A Black-Box Methodology for Probing Business Logic Vulnerabilities in Modern Day Web Applications

Adebanjo A Falade[1], Neeraj Kumari[2]

College of Computing Science & Information Technology TMU Moradabad 244001, Uttar Pradesh, INDIA

[1] faladeadebanjo@gmail.com
[2] neeraj.computers@tmu.ac.in

***Abstract*— in a web application, there are three vital security properties that should be reviewed; input validation, state integrity and logic correctness. The failure of web applications in passing these reviews is the main cause of vulnerabilities and successful exploitation.**

***Keywords*— Web Application, Modern Day, Black Box, business**

I. INTRODUCTION

While a significant amount of attention have been given to code-based and syntax-based attacks on web applications such as SQL Injection and Cross Site Scripting (XSS) [2], there exists, another class of attacks not easy to categorize or classify and can be seen as more of an art [2]. This attack has been referred to as business logic attack.

As a large number of business processes move into web-based technologies, web applications have become the core system for effecting business processes over the internet [2]. Business logic attacks take advantage of the ability of these web applications to manage important functions like user interaction, application security, application state and performance [3]. As business processes are specific to a particular organization, thus the business logic is specific to each web application and business logic attacks are also peculiar to their specific targets.

The purpose of this paper is to give an overview of business logic attack vectors and a methodology that can be used to access the business logic of web applications from an attacker point of view. Business logic vulnerabilities have mostly been realized only after it has been attacked, and with the wide variety of this form of attack, the full extent of its reach may never be known. This proposed methodology is based on a comprehensive study of the patterns of known attack vectors whilst providing insight on addressing the main cause of the vulnerability – faulty business logic.

II. THE ISSUE

Business logic attacks are becoming a more grievous issue in web application attacks as they are deeply tied into the company's business process. To find a lasting solution to these attacks, an understanding into the complexity of logical flaws and methods of attacking these vulnerabilities is needed. This will be discussed according to the selected studies [1], [3], [4] and [5].

III. THE COMPLEXITY OF LOGIC FLAWS

For every distinct feature added during web application development, the security design for the web application gets more complicated. The complication lies greater with ensuring logical correctness of the web application which makes

sure that everything is executed correctly as intended by the developers.

According to the logic correctness property proffered by [1],

"Users can only access authorized information and operations and are enforced to follow the intended workflow provided by the web application"

A web application, implementing business processes, is a complex application of many lines of code (which are error-prone), frameworks and components (including third party programs or extensions, for example, Facebook games), all working together to achieve the business requirements behind the development of the web application [1]. With the complexity of these business processes being implemented by web applications, the complexity of logic flaws that may exist, grows.

IV. UNDERSTANDING BUSINESS LOGIC ATTACK VECTORS

Logic vulnerabilities move beyond banking or commerce web applications to any other web application effecting a business process and will require knowledge of the application, the business process and the technology behind these web applications to successfully exploit. Business logic attack vectors have been collated from [3], [4], [5] to reveal attack pattern.

Abusing Workflows:

Abusing the web applications' workflow is common amongst logical based attacks as they are typically controlled by redirects and page transfers. For example, in the normal workflow of an application from A to B to C, an attacker to skip the straight line from A to C or go back to A from C [4]. A few techniques proffered by [4] for abusing workflows include;

a) Changing requests in a code path from HTTP POST to GET or vice versa.

b) Going through steps out of order or skipping steps that will normally verify or validate an action or information

c) Repeating a step or series of steps

d) Performing an unexpected action

A good example set by [3:10] explained how a user could go back during a wire transfer and change discount values after the last step with a valid token has been completed, giving that change in value, the same valid token and a valid, 'unauthorized' discount.

Exploit policies and practices:

Inadequacies with the web applications' policies and practices expose the web application to business logic attacks. A website may comply with all policies but may remain insecure because policies are not made with all aspects of security in mind [4]. For example, in banks, the U.S government established that records should be kept of any financial transaction that exceeds a daily aggregate limit of $10,000 [6] to identify money laundering and other suspicious activity. A money launderer can instead handle transactions as large as $9,876, lower than the limit as specified in the policy and thus, such transactions may go on undetected for a while as the web application is not obliged to flag such events.

