USING MACHINE LEARNING TO DETECT CROSS-SITE REQUEST FORGERY

MUDDHAM NIRMALA[1],SRIMATHI KESHETTI[2]
[1][2]Assistant Professor
Department Of CSE
Sree Chaitanya College of Engineering, Karimnagar

ABSTRACT:
In this project, we offer an approach that uses machine learning (ML) to identify vulnerabilities in online applications. Because web apps are so diverse and sometimes use bespoke programming techniques, they can be especially difficult to analyze. Because machine learning (ML) can leverage manually labeled data to incorporate human understanding of online application semantics into automated analysis tools, it is thus particularly beneficial for web application security. In order to create Mitch, the first machine learning solution for the black-box detection of Cross-Site Request Forgery (CSRF) vulnerabilities, we used our methodology. We were able to find 35 new CSRFs on 20 popular websites and 3 new CSRFs on production software thanks to Mitch.

I.   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.

II. LITERATURE SURVEY

1) Surviving The Web: A Journey Into Web Session Security

AUTHORS: Stefano Calzavara, Riccardo Focardi, Marco Squarcina, and Mauro Tempesta

The Web is the primary access point to on-line data and applications. It is extremely complex and variegate, as it integrates a multitude of dynamic contents by different parties to deliver the greatest possible user experience. This

3038

heterogeneity makes it very hard to effectively enforce security, since putting in place novel security mechanisms typically prevents existing websites from working correctly or negatively affects the user experience, which is generally regarded as unacceptable, given the massive user base of the Web However, this continuous quest for usability and backward compatibility had a subtle effect on web security research: designers of new defensive mechanisms have been extremely cautious and the large majority of their proposals consists of very local patches against very specific attacks. This piecemeal evolution hindered a deep understanding of many subtle vulnerabilities and problems, as testified by the proliferation of different threat models against which different proposals have been evaluated, occasionally with quite diverse underlying assumptions. It is easy to get lost among the multitude of proposed solutions and almost impossible to understand the relative benefits and drawbacks of each single proposal without a full picture of the existing literature. In this work, we take the delicate task of performing a systematic overview of a large class of common attacks targeting the current Web and the corresponding security solutions proposed so far. We focus on attacks against web sessions, i.e., attacks which target honest web browser users establishing an authenticated session with a trusted web application. This kind of attacks exploits the intrinsic complexity of the Web by tampering, e.g., with dynamic contents, client-side storage or cross-domain links, so as to corrupt the browser activity and/or network communication. Our choice is motivated by the fact that attacks against web sessions cover a very relevant subset of serious web security incidents and many different defenses, operating at different levels, have been proposed to prevent these attacks.

We consider typical attacks against web sessions and we systematise them based on: (i) their attacker model and (ii) the security properties they break. This first classification is useful to understand precisely which intended security properties of a web session can be violated by a certain attack and how. We then survey existing security solutions and mechanisms that prevent or mitigate the different attacks and we evaluate each proposal with respect to the security guarantees it provides. When security is guaranteed only under certain assumptions, we make these assumptions explicit. For each security solution, we also evaluate its impact on both compatibility and usability, as well as its ease of deployment. These are important criteria to judge the practicality of a certain solution and they are useful to understand to which extent each solution, in its current state, may be amenable for a large-scale adoption on the Web. Moreover, since there are several proposals in the literature which aim at providing robust safeguards against multiple attacks, we also provide an overview of them. For each of these proposals, we discuss which attacks it prevents with respect to the attacker model considered in its original design and we assess its adequacy according to the criteria described above.

2) Large-Scale Analysis & Detection Of Authentication Cross-Site Request Forgeries

AUTHORS: Avinash Sudhodanan, Roberto Carbone, Luca Compagna, Nicolas Dolgin, Alessandro Armando, and Umberto Morelli

Cross-Site Request Forgery (CSRF) attacks are one of the critical threats to web applications. In this paper, we focus on CSRF attacks targeting web sites' authentication and identity management functionalities. We will refer to them collectively as Authentication CSRF (Auth-CSRF in short). We started by collecting several Auth-CSRF attacks reported in the literature, then analyzed their underlying strategies and identified 7 security testing strategies that can help a manual tester uncover

vulnerabilities enabling Auth-CSRF. In order to check the effectiveness of our testing strategies and to estimate the incidence of Auth-CSRF, we conducted an experimental analysis considering 300 web sites belonging to 3 different rank ranges of the Alexa global top 1500. The results of our experiments are alarming: out of the 300 web sites we considered, 133 qualified for conducting our experiments and 90 of these suffered from at least one vulnerability enabling Auth-CSRF (i.e. 68%). We further generalized our testing strategies, enhanced them with the knowledge we acquired during our experiments and implemented them as an extension (namely CSRF-checker) to the open-source penetration testing tool OWASP ZAP. With the help of CSRFchecker, we tested 132 additional web sites (again from the Alexa global top 1500) and identified 95 vulnerable ones (i.e. 72%). Our findings include serious vulnerabilities among the web sites of Microsoft, Google, eBay etc. Finally, we responsibly disclosed our findings to the affected vendors.

