Prevention of Cross Site Scripting with E-Guard Algorithm

M. James Stephen
Associate Professor,
Dept. of I.T, ANITS,
Visakhapatnam, INDIA

P.V.G.D. Prasad Reddy
Professor, Dept. of CS&SE
Andhra University,
Visakhapatnam, INDIA

Ch. Demudu Naidu
Sr.Asst. Professor,
Dept. of I.T, ANITS,
Vizag, INIDA

Ch. Rajesh
Asst. Prof,
Dept. of I.T
Vizag, INIDA

ABSTRACT
In this world of networking where people around the globe are connected, Cross-site Scripting (XSS) has emerged to one of the most prevalent growing threat. XSS attacks are those in which attackers inject malicious codes, most often client-side scripts, into web applications from outside sources. Because of the number of possible injection location and techniques, many applications are vulnerable to this attack method. Even though the main reason for the vulnerability primarily lies on the server side, the actual exploitation is within the victim's web browser on the client side.

In this paper, we propose a passive detection system to identify successful XSS attacks. Based on a prototypical implementation, we examine our approach's accuracy and verify its detection capabilities. We compiled a data-set of HTTP request/response from 20 popular web applications for this, in combination with both real word and manually crafted XSS exploits; our detection approach results in a total of zero false negatives for all tests, while maintaining an excellent false positive rate for more than 80 percent of the examined web applications.

General Terms
Cross Site Scripting, E-Guard Algorithm, False positive, False Negative, White list, Black list, Gray list.

Keywords
XSS attack, Enhanced XSS Guard algorithm, E-Guard, Server-side detection, Client-side detection.

1. INTRODUCTION
For the last few years, Cross-site Scripting (XSS) in web applications had become one of the most prevalent types of security vulnerabilities [1]. SQL Injection affect the server side, but where as XSS attacks do not affect the server side but clients: the actual exploitation is within the victim's web browser. So, the operator of a web application has only very limited evidence of successful XSS attacks. XSS related problems are therefore often overlooked and sometimes they are recognized very late.

In this paper, we propose a server side Cross-site Scripting Detection System (XSSDS); And our approach is based on well known HTTP traffic monitoring (fig.1) and relies upon the following two observations:

1. There is a strong correlation between reflected XSS issues and the incoming parameters.

2. The set of all genuine java scripts in a web application is bounded.

This gives the basis for two detection approaches to identify effectively carried out reflected XSS attacks and to determine stored XSS code. Our approach does not require any changes to the actual application: Both attack detection methods depends exclusively on access to the HTTP traffic. Our approach is therefore applicable to all current web application technologies i.e., web servers and applications.

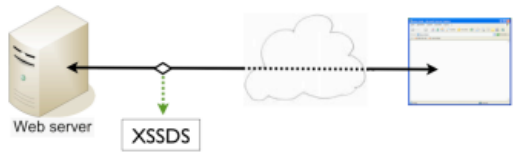

Figure 1. Passive XSS attack detection

2. TECHNICAL BACKGROUND
The term Cross site Scripting (XSS) [2] describes a class of string based code injections on web applications. XSS can be classified in three different types: reflected, stored, and DOM based XSS:

- Reflected XSS [3] is perhaps simple to understand. Part of webpage is dynamically created using content supplied by the user and the exploit code is contained within the user input.
- Stored XSS [3] in servers makes them vulnerable, when they store user supplied input in a server side data repository such as a file or a database. These attacks are particularly insidious because they have the potential to affect anybody who visits the site and to whom the content is displayed.
- DOM or Document Object Model XSS [3] exploits the document object model. DOM exposes this object model so that scripts can access the content and the metadata for the WebPages. Upper most in the DOM hierarchy for example is the document object which also provides access to other objects such as document. Location and the vulnerability occurs when these objects can be indirectly manipulated.

