X-Secure: Protecting Users from Big Bad Wolves

Robbie Binnie †, Colin McLean † Amar Seeam *, Xavier Bellekens †

*Middlesex University {initial.surname}@mdx.ac.uk
†Abertay University {name.surname}@abertay.ac.uk

***Abstract*—In 2014 over 70% of people in Great Britain accessed the Internet every day. This resource is an optimal vector for malicious attackers to penetrate home computers and as such compromised pages have been increasing in both number and complexity. This paper presents *X-Secure*, a novel browser plug-in designed to present and raise the awareness of inexperienced users by analysing web-pages before malicious scripts are executed by the host computer. *X-Secure* was able to detect over 90% of the tested attacks and provides a danger level based on cumulative analysis of the source code, the URL, and the remote server, by using a set of heuristics, hence increasing the situational awareness of users browsing the internet.**

I. Introduction

With web application based attacks being so prevalent, studies suggest that there are problems with the manner in which web pages are being created and although this is the programmers responsibility to solve, users must remain vigilant. With the vast number of non-technical people accessing web resources, along with technical users becoming overly complacent, a large number of attacks do eventually succeed [1].

Current research focuses on providing security by modifying the server side of the web application [2] [3] [4]. These applications provide good security but are often difficult to set in place and require numerous modifications to non-standard frameworks. *X-Secure* on the other hand, is provided to the user as a Google Chrome browser-plugin and does not require any modification from the server side, whilst providing acute security.

II. Cross Site Scripting

Cross site scripting (XSS) is one of the most common vulnerabilities exploited with 73% of all attacks carried out being of this type [5]. This is also apparent, due to vast amount of available literature identifying the different intrusion types [6] [5] [7].

There are three main types of XSS;

- Reflective
- Persistant
- Document Object Model (DOM)

Reflective attacks are the most common and persistent tend to be the most dangerous. Reflective attacks store malicious JavaScript code within a URL, which can be detected by URL searches. On the other hand, persistent attacks retrieve the malicious code from a database, which incorporates malicious code within the legitimate HTML and JavaScript; as such a set of heuristics have to be created to thwart these attacks efficiently. DOM based attacks focus on modifying the DOM environment in the browser, in order for the code to be run in a malicious manner and infect the user.

In order for an attack to take place, a malicious user must artifice the server/browser into thinking the information being sent to it is legitimate code. For this to occur, the HTML tag $<script>$ is used.

Under normal execution when the browser obtains this tag, everything after it is executed until the tag is closed. Once the server incorporates the string either from the database or the URL the browser has no way of deciphering if the script tag is legitimate or contains dangerous code. Hence if this tag is injected, the browser will believe it to be legitimate code.

However, not all XSS require a script tag to be present. It is also possible to include scripts within $<a\ href=>$ tags; however, it will have to be clicked in order for the code to be executed. If the text javascript: is present and the code is entered within the href attribute of the $<a\ ...>$ tag it will be rendered as regular code when clicked. An example is shown in Figure 2.

```
<A HREF="javascript:document.location=
'http://www.google.com/'">XSS</A>
```

Fig. 2. Redirection Attack Using Persistant Cross-Site-Scripting

The code shown in Figure 2 aims at redirecting the user to Google when the link is clicked. This type of attacks often evades filter detection, as they are signature based attacks, such as in intrusion detection systems [8]. A possible solution against this is proposed by Hodo et al. by using neural networks to detect anomalies in networks [9], a similar solution is presented for SQL injections by Sheykhkanloo et al. in [10].

The *onerror* attribute of HTML image tags are also able to contain scripts, in order for these to execute, the script

