IJCSN International Journal of Computer Science and Network, Volume 3, Issue 2, April 2014
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 0.274

Security Surface Analysis and Patching

[1] Pritish Niranjan, [2] Apurva Mangrulkar, [3] Tejaswini Bhandarkar, [4] Krushna Gupta

[1, 2, 3, 4] Department of Comp. Science & Engineering, Rajiv Gandhi College of Engineering & Research
Nagpur, Maharashtra, India

Abstract - Web Applications and Web Services drive the current iteration of the web. They are emerging to serve new platforms and new devices with an ever-increasing scope of information and services. With the fast development, there is also an ever increasing need of safety for these services. Nothing on the web can be stated as fully secure. Hence, there is a need of Security Analysis and conquering the security threats. Security analysis and patching are hypercritical to the improvement of software security. We use the web application's attack surface measurement as a pointer of the system's security; the larger the attack surface, the insecure the system. Our prime aim is to minimize this surface. Also we perform attacks on the system to expose its vulnerabilities, and remove them by developing logics and implementing these to the applications code which is also a method called patching. This paper will provide us in general, the knowledge to handle security- analysis and improvement of real web applications and protect it from majority of attacks. The attack surface of a system portrays the exposure of application objects to attackers and is affected primarily by architecture and design decisions. To reduce its overall vulnerability reducing the attack surface of an application is expected. The online applications have an increasing number of users, increasing their appeal to attackers, in spite of the legion of programmers and developers engaged to fend them.

Keywords - *Security Surface, security threats, Security Analysis, Attack Surface, Web Application Security and Vulnerability.*

1. Introduction

What Do We Mean By Security

Security basically means protecting assets. Assets may be palpable items, such as a Web page or your customer database — or they may be less tangible, such as company's reputation. Security is not a destination, it's a path. As you analyze your infrastructure and applications, you identify potential threats and understand to what extent each threat presents a degree of risk. Security is about risk handling and implementing effective countermeasures. Web application exploits run the technical spectrum of complex buffer overflows to single-character manipulations of the URI. The second most important tool in the Web security repository is a tool for sending raw HTTP requests.

Threats may include hackers, malcontented coders, Anonymous criminal establishments, tsunamis, disk failures, stumbling over power cords, or anything else with the potential to adversely affect your site. They represent participators—who or what which acts upon a site. The key to designing a good defense is to thoroughly understand how an attack works. An understanding of the limitations of a countermeasure and the probability of it being bypassed by other vulnerabilities is very critical to the applications security defense. Some countermeasures might show up innumerable times, while others would make only a brief appearance. The contributions of this paper are:

(1) We present important aspects in web application security management, which poses constitutional, inherent challenges for building secure web applications. The root cause of corresponding vulnerabilities is the failure of web applications to fulfill the above security properties, which permit for successful exploits.
(2) In this paper we focus on building vulnerability-free web applications, identifying and fixing vulnerabilities, probable causes or motives behind the introduction of attacks, resources at stake and discuss web hacking.

We structure the rest of this paper as follows. We first describe foundations of security and its characteristics in Section 2.1. Then, we illustrate the essential security paradigms critical to the security of any web application in Section 2.2. In section 2.3 we define attack surface. Motives behind attacking web applications are discussed in section 2.4. Web application hacking and its different aspects are described in section 2.5. In section 2.6 we discuss the resources which can be affected. Patching technique and use of patches are given in section 2.7.
We conclude our paper in Section 3.

2. The Foundations of Security

2.1 Security depends on the following elements:

2.1.1 Authentication

Authentication casts the question: "who are you?" It is the process of distinctly identifying the clients of an application or service.

2.1.2 Authorization

Authorization addresses the question: "what can you do?" It is the process which rules the resources and operations that the authenticated client is permitted to access. Databases, files, tables, rows, along with system-level registry keys and configuration data are the main resources of any application. Operations comprise of performing transactions such as selling or purchasing a product, transferring money, online consultation, registration or increasing a customer's credit rating.

2.1.3 Confidentiality

Confidentiality, also referred to as secrecy, is nothing but making sure that data remains private as well as confidential, and that it cannot be used, accessed or viewed by unauthorized users or attackers who track the flow of traffic across a network. To enforce confidentiality encryption is frequently used.