3) State Of The Art: Automated Black-Box Web Application Vulnerability Testing
AUTHORS : Jason Bau, Elie Bursztein, Divij Gupta, and John C. Mitchell
Black-box web application vulnerability scanners are automated tools that probe web applications for security vulnerabilities. In order to assess the current state of the art, we obtained access to eight leading tools and carried out a study of: (i) the class of vulnerabilities tested by these scanners, (ii) their effectiveness against target vulnerabilities, and (iii) the relevance of the target vulnerabilities to vulnerabilities found in the wild. To conduct our study we used a custom web application vulnerable to known and projected vulnerabilities, and previous versions of widely used web applications containing known vulnerabilities. Our results show the promise and effectiveness of automated tools, as a group, and also some

limitations. In particular, "stored" forms of Cross Site Scripting (XSS) and SQL Injection (SQLI) vulnerabilities are not currently found by many tools. Because our goal is to assess the potential of future research, not to evaluate specific vendors, we do not report comparative data or make any recommendations about purchase of specific tools.

4) Why johnny can't pentest: An analysis of black-box web vulnerability scanners
AUTHORS : Adam Doup´e, Marco Cova, and Giovanni Vigna
Black-box web vulnerability scanners are a class of tools that can be used to identify security issues in web applications. These tools are often marketed as "point-and-click pentesting" tools that automatically evaluate the security of web applications with little or no human support. These tools access a web application in the same way users do, and, therefore, have the advantage of being independent of the particular technology used to implement the web application. However, these tools need to be able to access and test the application's various components, which are often hidden behind forms, JavaScript-generated links, and Flash applications. This paper presents an evaluation of eleven black-box web vulnerability scanners, both commercial and open-source. The evaluation composes different types of vulnerabilities with different challenges to the crawling capabilities of the tools. These tests are integrated in a realistic web application. The results of the evaluation show that crawling is a task that is as critical and challenging to the overall ability to detect vulnerabilities as the vulnerability detection techniques themselves, and that many classes of vulnerabilities are completely overlooked by these tools, and thus research is required to improve the automated detection of these flaws.

5) Mitch: A Machine Learning Approach To The Blackbox Detection Of Csrf

Vulnerabilities AUTHORS: Stefano Calzavara, Mauro Conti, Riccardo Focardi, Alvise Rabitti, and Gabriele Tolomei

Cross-Site Request Forgery (CSRF) is one of the oldest and simplest attacks on the Web, yet it is still effective on many websites and it can lead to severe consequences, such as economic losses and account takeovers. Unfortunately, tools and techniques proposed so far to identify CSRF vulnerabilities either need manual reviewing by human experts or assume the availability of the source code of the web application. In this paper we present Mitch, the first machine learning solution for the black-box detection of CSRF vulnerabilities. At the core of Mitch there is an automated detector of sensitive HTTP requests, i.e., requests which require protection against CSRF for security reasons. We trained the detector using supervised learning techniques on a dataset of 5,828 HTTP requests collected on popular websites, which we make available to other security researchers. Our solution outperforms existing detection heuristics proposed in the literature, allowing us to identify 35 new CSRF vulnerabilities on 20 major websites and 3 previously undetected CSRF vulnerabilities on production software already analyzed using a state-of-the-art tool.

III. SYSTEM ANALYSIS
SYSTEM ARCHITECTURE:
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 OF EXISTING SYSTEM:

➢ 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 OF PROPOSED SYSTEM:

➢ 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 STUDY
FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- •   ECONOMICAL FEASIBILITY
- •   TECHNICAL FEASIBILITY
- •   SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

  This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

IV.   **IMPLEMENTATION:**
  MODULES DESCRIPTION:
- •   User
- •   Admin
- •   False Positives and False Negatives
- •   Machine Learning Classifier

MODULES DESCRIPTION:
User:

The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user register then admin can activate the customer. Once admin activated the customer then user can login into our system. User can do the data preprocess. First required running website name. By using that website the user can test the csrfs. By help of bolt tool the user can fetch related all csrfs and generated algorithm names. The result will be stored in json files. Later the user can get the results of Mitch dataset. The mitch dataset tested for POST method as well GET method to. The result will be displayed on the browser.

Admin:

Admin can login with his credentials. Once he login he can activate the users. The activated user only login in our applications. The admin can set the training and testing data for the project of the Mitch Dataset. The user search all urls related csrf token admin can view in his page. The admin can also check the POST

method performed data from the dataset and GET method related data also.

False Positives and False Negatives:

Mitch produces a false positive when it returns a candidate CSRF that cannot be actually exploited. This is something relatively easy to detect by manual testing, though this process is tedious and time-consuming. In general, it is not possible to reliably identify when Mitch produces a false negative, because this would require to know all the CSRF vulnerabilities on the tested websites. To estimate this important aspect, we keep track of all the sensitive requests returned by the ML classifier embedded into Mitch and we focus our manual testing on those cases. This is a reasonable choice to make the analysis tractable, because we first showed that the classifier performs well using standard validity measures.

