

Control Essentials

Volume III, Issue I

January 2008

Business Consulting

- Regulatory Support
 - Policies and Procedures
 - Regulatory Compliance
 - Sarbanes-Oxley
 - GLBA
 - HIPAA
 - BSA
 - PCI (Payment Card Industry) Security Audit
 - Information System Audit Outsourcing
- Enterprise Support
 - Business Continuity Planning
 - Attestation and Assurance
 - SAS-70
 - E - Discovery
 - Forensic Examinations
 - Security Remediation

Technology Consulting

- Application Security
 - Application Security Assessment
 - Application Penetration Test
 - Web Application Security Assessment
 - Web Application Penetration Test
- Network Security
 - Risk Assessment
 - Vulnerability Assessment
 - Penetration Testing
 - Internal
 - External
 - Social Engineering
 - War Dialing
 - War Driving
 - Blue Snarfing
 - Wireless Security Assessment
 - Information System Security Audit
 - Information System Control Audit
 - Network Architecture Design and Assessment
 - Physical Security Assessment
 - Log Analysis

Pandora's Box

How many times have you waited for the launch of the next cool web application that promises to end your worries and make your life easier? With you hanging in the fine balance of enthusiasm and rational thought, let's take a closer look at web applications.

Web applications can be like Pandora's Box, but with more than just troubles. They are often personal favorites of hackers. E-banking applications, online retailers, and government websites, among others, have been prime targets in the past with hackers stealing sensitive information and causing permanent damage to reputations. So before opening your Pandora's Box, be aware that there may be evil inside.

Let's take a minute to introduce Mr. Evil. Mr. Evil is a reclusive, anti-social character who spends most of his time with his computer on one side and a stack of Coca-Cola cans on the other. Through the course of our discussion, we will have a number of guest appearances from Mr. Evil so do keep an eye out for him.

Session Hijacking

Web applications use a technique called *session management* to "remember" the identity and activities of users. Typically, web applications use session tokens such as cookies, URL base and custom headers and hidden form fields to accomplish this purpose. Session token contains a unique, randomly generated session identifier (typically alpha-numeric) that is assigned to a user after he has successfully logged into the application. The session identifier is used as valid authentication "proof" for the subsequent communications between the server and the user. Session identifiers usually last until the user logs out or the session times out.

Although session tokens are very convenient and facilitate the user's experience with an application, if not properly managed, they can be a mouthwatering target for hackers like Mr. Evil. For instance, if Mr. Evil were to steal the sessionID of another user, he would be able to subvert the identification mechanism, impersonate the user and have full access to his data and private information. Mr. Evil can accomplish this in many ways. Let's take a look at a few.

Session Prediction and Brute Force

Many web applications use customized algorithms to generate the value of the sessionIDs. Quite often these algorithms create predictable sessionIDs by incrementing the previous value by 1 or by a time interval. At this point, Mr. Evil would have to gather a sample of session tokens and analyze it in order to discover common patterns. A simple way of gathering sessionIDs is to run automated tools that send a large number of requests to the server. With a sample of, say, 100 requests, Mr. Evil would be able to observe the level of entropy and randomness present in the sessionID value generation scheme. Let's take an example:

SessionID
38757686995785
23157686993240
12357686997098
65757686991022
22757686991799
13757686991122

The above table clearly shows a common pattern in the SessionIDs. Therefore, Mr. Evil could try and predict the remaining values by further observing a larger sample of SessionIDs or perform a brute force attack. A brute force attack involves trying all possible combinations of values until a valid SessionID is found.

So what exactly is Mr. Evil going to do with all of this information? The answer: Session Impersonation.

Once Mr. Evil has obtained a valid SessionID, he can use it to impersonate a website user and access the website with the compromised privileges of the valid user. The only requirement for this attack to succeed is that the user's session still be active.

Session Fixation

In a session fixation attack, Mr. Evil would need to fix the sessionIDs of his victims before they login. Mr. Evil will have to first get hold of a valid sessionID and then try to make his victim login to the web application using the sessionID he prepared for them. Let's take a real case scenario:

Step1: Mr. Evil obtains a valid sessionID by simply visiting the targeted web application or by performing session prediction or a brute force attack.

Step2: Mr. Evil prepares a phishing e-mail or forum post to entice visitors to click on a URL leading to the target website. This URL will have the embedded sessionID collected during the first step. After the victim logs in, Mr. Evil will take control over the session.

