

We propose a technique to automatically detect session fixation vulnerabilities in web applications. Our technique is effective in detecting vulnerabilities because it executes a real session fixation attack and check whether it is successful or not. Technically, our technique first checks whether an SID is changed after a user's login, then it goes on to the phase of attack simulation. In the experiment, our system successfully detected vulnerabilities in our original test cases. We also performed a test on a real world web application, in which our system detected a vulnerability.

2. SESSION FIXATION

Session fixation [1] is an attack technique that forces a visitor to use an SID that the attacker prepared. After the visitor's login, the attacker can masquerade as the visitor by accessing the web application with the SID.

In Figure 1, the attacker logs into the vulnerable web application to obtain an SID, which is usually contained in an HTTP header or a part of the response document (Step 1 & 2). Then, the attacker extracts the SID to embed it into an anchor and lure the victim into clicking on it to send a request to the web application (Step 3 & 4). The web application establishes a session with the visitor. After the victim's login while the session is valid, the attacker can spoof the victim's identity (Step 5 & 6).

Session fixation can be avoided by assigning a new SID each time user logs in to avoid using the SID that the attacker prepared. Restricting the SID usage, for example, by binding an SID to another information such as a special token or the browser's network address, is also effective to prevent session fixation. These countermeasures should be implemented in web applications. Even when the web application is already in service, the security should be tested. However, it is laborious to build a test environment and requires a detailed knowledge of session fixation.

3. PROPOSAL

We propose an effective technique for detecting session fixation vulnerabilities in web applications by actually attempting to perform session fixation attacks. Our technique automatically performs all the steps in session fixation as real attackers do, from the acquisition of an SID to the attacker's malicious login after the valid user's login.

While current session fixation testing requires knowledge and labors, our system can alleviate the burden of test operators by only offering several simple information as follows.

- The parameter name that contains an SID (e.g., PHPSESSID in PHP)
- Attacker and victim's login information (e.g., user name and password)
- Special keywords that only appear in the response message after a valid user's login. (e.g., '`Welcome victim`')

With these information, our technique automates the detection of vulnerabilities with the following three steps.

3.1 Packet Capturing

Our system lies between a user's browser and a web application to intercept innocuous HTTP packets between them (Figure 2). When the user browses the web pages before and after his login with his browser, our system captures all the packets to observe the change of SIDs.

3.2 Initial Inspection

Our system extracts the SIDs from the packets intercepted. If the SIDs has changed at the user's login, this step concludes it is not vulnerable since the change of the SIDs at the login is an effective countermeasure. Otherwise, although it might be vulnerable, it also can be safe due to another countermeasure implemented. To make it clear, it goes to the next step for further inspection.

3.3 Attack Simulation

In this step, our system launches its attack simulator that automatically generates the same environment as a real attacker performs session fixation attacks. The simulator has a virtual attacker and a virtual victim. In the same scenario described in Section 2, the attacker first acquires an SID by logging in with the attacker's information the user initially gave to our system.

After letting the victim access and login to the web application with the SID that the attacker obtained, the attacker check to see if he can successfully log into the web application with the victim's identity. To this end, our system searches the content of the response document for the special keyword that the user gave to our system. If the special keyword appears in the response document, our technique considers this web application is vulnerable.

4. EXPERIMENTS

We implemented a prototype version of our technique against two types of session fixation in terms of where an SID is contained in: a URL or a cookie separately. Other than these, an SID can be delivered via a hidden field in an HTTP response. Additionally, in our current implementation, the information that the user specifies is hard-coded.

To confirm the effectiveness of our technique, we performed experiments against web applications that we created deliberately vulnerable to session fixation, and against a real world web application.

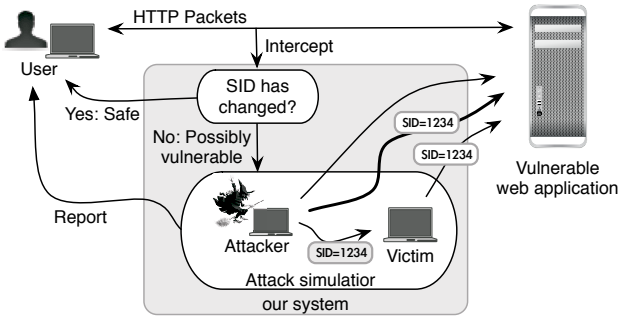

Figure 2: Design of our system

4.1 Original Test Cases

A web application we created works as a real world web application in terms that it issues an SID in response to the visitor's first access, but does not re-issue a new SID after the visitor's login, thus it is vulnerable to session fixation. Our system could find this vulnerability and we also confirmed that it did not raise a false alert to another version of the web application that we had modified to re-issue a new SID.

Another web application we created assigns a special token to each visitor at his login and binds it to an SID for identifying visitors. Thus it is safe even when a new SID does not re-issue at login. We confirmed the attack simulator in our system detected the vulnerability.

4.2 Testing for Real World Application

We also executed our system against a real world web application: Mambo [2]. The login page of Mambo was vulnerable to session fixation and we confirmed it by hand before testing with our system.

In this experiment, we gave attacker and victim information in advance: '`attacker`' for the virtual attacker's user name and password, and '`victim`' for the virtual victim's user name and password. We also set a special keyword as '`Hi, victim`' that indicates the victim's login. Our system could detect the vulnerability.

5. CONCLUSION

We proposed a technique to automatically detect session fixation vulnerabilities in web applications by executing real session fixation attacks. It can reduce laborious work for security checking against session fixation. In our experiment, a prototype version of our system could detect a vulnerability in a real world web application.

6. REFERENCES

[1] M. Kolšek. Session Fixation Vulnerability in Web-based Applications. http://www.acrossecurity.com/papers/session_fixation.pdf, December 2002.

[2] SecurityFocus. Mambo 4.6.2 CMS - Session fixation Issue in backend Administration interface. http://www.securityfocus.com/archive/1/475241, August 2007.

[3] WhiteHat Security, Inc. Website Security Statistic Report (8th Edition). http://www.whitehatsec.com/home/resource/stats.html, November 2009.