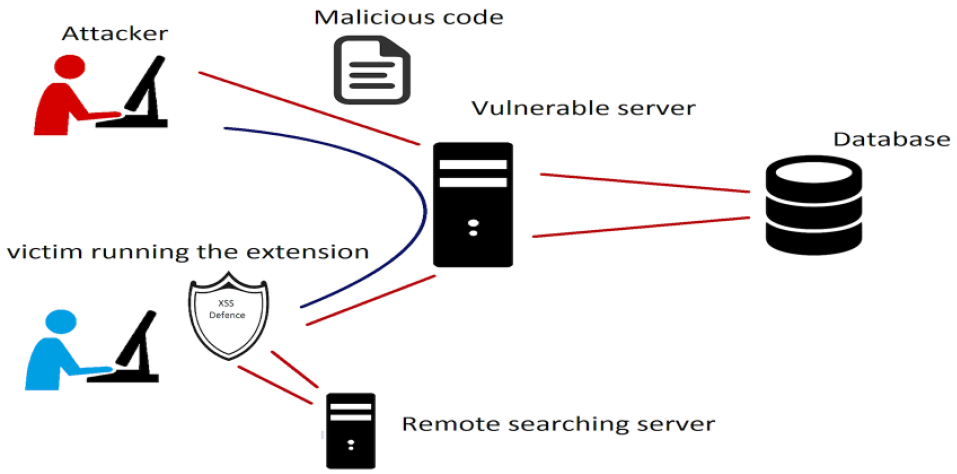

Fig. 1. *X-Secure* Protecting Users from Malicious Persistent and Reflective XSS Attacks

does not require any tags or the text *script* to be present. It also does not require the user to click any link. When an image is requested but not obtained the browser will check for the *onerror* attribute of the tag, if a malicious attacker has injected XSS to this attribute, the malicious code will automatically be executed by the browser.

A. Heuristics

Regular expressions have become more common within computing and as such many developers have invested significant time improving their usability, bordering on allowing natural language processing. Although many of the improvements with regular expressions have focused on text processing, it is also be possible to use these improvements to help detect malicious code within web pages, as demonstrated in [11] and [12].

III. Methodology

X-Secure consists of four main components. These are designed with efficacy and usability in mind in order to provide the user with a simple interface while drastically increasing its safety. Figure 1 (Red) Demonstrates how *X-Secure* is able to deflect persistent XSS attacks and reflective attacks (Blue) by making use of its four main components.

a) Application Controller:: The first is the controller, which is used to run all of the individual threat detecting techniques and correlate the results. The controller is also used to communicate with the user interface. The controller also provides a danger rating to the user, based on the number and types of attacks found in a single web page. This functionality allows tailored danger ratings. Moreover a blocking box is also provided to the user, allowing a notification to be displayed to the user before any suspicious code is executed within the page, inherently preventing the user to get harmed.

b) URL Scanning:: Another key component of *X-Secure* is the function that will determine if an attack is present within the URL. This component takes a URL string and searches it for different signatures. Originally a centralised JavaScript Object Notation (JSON) string was used to store the list characteristics; however, loading the data from another file was detrimental to performance and as such a pre-defined array within the script is now used. The array is based on the OWASP cheat sheet [13] which contains a list of all possible strings that contain the following character $<$. Scanning the URLs for this character allows reflective cross site scripting attacks to be prevented [14].

Using the Chrome extension manifest it is possible to run a section of code on a webpage before the rest of the webpage is executed. Running this component at *document_start* guarantees that the URL can be scanned before any code is executed as shown in Figure 3.

c) Local Analysis:: In order to detect persistent attacks a search component was developed. To avoid synchronicity issues the component was created to operate locally on the host computer; increasing the throughput of the search engine. The search component makes use of numerous regular expressions in order to detect persistent cross site scripting

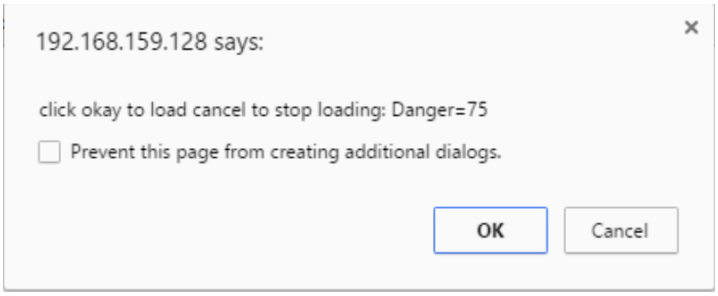

Fig. 3. Pop-up executing before the Malicious Javascript is Executed

TABLE I
ATTACK TYPES, AND DANGER RATING LEVEL PROVIDED TO THE USER.

Web Page ID	Attack Type	Attack Code	Detected	Defeated	Danger Level
1	URL scan	*data=alert("hello world")*	Yes	Yes	65
1	URL scan	*data= <script >alert("hello world") </script >*	Yes	Yes	65
1	URL scan	*data=%3Cscript >alert("hello world")%3C/script >*	Yes	Yes	65
2	Local search	*window.open("hello" + document.cookie)*	Yes	Yes	65
3	Local search	*<img src="image1.jpg" + document.cookie/ >*	Yes	Yes	75
2	Local search	*$.ajax(type: "post", url:,192.168.123.2, data:data);*	Yes	Yes	37
2	Remote,Search	*<img src="192.168.1.13/image1.jpg"+OtherVar/ >*	Yes	Yes	2
3	Remote,Search	*<img src="image1.jpg" + document.cookie/ >*	Yes	No	20
4	Remote,Search	*window.open("test.php?cookie=",+ document.cookie)*	No	No	0
4	Remote,Search	*<img src="image1.jpg" + document.cookie/ >*	Yes	No	22

attacks present within the source code.