SQL Injection

SQL injection is an attack that exploits inadequate input validation controls in a web application. SQL injection is a simple yet powerful attack that could allow an attacker to retrieve sensitive data from a database, execute operating system commands and read/modify the local file system. This attack is based on the manipulation of SQL queries. SQL is the programming language used for interacting with databases. Since the SQL language has many "dialects", that is, the syntax of a query in SQL can vary based on the database type (e.g. Oracle, MySQL, MS SQL, etc.). The first step for Mr. Evil is to identify the type of database he is going to hack into.



Determining the Database Type

A simple method to determine the database type is intentionally creating an error that varies depending on the database implementation. For instance, injecting the SQL statement using the string concatenation character is a common way of accomplishing this. Different database implementations use different characters to concatenate strings:

	MS SQL	Oracle	MySQL
String Concatenation Character	+		concat(str1,str2)

Let's see how Mr. Evil can take advantage of the above information:

Step1: From the web interface, Mr. Evil enters something like "evil@ + evil.com". The input will be concatenated by the web application to a pre-prepared string so that the resulting SQL statement will be something like:

```
SELECT SSN, FirstName, LastName FROM Customers where email = "evil@" + "evil.com";
```

Step2: The above SQL statement will be executed without any errors by a Microsoft SQL database. Microsoft SQL will translate "evil@" + "evil.com" into "evil@evil.com" and run the query without problems. Another database, like Oracle for instance, would complain about the "+" character and respond with an error message.

Step3: If Mr. Evil does not receive an error message, he will know that the database is a Microsoft SQL database. However, even if Mr. Evil receives an error, it is very likely that the details of the error message will reveal the database type name. By performing such tests, Mr. Evil can identify the database type in a relatively short time.

Interacting with the Operating System and the File System

After information gathering, it is now time for Mr. Evil to exploit the vulnerabilities of the web application at a deeper level. All commercial databases implement local file system access to export and import files to/from a database. Additionally, several Microsoft SQL database stored procedures allow direct interactions with the server's operating system. So, if Mr. Evil identifies a way to manipulate these objects and has enough privileges, he can read and write files, create users, and run programs/scripts. Let's take a look at some commands that can be embedded into SQL statements to cause data leakage:

Database	Example of Command
MySQL	load file('/etc/passwd')
MySQL	load data infile 'c:\myfile.txt' into table myt;
Microsoft SQL	BULK INSERT myt FROM 'c:\myfile'

Cross Site Scripting (XSS)

Cross Site Scripting (XSS) is an attack that uses a slightly different approach from the other methods we have seen so far. It targets the web application users rather than the application itself. XSS involves tricking legitimate users into executing malicious code unintentionally. XSS attacks thrive on the concept of exploitation of common web application vulnerabilities such as insufficient input validation combined with social engineering (e.g. phishing) techniques. The commonly used programming languages for XSS attacks include Javascript, ActiveX and Livescript, among others.

Universal XSS Attack

Universal XSS attacks take advantage of the features of Document Object Model (DOM). DOM is a technology that allows programs and scripts to dynamically access and update documents including web pages, pdf files, etc. The following attack scenario is just one of the many that Mr. Evil could use:

Step1: Mr. Evil sends an email to his victims enticing them to read his article (my_article.pdf). However, the link also contains a malicious script:

`http://www.mysite.com/my_article.pdf#a= javascript:execute_this_malicious_code();`

Step2: When the victim clicks on the link, the web browser (e.g. Internet Explorer, Firefox, etc.) will send the request for the pdf article to the server. The browser, however, will not send the portion of the link that follows the # sign. Then, Acrobat Reader will be launched to open the pdf file. At this point the entire URL will be passed to Acrobat Reader, along with the javascript, and will lead to the execution of the javascript.

Cross Site Request Forgery (XSRF)

Although Cross Site Request Forgery (XSRF) sounds very similar to Cross Site Scripting (XSS), it uses a completely different approach. In a XSS attack, the "trusted" party is the server (or what the user believes to be the server) and the user's browser executes the code coming from the trusted source. In a XSRF attack, a legitimate user of a website is involuntarily directed to make a request to the server, which executes the request without verifying whether it was the actual user's intent. A XSRF attack can be carried on as described below:

Step1: Mr. Evil controls a server (www.MrEvil.com) where he has posted a link containing malicious code. The scope of the code is transferring money from the checking account of the victim to Mr. Evil's account.