2.2 An Attack Surface

The attack surface of a system represents the susceptivity of application objects and their exposure to attackers. Design component decisions and system architecture are the key determinants of the attack surface. Reducing the attack surface of an application is expected to reduce its overall vulnerability.

2.3 Why Attack Web Applications

The encouragement for hacking is numerous and has been discussed at length for many years in a variety of forums. It is very important to point out that features of web applications which make them so alluring to attackers. Understanding these elements leads to a much clearer perspective on what safeguards need to be put in place to reduce a risk.

2.3.1 Ubiquity: Web applications are increasingly used today for critical services and continue to spread rapidly across public and private networks. Web attackers are unlikely to encounter a shortage of juicy targets anytime soon.

2.3.2 Simple techniques: Web app attack techniques are fairly easily understood, even by the layperson, since application input fairly trivial. Compared to the knowledge required to attack more complex applications or operating systems (for example, crafting buffer overflows), attacking web apps is a piece of cake.

2.3.3 Anonymity: The Internet still has many unaccountable regions today, and it is fairly easy to launch attacks with little fear of being traced. Web hacking in particular is easily laundered through open HTTP/S proxies that remain plentiful on the Net. Sophisticated hackers will route each request through a different proxy to make things even harder to trace. Arguably, this remains the primary reason for the proliferation of malignant attacking, because this anonymity strips away one of the primary deterrents for such behavior in the physical world (i.e., being caught and punished).

2.3.4 Bypasses firewalls: Inbound HTTP/S is permitted by most typical firewall policies but this is not a vulnerability of the firewall—it is an administrator-configured policy. Even better (for attackers, that is), this configuration is probably going to increase in frequency as more and more applications migrate to HTTP.

2.3.5 Custom code: with the proliferation of easily accessible web development platforms like ASP.NET and LAMP (Linux / Apache/ MySQL/ PHP), most web applications are assembled by developers who have little prior experience.

2.3.6 Immature security: HTTP doesn't even implement sessions to separate individual users. The basic authentication and authorization plumbing for HTTP was bolted on years after the technology became popular and is still evolving to this day. Many developers code their own and get it wrong.

2.3.7 Constant change: Usually many people constantly "touch" a web application: developers, system administrators, and content managers of all stripes.

2.3.8 Money: Whether through direct break-ins to web servers, fraud directed against web end users (aka phishing), or extortion using denial of service, the disastrous situation today is that web crime pays.

2.4 Web Application Hacking

We define a web application as one that is accessed via the Hyper Text Transfer Protocol, or HTTP. Thus, the essence of web hacking is tampering with applications *via HTTP*. There are three simple ways to do this:

- Directly employing the web applications via its graphical web interface.
- Intruding with the Uniform Resource Identifier, or URI.
- Tampering with HTTP elements not contained in the URI. HTTP headers are generally used to store additional information about the protocol level transaction.

2.4.1 Weak Spots

Here is a quick analysis of the types of attacks that are typically made against each component of web apps:

2.4.2 Web Platform: Web platform software vulnerabilities, including underlying infrastructure like the HTTP server software and the development framework used for the application (example for it is ASP.NET or PHP).

2.4.3 Web Application: Authentication attacks, authorization attacks, attacks for site structure, validation, application logic and management interfaces.

2.4.4 Database: Privileged commands are run via database queries and attackers use query manipulation to return and damage excessive datasets. The most devastating attack here is SQL injection.

2.4.5 Web Client: Execution of Active content, exploitation of client software vulnerability, cross-site scripting errors, and phishing.

2.4.6 Availability: One of the greatest threats any publicly accessible web application can face is definitely the denial of service (DoS) attack which is frequently neglected in the hurry to address more sensational "hacking" attacks.

Some security- applicative examples of HTTP headers include:

• *Authorization:* It defines what rights or actions the legitimate user is provided by the application. It's a function of specifying access rights to resources.

• *Cache-control:* Defines whether intermediate proxy servers should cache a copy of the request.

• *Cookies:* Most commonly used to reserve custom authentication/session tokens for applications.

2.5 Input Validity

Since all successive security decisions are made based on the identity established by the supplied credentials it is evident that authentication plays a critical role in the web application security. This paper exhibits threats to common web authentication mechanisms, as well as threats that penetrate authentication controls effectively. We've organized our discussion in the following section around the most generic types of authentication prevalent on the Web at the time of this writing:

Username/password: This is the most prevalent procedure of authentication on the Web because of its simplicity.

