Net Helplessness Detection: The Case of Cross-SiteRequest Forgery

D.Shine Rajesh
Professor
Department of Information Technology
Malla Reddy Engineering College for Women
(UGC-Autonomous)
Maisammaguda, Hyd-500100, Telangana, India.

Vanga Keerthi Sree
Student
Department of Information Technology
Malla Reddy Engineering College for Women
(UGC-Autonomous)
Maisammaguda, Hyd-500100, Telangana, India.

Shravani Puppala .
.Student
Department of Information Technology
Malla Reddy Engineering College for Women
(UGC-Autonomous)
Maisammaguda, Hyd-500100, Telangana, India.

Vadathya Sravani
Student
Department of Information Technology
Malla Reddy Engineering College for Women
(UGC-Autonomous)
Maisammaguda, Hyd-500100, Telangana, India.

Abstract— In this project, we propose a methodology to leverage Machine Learning (ML) for the detection of web application vulnerabilities. Web applications are particularly challenging to analyses, due to their diversity and the widespread adoption of custom programming practices. ML is thus very helpful for web application security: it can take advantage of manually labeled data to bring the human understanding of the web application semantics into automated analysis tools. We use our methodology in the design of Mitch, the first ML solution for the black-box detection of Cross-Site Request Forgery (CSRF) vulnerabilities. Mitch allowed us to identify 35 new CSRFs on 20 major websites and 3 new CSRFs on production software.

INTRODUCTION

Web applications are the most common interface to security sensitive data and functionality available nowadays. They are routinely used to file tax incomes, access the results of medical screenings, perform financial transactions, and share opinions with our circle of friends, just to mention a few popular use cases. On the downside, this means that web applications are appealing targets to malicious users (attackers) who are determined to force economic losses, unduly access confidential data or create embarrassment to their victims. Securing web applications is well known to be hard.

There are several reasons for this, ranging from the heterogeneity and complexity of the web platform to the adoption of undisciplined scripting languages offering dubious security guarantees and not amenable for static analysis. In such a setting, black-box vulnerability detection methods are particularly popular. As opposed to white-box techniques which require access to the web application source code, black-box methods operate at the level of HTTP traffic, i.e., HTTP requests and responses. Though this limited perspective might miss important insights, it has the key advantage of offering a language-agnostic vulnerability detection approach, which abstracts from the complexity of scripting languages and offers a uniform interface to the widest possible range of web applications. This sounds appealing, yet previous work showed that such an analysis is far from trivial. One of the main challenges there is how to expose to automated tools a critical ingredient of effective vulnerability detection, i.e., an understanding of the web application semantics. Example: Cross-Site Request Forgery (CSRF) Cross-Site Request Forgery (CSRF) is a well-known web attack that forces a user into submitting unwanted, attacker controlled HTTP requests towards a vulnerable web application in which she is currently authenticated. The key concept of CSRF is that the malicious requests are routed to the web application through the user's browser, hence they might be indistinguishable from intended benign requests which were actually authorized by the user.

A typical CSRF attack works as follows:

1) Alice logs into an honest yet vulnerable web application, e.g., her preferred social network. Session authentication is implemented through a session cookie that is automatically attached by the browser to any subsequent request towards the web application;

2) Alice opens another tab and visits an unrelated website, e.g., a newspaper website, which returns a web page including malicious advertisement;

3) The malicious advertisement sends a cross-site request to the social network using HTML or JavaScript, e.g., asking to "like" a given political party.
Since the request includes Alice's cookies, it is processed in her authentication context at the social network. This way, the malicious advertisement can force Alice into putting a "like" to the desired political party, which might skew the result of online surveys.
Notice that CSRF does not require the attacker to intercept or modify user's requests and responses: it suffices that the Preventing CSRF

To prevent CSRF, web developers have to implement explicit protection mechanisms. If adding extra user interaction does not affect usability too much, it is possible to force re-authentication or use one-time passwords / CAPTCHAs to prevent cross-site requests going through unnoticed. In many cases, however, automated prevention is preferred: the recently introduced SameSite cookie attribute can be used to prevent cookie attachment on cross-site requests, which solves the root cause of CSRF and is highly recommended for new web applications. Unfortunately, this defense is not yet widespread and existing web applications typically filter out cross-site request by using any of the following techniques:

1) checking the value of standard HTTP request headers such as Referrer and Origin, indicating the page originating the request;
2) checking the presence of custom HTTP request headers like X-Requested-With, which cannot be set from a cross-site position;
3) checking the presence of unpredictable anti-CSRF tokens,set by the server into sensitive forms.