Similarly, in 2008, a man was convicted of defrauding Apple of 9,000 iPod shuffles [7] by exploiting their policy which states that,

"You will be asked to provide a major credit card to secure the return of the defective iPod shuffle. If you do not return th e defective iPod shuffle to

Apple within 10 business days of receiving your replacement, Apple will charge you for the replacement." [16]

He successfully exploited this by requesting replacement using credit cards which were just past their limit. These cards were valid, even if they could not be charged later on, and thus obeyed the policy.

However, not all attacks concerning policy bypass is for financial gain or money laundering as evident in the Time Magazine online poll manipulation case in April 2009 [8]. First, the attackers exploited a business policy of "one vote equals one person" because the website had no rate limit, neither authentication nor validation mechanism. Later, Time added counter-measures which included authentication and validation using a salted hash. However, another flaw appeared as the salt was embedded in the client-side Adobe Flash application [9] and again it was possible to manipulate the votes.

There are other examples just like the above examples in [4], but it is peculiar to the organization's policies/practices which are always prone to loopholes, and these loopholes are the vulnerabilities [5].

Induction:

Induction has to do with inference from information provided within the code and behaviour of the web application. It is possible for an attacker to carry out some form of induction from suspicious easily guessable/predictable parameter names and **predict, forge or manipulate** legitimate requests. Parameter names in most HTTP GET and POST requests in the form of name/value pairs, XML, JSON or Cookies are guessable, predictable and can be tampered with, as a result. Sometimes, this may require a combination of logical guessing, brute-forcing and creative tampering to decipher the logic. Below are useful attack vectors of induction:

V. Authentication parameters and privilege escalation:

Because applications can manage access control lists and privileges, any authenticated user has access to some internal parts of the application but if authorization implementation is weak, it could likely include problems such as accessing another user's account or acquiring greater permissions than what was originally assigned at login. For example, if an application passes ACLs as cookies at time of authentication, this information can be tampered and exploited. A certain parameter becomes a target if the parameter name suggests ACL or permissions. The target value is now evaluated, predicted and tampered. The value to be tampered may be hex, binary, string, etc and tampering can involve changing bit patterns (1 to 0) or permission flags (Y to N, R to W) [3].

Critical Parameter Manipulation and Access to Unauthorized Information/Content:

When the business logic of an application is processing parameters such as name-value pairs (which are guessable and can be tampered with), without proper validation, this allows a malicious user to perform unauthorized functions. For example, a banking application, after authentication, allows the user to request authorized functions and while making these requests, some parameters are being supplied to the application such as the "accountid" parameter. If this parameter is easily guessable, then an attacker can successfully inject another user's accountid. Also if the application does not do a validity check to map the existing session to the original logged in account, then another user's information gets disclosed [3]. A similar example is the Binary.com privilege escalation case [10] where the PIN parameter was visible in an <iframe> tag. Guessing another user's PIN was enough to get into that user's account without proper validation [5].

c. Developer's cookie tampering and business process/logic bypass:

This is an easy way to create a logical bypass to perform functions ordinarily not open to the malicious user. When authentication occurs, to maintain the state over HTTP the developer may decide to set cookies in the browser. This way, the cookies can be tampered while it is being passed to, for example, upgrade membership from silver to platinum [3].

VI. METHODOLOGY FOR ATTACKING BUSINESS LOGIC VULNERABILITIES IN WEB APPLICATIONS

This proposed methodology is a black-box methodology at its preliminary stage that focuses on testing for most common business logic vulnerabilities and ways of exploiting them. This is done with no prior knowledge of the system or no information provided by the organization. The methodology is divided into 4 phases:

1. Profiling phase
2. Analysis phase
3. Test/Attack phase
4. Evaluation phase

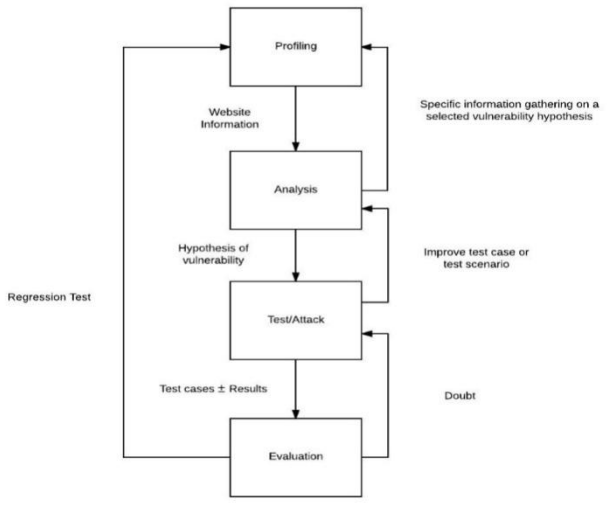

Fig.1: Proposed Methodology for Attacking Business Logic Vulnerabilities

VII. CONCLUSION AND RECOMMENDATION

This proposed methodology shows how business logic vulnerabilities can be discovered and attacked. In essence, it shows how it can be tested for. Protecting an application against business logic attacks can be daunting as every aspect of the application needs to be considered as a potential attack surface. Business logic attacks is an art that demands creativity to detect and exploit, however, fixing the problem may not be as easy as patching a component (evident by the complex nature of business logic flaws) and so, reiterations of the methodology is necessary.

An extensive testing of the methodology is recommended as this aspect was not covered by this paper.

REFERENCES

[1]Xiaowei Li and Yuan Xue (2011). A Survey on Web Application Security. Department of Electrical Engineering and Computer Science. Vanderbilt University. [Retrieved 21 March, 2017].

[2]Erik Couture, (2013).Web Application Injection Vulnerabilities. A Web App's Security Nemesis? GIAC (GWAPT) Gold Certification. [Retrieved 21 March, 2017].

[3]Rapid7 Whitepaper, (2015). Top 10 Business Logic Attack Vectors: Attacking and Exploiting Business Application Assets and Flaws – Vulnerability Detection to Fix. [Retrieved 21 March, 2017].

[4]Mike Shema, 2010. Seven Deadliest Web Application Attacks. The Seven Deadliest Attacks Series. Syngress Burlington, [Retrieved 21 March, 2017].

[5]Peter Yaworski, (2016). Web Hacking 101: How to Make Money Hacking Ethically. [Retrieved 21 March, 2017.

[6]"FinCEN's Mandate From Congress | FinCEN.gov", Fincen.gov, 2017. [Online]. Available: https://www.fincen.gov/resources/fincens-mandate-congress. [Accessed: 25 April, 2017].

[7]"InformationWeek - iPod Repairman Charged With Defrauding Apple", Informationweek.com, 2017. [Online].Available:http://www.informationweek.com/desktop/ipod-repairman-charged-with-defrauding-apple/d/d-id/1077846?print=yes. [Accessed: 25 April, 2017].

[8]E. Schonfeld, "Time Magazine Throws Up Its Hands As It Gets Pwned By 4Chan", TechCrunch, 2017. [Online].Available:https://techcrunch.com/2009/04/27/time-magazine-throws-up-its-hands-as-it-gets-pwned-by-4chan/.[Accessed: 25 April, 2017].

[9]"Hackers stuff ballot box for Time Magazine's top 100 poll", Theregister.co.uk, 2017. [Online]. Available: https://www.theregister.co.uk/2009/04/17/time_top_100_hack/. [Accessed: 25 April, 2017].

[10] "Binary.com disclosed on HackerOne: login to any user's cashier...", HackerOne, 2017. [Online]. Available:https://hackerone.com/reports/98247. [Accessed: 25 April, 2017].

[11] Noa Bar-Yosef, 2009. Business Logic Attacks – BATs and BLBs. The OWASP Foundation. https://www.owasp.org/images/2/27/BNL09_OWASP_Benelux_2009,_Business_Logic_Attacks_-_v2.pdf. [Retrieved 21 March, 2017].

[12] "Business Logic Attacks", Ciphertech.com.tw, 2017. [Online]Available http://www.ciphertech.com.tw/index.php/impervawa-kbinfo/451-business-logic-attacks. [Accessed: 26 April, 2017].

[13] "Shopify disclosed on HackerOne: An administrator without any...", HackerOne, 2017. [Online]. Available: https://hackerone.com/reports/100938. [Accessed: 26 April, 2017].