Machine Learning Classifier:

The ML classifier used by Mitch was trained from a dataset of around 6000 HTTP requests from existing websites, collected and labeled by two human experts. The feature space X of the classifier has 49 dimensions, each one capturing a specific property of HTTP requests. Those can be organized into following categories.

following set of numerical features:

numOfParams: the total number of parameters;

numOfBools: the number of request parameters bound to a boolean value;

numOfIds: the number of request parameters bound to an identifier, i.e., a hexadecimal string, whose usage was empirically observed to be common in our dataset;

numOfBlobs: the number of request parameters bound to a blob, i.e., any string which is not an identifier;

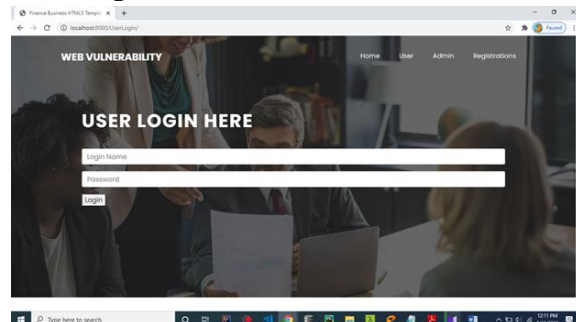reqLen: the total number of characters in the request, including parameter names and values.
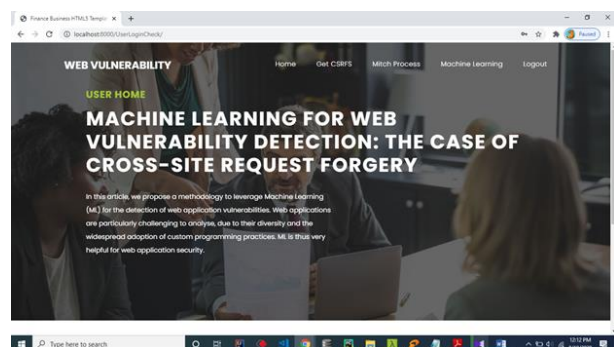
V. SCREEN SHOTS;
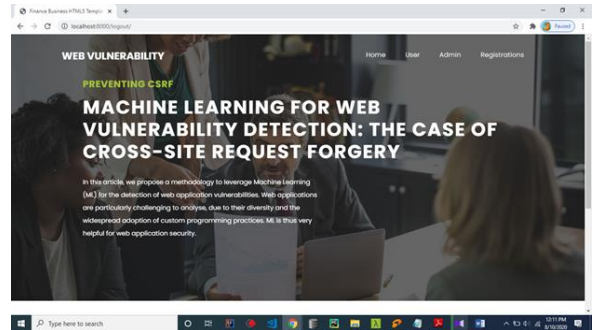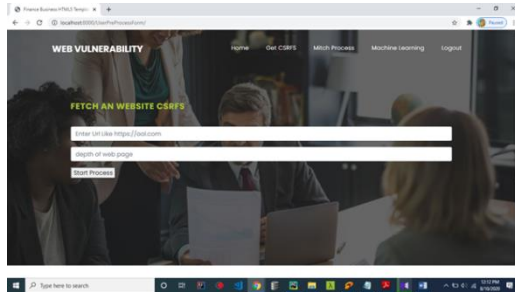

User Registration Form
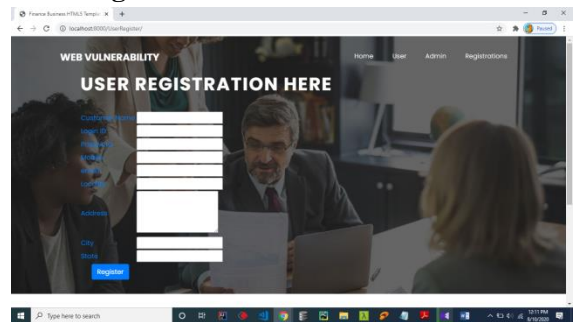
User Login Form:

User Home:

Getting website csrfs:

3043

Scanning urls:

Further Enhancement

This work provides the most up to date and comprehensive account of the nature of the CSRF attacks and corresponding solution to thwart these attacks. However, as the new knowledge pioneered in this work takes hold, the future extensions of this knowledge are bound to happen. Some areas of further work, in future, are identified as follows: As a future extension of this work it may be possible to perform the Bayesian estimation by using 1-99 (configurable) threshold probability ratio of suspect CSRF page to safe pages since anything lower than 1% of the threshold can either be random or an unexpected variance] Incorporating routine CSRF scanning in commercial anti-malware • Browser specific and generic anti CSRF solutions.

VI. CONCLUSION

Because web apps are so diverse and often use specialized programming techniques, they can be especially difficult to analyze. Because machine learning (ML) may leverage manually labeled data to provide the human understanding of online application semantics to automated analysis tools, ML is thus particularly useful in the web scenario. By creating Mitch, the first machine learning solution for the blackbox detection of CSRF vulnerabilities, and evaluating its efficacy through experiments, we have verified this assertion. We expect that our methodology will be useful to other researchers in detecting different types of vulnerabilities in online applications.

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 Doup´e, 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.