A recent paper discusses the pros and cons of these different solutions. However, all three options suffer from the same limitation: they require a careful and fine-grained placement of security checks. For example, tokens should be attached to all and only the security-sensitive HTTP requests, so as to ensure complete protection without harming the user experience.
Using a token to protect a "like" button is useful to prevent the attack discussed above, yet having a token on the social network homepage is undesirable, because it might lead to rejecting legitimate cross-site requests, e.g., from clicks on the results of a search engine indexing the social network. In the end, finding the "optimal" placement of anti-CSRF defenses is typically a daunting task for web developers. Modern web application development frameworks provide
Automated support for this, yet CSRF vulnerabilities are still routinely found even in top-ranked websites. This motivates the need for effective CSRF detection tools. But how can we provide automated tool support for CSRF detection if we have no mechanized way to detect which HTTP requests are actually security-sensitive.are passed - No splits.
This work presents the most current and comprehensive understanding of a not very well understood web vulnerability known as the CSRF (Cross-Site Request Forgery) and provides specific solutions to identify and defend CSRF vulnerabilities. The immediate benefits of this work include tangible and pragmatic application framework for use by individuals, organizations and developers, either as consumers or providers of web services. This work responds directly to the challenges of keeping pace with the evolving cyber technologies and vulnerabilities that increasingly expose businesses towards privacy and identity theft specific attacks, where the traditional anti-virus and anti-spyware approaches fail. The urgency to come up with appropriate detection and defensemechanism against the lethal CSRF attacks is indicated due to expanding cloud based technologies, HTML5, Semantic Web, and various emerging security frameworks comprised of inchoate vestigial of "Big Data" that demand exceedingly evolved defense mechanisms. A methodical approach is used to investigate CSRF attacks and remedies are proposed by introducing a novel distinctive set of algorithms that use intelligent assumptions to detect and defend CSRF. In this work, design details of a CSRF Detection Model (CDM), implantation and experimentation results of CDM are elaborated to detect, predict and provide solutions for CSRF attacks on contemporary Web Applications and Web Services environment. Additionally, CDM based recommendations for users and providers of cyber security products and services are presented. Cross-Site Request Forgery (CSRF) attack causes actions on a web application without the knowledge of the user in an authenticated browser session. CSRF attacks specifically target state-changing requests like transferring funds, changing email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application. CSRF, also known as the Sleeping Giant, was considered to be one of the top 5 web vulnerabilities only 4 years ago. Even so, at least 270 incidents of CSRF attacks have been reported as of 2016. Not much has improved in terms of new CSRF solutions since the CSRF problem appeared in the horizon in 2010. Cross-Site Reference Forgery (CSRF) and Cross-Site Scripting (XSS) vulnerabilities have received much attention recently. An XSS attack, one of the top 3 current cyber security challenges, occurs when an attacker injects malicious code (typically JavaScript), including a CSRF attack code, into a site for the purpose of targeting users of the site, e.g., sites that allow posting comments. According to the Open Web Application Security Project (OWASP), an open web community dedicated to address cyber security challenges, CSRF is one of the top eight cyber security vulnerabilities in the world, today. While CSRF attacks are simple to create and exploit, amazingly, they are difficult to identify and mitigate.
A search for "Cross Site Scripting" (which differs from CSRF) on the ACM Digital Library returned 117 papers, while a search for "CSRF" returned only four papers. A search for "XSS" on Safari Books Online (a collection of over 5000 books on technology) showed the term appeared in 96 books, while "CSRF OR XSRF" appeared in only 13 books. Very few CSRF solutions are developed and implemented. Even so, while current solutions still lack common applicability all the pieces for large scale massive CSRF attacks are already in place [53]. This state of the current relentless CSRF attacks and meager defenses dynamics is the primary motivation for undertaking this study.

EXISTING SYSTEM

In the existing system Securing web applications is well known to be hard. There are several reasons for this, ranging from the heterogeneity and complexity of the web platform to the adoption of undisciplined scripting languages offering dubious security guarantees and not amenable for static analysis. Though this limited perspective might miss important insights, it has the key advantage of offering a language-agnostic vulnerability detection approach, which abstracts from the complexity of scripting languages and offers a uniform interface to the widest possible range of web applications.

- **Disadvantages -** In white-box techniques which require access to the web application source code.
- Black-box methods operate at the level of HTTP traffic, i.e., HTTP requests and responses.
- **Algorithm**: Burp and ZAP tools

PROPOSED SYSTEM

Cross-Site Request Forgery (CSRF) is a well-known web attack that forces a user into submitting unwanted, attacker controlled HTTP requests towards a vulnerable web application in which she is currently authenticated. The key concept of CSRF is that the malicious requests are routed to the web application through the user's browser, hence they might be indistinguishable from intended benign requests which were actually authorized by the user. The CSRF does not require the attacker to intercept or modify user's requests and responses: it suffices that the victim visits the attacker's website, from which the attack is launched. Thus, CSRF vulnerabilities are exploitable by any malicious website on the Web.