In order to stop the attack before it has executed, the full DOM requires a thorough scan; however; the full DOM is only available after all of the code within it has run. To circumvent this behavior, our application makes use of a JQuery function, allowing a synchronous request of a web page, hence our application is able to scan the webpage before being displayed, and the attack being executed.

X-Secure currently focuses on three main type of attacks: AJAX post cross domain; Opening window with *document.cookie* passed as an argument and loading an image with *document.cookie*. These were selected due to their severity, and their potential to allow a malicious user to gain authentication cookies. Moreover *X-Secure* is able to detect IP addresses with the code increasing the danger level provided by our application as malicious attacks URL often contain a server IP address rather than a domain name [15].

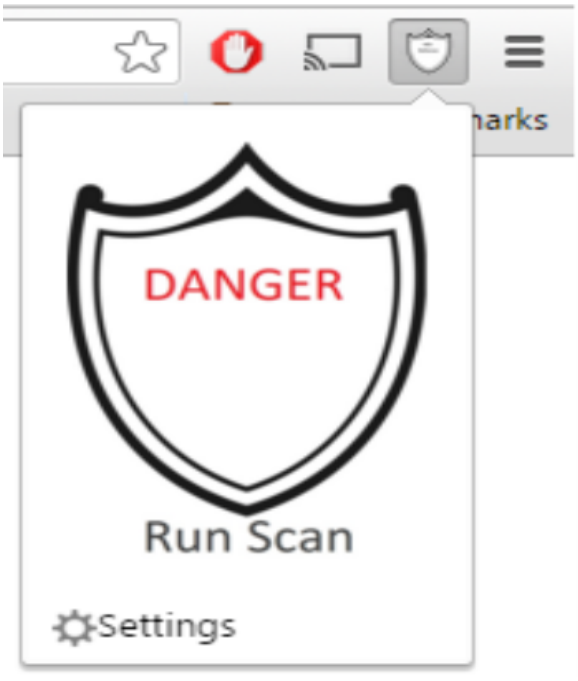

Fig. 4. Raising the Situational Awareness of the User through Graphic Design

d) Remote Update:: X-Secure is provided with a server analysis component allowing the extension to stay up to date at all times. The remote analysis automatically includes new attacks detected to the database engine by providing the new signatures directly to *X-Secure*. This can be enabled by the user in the options. This feature allows improvement of the services provided and will increase the number of signatures on the server side without having to modify the application source code. This technique is used by numerous application such as Adblock, through their filter lists [16].

IV. PRELIMINARY RESULTS

In order to demonstrate the resilience of *X-Secure* four web pages exploiting different vulnerabilities have been created. These allow *X-Secure* to be analysed against different test cases and provide the authors with feedback.

- The first webpage contained a non-persistent attack, in which data passed to the URL is insecurely echoed onto the page.
- The second page is used to simulate potential persistent attacks, making use of a hard coded section that has the characteristic of an attack.
- The third page is a dynamic web page using a database with no input sanitisation allowing the tester to create custom attacks and check the effectiveness of the extension.
- The fourth page is used to test the server side analysis by posting the DOM of a custom web page.

Table I provides preliminary results for different types of attack. As shown, a danger value is returned along with the result (blocked/ not blocked). Moreover the attack code is also provided along with the type of analysis provided by our application.

Table I demonstrates that the application is able to detect over 90% of the attacks including unknown attacks and provides a danger accuracy for most of the attacks. It also demonstrates that our application is able to avoid 70% of the attacks before execution.

The remote search was unable to detect attack 9 due to not having any regular expression associated with this attack being present in the database; however, once the database was updated this attack was detected and reported to the user. This attack was only run and included within the results to demonstrate that no code has to be changed in order to improve the capabilities of the searching function.

When used on a larger website the application demonstrated a drop in performances, this lead to the decision modifying the core of the search engine and provide the application with asynchronous results.

This core modification leads to a performances increase, however it also increases the risk of successful attacks. The trade-of however provided the user with seamless browsing, whilst keeping them informed on the health status of the webpage with only a slight delay.