Strong(er) authentication: Stronger forms of authentication i.e. token and certificated-based authentication is being provided by many websites since it's widely distinguished that username/ password authentication has underlying weaknesses.

• *Authentication services:* Many web sites outsource their authentication to Internet services which enforces a proprietary management of identity and authentication protocol.

2.6 Web Application Attacks

Cross-site scripting, SQL injection etc. is examples of input validation attacks. They are distinguished on the basis of the sites of execution of the malicious contents. We illustrate the most popular validation attacks in the following.

i) SQL Injection: The methods for hacking SQL injection have advanced inordinately over the last 10 years while the elementary programming errors that lead to these vulnerabilities have not changed. This is an utterly asynchronous evolution wherein the hacks become simpler and more effectual while simple counteractants remain absent. Manipulation of the commands passing between the web application and its database is made possible to the attacker by the SQL injection vulnerabilities. Databases are the parts of an application that drive the dynamic content, reserve product catalogs, track orders, maintains profiles, and perform many other tasks behind the scenes. New information might be inserted into the database; stored information might be updated; information can also be removed from the database. In all these cases the web site executes a

database (SQL) command with a specific intention. SQL injection goals include thievery of credit cards, circumventing security checks, or executing code on the database, authentication check bypassing —or create a divide by zero error; perhaps leading to crashing of the application.

ii) Cross-Site Scripting: when malicious contents within user input flow into web responses without sanitization, a cross-site scripting (XSS) attack is commenced.
This attack usually thrives among web sites, needing no more aliment than HTML tags and a scattering of JavaScript to thoroughly vanquish a site's security. The assumed-to-be safe sites we use for news, transactions, social networking, email, banking, and more also fall prey to this attack. Generally, depicted as HTML injection, XSS is the pervasive and persistent spider of the web.

iii) Cookie abuse: Despite a somewhat checkered security history, cookies remain the most popular form of session /authorization management within web applications. Owing to their central role, malicious hackers have devised numerous ways to manipulate, hijack, steal, or in other words, abuse cookies. The long history of security attacks targeting cookies is not indicative of a design problem with them but rather an evidence of just how these tiny bits of data are to vital to authentication, authorization and state management in servers of application.

2.6 Resources

Typically, the eventual goal of the attacker is to gain unauthorized access to web application resources. Let's throw some light on the kinds of resources a web application holds. Although web applications can have many layers most of them have three: presentation layer, logic layer, and data layer. The *presentation* layer usually comprises of a Hyper Text Markup language (HTML) page, generated by scripts either statically or dynamically. These pages are generally not that useful to attackers. The exact could be said about the *logic* layer, although often the developers make mistakes at this tier that leads to jeopardy of other aspects of the application. Customer data, credit card numbers, and so on are present on the data layer, hence it is said that "The juiciest information lies on the data tier".

2.7 What is a Patch

A patch is an addition of code to overcome the flaws or loops in the application code. Thus, we define patching technique as the process to insert the patches into the code as per required. Patches are used to mend the code when flaws in the code are encountered.

3. Conclusion

This paper provides an all-around survey of neoteric research results in the domain of web application security. We described foundation characteristics of web application security, identified important security attributes that secure web applications should preserve and also discussed motive behind hacking, resources and important vulnerabilities. An ever-escalating number of attacks target your application. They directly penetrate through your environment's front door using HTTP. The conventional model and the reliance on firewall and host shields are not sufficient when used in solitude. Securing the application involves securing three layers: the network layer, host layer, and the application layer. A safe, secure network and host platform infrastructure is a must. Additionally, your applications must be designed and constructed using secure design and development guidelines following antediluvian security principles.

References

[1] Pratyusa Manadhata and Jeannette M. Wing. "An Attack Surface Metric." Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, July 2005. CMU-CS-05-155.

[2] Thomas Heumann, Jorg Keller,"Quantifying the Attack Surface of a Web Application", University of Hagen, Dept. of Mathematics and Computer Science**.**

[3] A. Tajpour, "SQL injection detection and prevention tools assessment" -Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference, 9-11 July 2010.

[3] A. Tajpour, "SQL injection detection and prevention tools assessment" -Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference, 9-11 July 2010.