- **Advantages** – The value of standard HTTP request headers such as Referrer and Origin, indicating the page originating the request.
- The presence of custom HTTP request headers like X-Requested-With, which cannot be set from a cross-site position.
- The presence of unpredictable anti-CSRF tokens, set by the server into sensitive forms.
- **Algorithm**: RandomForestClassifier

SYSTEM ENVIRONMENT

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

CONCLUSION-

Web applications are particularly challenging to analyse, due to their diversity and the widespread adoption of custom programming practices. ML is thus very helpful in the web setting, because it can take advantage of manually labeled data to expose the human understanding of the web application semantics to automated analysis tools. We validated this claim by designing Mitch, the first ML solution for the blackbox detection of CSRF vulnerabilities, and by experimentally assessing its effectiveness. We hope other researchers might take advantage of our methodology for the detection of other classes of web application vulnerabilities.

REFERENCES

[1] Stefano Calzavara, Riccardo Focardi, Marco Squarcina, and Mauro Tempesta. Surviving the web: A journey into web session security. ACM Comput. Surv., 50(1):13:1–13:34, 2017

[2] Avinash Sudhodanan, Roberto Carbone, Luca Compagna, Nicolas Dolgin, Alessandro Armando, and Umberto Morelli. Large-scale analysis & detection of authentication cross-site request forgeries. In 2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017, pages 350–365, 2017.

[3] Stefano Calzavara, Alvise Rabitti, Alessio Ragazzo, and Michele Bugliesi. Testing for integrity flaws in web sessions. In Computer Security - 24rd European Symposium on Research in Computer Security, ESORICS 2019, Luxembourg, Luxembourg, September 23-27, 2019, pages 606–624, 2019.

[4] OWASP. OWASP Testing Guide. https://www.owasp.org/index.php/ OWASP Testing Guide v4 Table of Contents, 2016.

[5] Jason Bau, Elie Bursztein, Divij Gupta, and John C. Mitchell. State of the art: Automated black-box web application vulnerability testing. In 31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berkeley/Oakland, California, USA, pages 332–345, 2010.

[6] Adam Doupe, Marco Cova, and Giovanni Vigna. Why johnny can't ´ pentest: An analysis of black-box web vulnerability scanners. In Detection of Intrusions and Malware, and Vulnerability Assessment, 7th International Conference, DIMVA 2010, Bonn, Germany, July 8-9, 2010. Proceedings, pages 111–131, 2010.

[7] Adam Barth, Collin Jackson, and John C. Mitchell. Robust defenses for cross-site request forgery. In Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008, pages 75–88, 2008.

[8] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. Foundations of Machine Learning. The MIT Press, 2012.

[9] Michael W. Kattan, Dennis A. Adams, and Michael S. Parks. A comparison of machine learning with human judgment. Journal of Management Information Systems, 9(4):37–57, March 1993.

[10] D. A. Ferrucci. Introduction to "This is Watson". IBM Journal of Research and Development, 56(3):235–249, May 2012.

[11] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587):484–489, Jan 2016.

[12] Michele Bugliesi, Stefano Calzavara, Riccardo Focardi, and Wilayat Khan. Cookiext: Patching the browser against session hijacking attacks. Journal of Computer Security, 23(4):509–537, 2015.

[13] Stefano Calzavara, Gabriele Tolomei, Andrea Casini, Michele Bugliesi, and Salvatore Orlando. A supervised learning approach to protect client authentication on the web. TWEB, 9(3):15:1–15:30, 2015.

[14] Stefano Calzavara, Mauro Conti, Riccardo Focardi, Alvise Rabitti, and Gabriele Tolomei. Mitch: A machine learning approach to the blackbox detection of CSRF vulnerabilities. In IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019, pages 528–543, 2019.

[15] Giancarlo Pellegrino, Martin Johns, Simon Koch, Michael Backes, and Christian Rossow. Deemon: Detecting CSRF with dynamic analysis and property graphs. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017, pages 1757–1771, 2017.

Stefano Calzavara is a tenure-track assistant professor at Universita Ca' ` Foscari Venezia, Italy. He received a PhD in Computer Science at Universita` Ca' Foscari Venezia, Italy, in 2013. His main research interests are formal methods and web security. Contact him at calzavara@dais.unive.it.

Mauro Conti is a full professor at University of Padua, Italy. He received a PhD in Computer Science at Sapienza University of Rome, Italy, in 2009. His main research interestes are computer security and privacy. Contact him at conti@math.unipd.it.

Riccardo Focardi is a full professor at Universita Ca' Foscari Venezia, Italy. ` He received a PhD in Computer Science at University of Bologna, Italy, in 1999. His main research interests are computer security and formal methods. Contact him at focardi@unive.it.

Alvise Rabitti is a security officer at Universita Ca' Foscari Venezia, Italy. He ` received a bachelor degree in Computer Science from Universita Ca' Foscari ` Venezia, Italy, in 2013. His main research interests are web security and privacy. Contact him at alvise.rabitti@unive.it.

Gabriele Tolomei is an associate professor at Sapienza University of Rome, Italy. He received a PhD in Computer Science at Universita Ca' Foscari ` Venezia, Italy, in 2011. His main research interests are machine learning and web search.