In order to increase the longevity of the application, an external database is used to provide *X-secure* with new heuristics, and increase its awareness and detection capabilities.

Moreover *X-secure* improves the threat analysis and increases the danger level based on previous attacks detected. As previous attacks provide similar heuristics *X-secure* is able to raise the danger level of a page based on its current knowledge and assessment of the page, hence providing the user with a semi-intelligent application whilst protecting him from external threats. The application also provides a visual level of danger allowing to raise the users situational awareness as shown in Figure 4

V. Conclusion

This paper presented *X-Secure*, a novel cross site scripting extension for the Chrome browser, providing the user with a higher level of security when accessing the internet and can evade reflective, persistent and DOM based attacks.

In this work *X-Secure* was evaluated against numerous different attacks and was provided with local and remote scanning capabilities. The extension is also able to provide the user with a danger level, ultimately raising the user awareness. *X-Secure* demonstrated a high level of accuracy as well as learning capabilities. In future work, it is planned to increase the remote database with user feedback as well as implementing a Bayesian classifier in order to detect new attacks and increase the accuracy. The extension will also be released in the Google Chrome Web Store.

References

[1] D. Stuttard and M. Pinto, *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*. John Wiley & Sons, 2011.

[2] P. Bisht and V. N. Venkatakrishnan, *Detection of Intrusions and Malware, and Vulnerability Assessment: 5th International Conference, DIMVA 2008, Paris, France, July 10-11, 2008. Proceedings*, ch. XSS-GUARD: Precise Dynamic Prevention of Cross-Site Scripting Attacks, pp. 23–43. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[3] P. Wurzinger, C. Platzer, C. Ludl, E. Kirda, and C. Kruegel, "Swap: Mitigating xss attacks using a reverse proxy," in *Proceedings of the 2009 ICSE Workshop on Software Engineering for Secure Systems*, pp. 33–39, IEEE Computer Society, 2009.

[4] J. Weinberger, P. Saxena, D. Akhawe, M. Finifter, R. Shin, and D. Song, "A systematic analysis of xss sanitization in web application frameworks," in *Computer Security–ESORICS 2011*, pp. 150–171, Springer, 2011.

[5] S. Fogie, J. Grossman, R. Hansen, A. Rager, and P. Petkov, *XSS Attacks: Cross Site Scripting Exploits and Defense*. Elsevier Science, 2011.

[6] S. McClure, J. Scambray, and G. Kurtz, *Hacking Exposed 7: Network Security Secrets and Solutions*. Hacking Exposed, McGraw-Hill Education, 2012.

[7] A. Y, *The Art of Hacking*. Lambert Academic Publishing, 2012.

[8] X. J. Bellekens, C. Tachtatzis, R. C. Atkinson, C. Renfrew, and T. Kirkham, "A highly-efficient memory-compression scheme for gpu-accelerated intrusion detection systems," in *Proceedings of the 7th International Conference on Security of Information and Networks*, p. 302, ACM, 2014.

[9] E. Hodo, X. Bellekens, A. Hamilton, P.-l. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of iot networks using artificial neural network intrusion detection system," in *Networks, Computers and Communications, The 2016 International Symposium on*, May 2016.

[10] N. M. Sheykhkanloo, "Employing neural networks for the detection of sql injection attack," in *Proceedings of the 7th International Conference on Security of Information and Networks*, SIN '14, (New York, NY, USA), pp. 318:318–318:323, ACM, 2014.

[11] X. J. A. Bellekens, C. Tachtatzis, R. C. Atkinson, C. Renfrew, and T. Kirkham, "Glop: Enabling massively parallel incident response through gpu log processing," in *Proceedings of the 7th International Conference on Security of Information and Networks*, SIN '14, (New York, NY, USA), pp. 295:295–295:301, ACM, 2014.

[12] G. Wassermann and Z. Su, "Static detection of cross-site scripting vulnerabilities," in *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*, pp. 171–180, IEEE, 2008.

[13] "XSS filter evasion cheat sheet." https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet. Accessed: 2016-03-30.

[14] X.-h. Zhang and Z.-j. Wang, "Notice of retraction a static analysis tool for detecting web application injection vulnerabilities for asp program," in *e-Business and Information System Security (EBISS), 2010 2nd International Conference on*, pp. 1–5, IEEE, 2010.

[15] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1245–1254, ACM, 2009.

[16] A. K. Singh and V. Potdar, "Blocking online advertising-a state of the art," in *Industrial Technology, 2009. ICIT 2009. IEEE International Conference on*, pp. 1–10, IEEE, 2009.