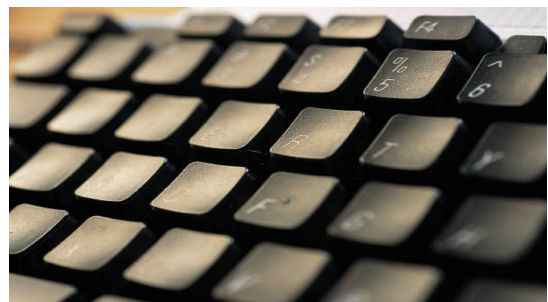
Step2: The victim logs into the e-banking system of his bank and performs his routine operations. The e-banking system assigns a SessionID to the victim.

Step3: The victim visits www.MrEvil.com (e.g. by clicking on a link or through spam) which displays regular content as well as the link containing the malicious code. The user then clicks on that link.

Step4: The malicious code directs the user's browser to the e-banking system of the bank. The e-banking system authenticates the user's browser with the valid SessionID that was previously assigned to the user. The malicious code is executed and Mr. Evil happily goes ahead and transfers the money from the victim's checking account.

Although the example above sounds pretty neat, Cross Site Request Forgery is actually quite difficult to perform. The following requirements must be met for Mr. Evil to succeed:

1. The user has already authenticated into the e-banking system and the session token is still active.
2. The e-banking system allows actions to be performed without the need for re-authenticating the user using, for example, session tokens.
3. Mr. Evil must know how to interact with the e-banking system and must know some intimate details such as knowing what function to call and which parameters to use.
4. Mr. Evil also needs to fool the valid user and entice him into clicking on the link containing the malicious code.



Conclusion

As with the old tale, Pandora could save only Hope in the box. There is Hope with web application security too despite the fact that web applications are typically built with functionality in mind rather than security. Software programmers and engineers work in a very complex environment which requires them to balance the functionality of the applications with the security requirements. The complexity compounded with inexperience and accidental errors opens the doors to security breaches. Therefore, it is essential that security be considered from the earliest stage of the web application development life cycle. If security is introduced only at the end of the development life cycle or somewhere in between, the time and cost necessary to implement it will increase considerably. Furthermore, periodic code reviews should be performed in order to assess the overall security of the web application, identify the vulnerabilities and proactively mitigate the risks involved.

Enterprise Risk Management At A Glance

ERM professionals bring a unique mix of strong academic backgrounds complemented by professional experience ranging from "Big Four" consulting firms to major corporations and prestigious professional certifications. A snapshot of ERM's profile can be seen below:

Education

Qualifications

M. S. in Computer Information Systems
M. S. in Information Networking
M. S. in Management Information Systems
Master of Accounting Information Systems
Master of Business Administration

Universities

Carnegie Mellon University, Pittsburgh, Pennsylvania
Syracuse University, Syracuse, New York
Xavier University, Cincinnati, Ohio
University of Miami, Miami, Florida
Florida International University, Miami, Florida

Certifications

Certified Public Accountant (CPA)
Certified Information Systems Security Professional (CISSP)
Certified Information Systems Auditor (CISA)
Certified Information Systems Manager (CISM)
Certified Information Technology Professional (CITP)
GIAC Security Essentials Certification
GIAC Systems and Network Auditor
Microsoft Certified Professional

Prior Work Experience

PriceWaterhouse Coopers
CERT® Coordination Center
SONY Electronics Latin America, Inc.
RJR Nabisco
Diageo plc
Arthur Young
Carnegie Mellon CyLab
Evertec Inc.
American Bankers Insurance Group
Chesebrough Pond's
Starboard Cruise Services, Inc.
Demotte Consulting, Ltd.

Some of our Clients...

ABN AMRO Private Banking
Bacardi-Martini, Inc.
CitiBank
Carnival Cruise Lines
Commerce Bank
Florida Power & Light Company
Knight Ridder
North Broward Hospital District
Ocean Bank
Rinker Materials
Sylvania Lighting International
The International Bank of Miami

KEEPING WATCH OVER YOUR BEST BUSINESS INTERESTS

enterprise risk management

The Control Professionals

800 S. Douglas Road, North Tower (# 835),
Coral Gables, FL 33134.
P: (305) 447 6750 F: (305) 447 6752
Email: info@emrisk.com Web: www.emrisk.com