3. DETECTION MECHANISMS
3.1 Reflected XSS through direct data inclusion
There are largely two distinct countermeasures for XSS prevention in real life web applications: input filtering and output sanitation. Input filtering describes the process of validating all incoming data. Suspicious input that might contain

a code injection payload is either rejected, encoded or the offensive parts are removed using removal filters. The protection approach implemented by these filters relies on removing predefined keywords, such as <script, javascript , or document. Such filtering approaches are, however, error prone due to incomplete keyword list or non recursive implementations [4]. If output sanitation is employed, certain characters, such as <, “, or’ , are HTML encoded before user supplied data is inserted into the outgoing HTML. Both of the above protections are known to frequently fail, either through erroneous implementation or because they are not applied to the complete set of user supplied data.

3.2 Reflected XSS detection by request/response matching

Our detection mechanism for reflected XSS is based on the observation that reflected XSS implies a direct relationship between the input data(e.g., HTML parameters) and the injected script. More precisely: the injected script is fully contained both in the HTTP request and the response. Reflected XSS should therefore be detectable by simply matching incoming data and outgoing java script using an appropriate similarity metric. It is crucial to emphasize that we match the incoming data only against script code found in HTML. Non script HTML content is ignored for our script extraction technique.

3.3 Install online Enhanced-XSS software in user’s computers:

Despite all the above efforts, it is still possible for the users to visit the spoofed Web sites. As a last defense, users can install Enhanced XSS guard in their computers. The Enhanced XSS guard we have developed divides the list of available websites into three categories: blacklist/whitelist based and grey-based.

• Category I: When a user visits a Web site, the Enhanced XSS guard scans the page source of that website and recognizes the scripts present in that site and checks whether these scripts matches with the black list scripts stored in the database. If it is found to match with the blacklist scripts, then the user is warned about the circumstances.

• Category II: When a user visits a Web site, the Enhanced XSS guard scans the page source of that website and recognizes the scripts present in that site and checks whether these scripts matches with the white list scripts stored in the database. If it is found to match with the white list scripts, then the page is forwarded to the respective domain.

• Category III: When a user visits a Web site, the Enhanced XSS guard scans the page source of that website and recognizes the scripts present in that site and checks whether these scripts matches with both the white list and blacklist scripts stored in the database. If it is found to match equally with both the scripts, then the web page is named as a grey site and it can be tested later for judging its kind.

4. XSS- A GROWING THREAT

In spite of all the existing mechanisms XSS attacks are still remained as a growing threat [7] and all computer users have to be aware of this. XSS attacks are those where an attacker can exploit the code of a webpage stored in the server. By doing so the attacker can successfully perform the tasks such as redirecting to his own webpage or collect the user’s credible information by the use of cookies.

The seriousness of XSS can be estimated by its recent attack on REDDIT [5], a famous social networking site on September 2009. The XSS worm first was created when a Reddit user posted a malicious script as a comment to a widely read story on the site, Mikko Hypponen, chief research officer at anti-virus firm F-Secure. It quickly spread when users hovered their mouse over text in a comment, which invoked a command to send further comments to other Reddit threads.

"People reading comments ended up sending massive amounts of new comments to Reddit threads.” Jeremy Edberg, senior product developer at Reddit, explained that the worm's author actually took advantage of two bugs that enabled him to perpetrate the infection. One of the flaws could be exploited by placing an MD5 hash function at the end of every comment.

Reddit is just the latest social networking site to fall victim to a XSS attack. Twitter experienced a similar incident in April.

5. THE ENHANCED XSS GUARD ALGORITHM

Enhanced XSS guard (in short we call E-Guard) works by analyzing the probability of scripts matched with white listed and black listed sites.. The algorithm is illustrated in Fig.4

5. 1 Enhanced XSS Guard Algorithm

The following terminologies are used in the algorithm.

```
WL: Whitelist;
BL: Blacklist;
GL: Greylist;
int XSSGuard(GL} {
Read every line from the console(GL)
If( character==’<’){
Increment the array of scripts
Until character==’>’
}
Print number of scripts in the input file.
Read the input file from the console
//checking with white listed scripts
If ( scripts in GL   matches with scripts in WL){
Increment the array of scripts
Print the matched scripts
Print number of matched scripts
}
//checking with black listed scripts
Read the input file from the console
If ( scripts in GL   matches with scripts in BL){
Increment the array of scripts
Print the matched scripts
Print number of matched scripts
}
If(number of matched scripts with WL>BL){
Print the site as WL and redirect it to the respective website
}
Else if(number of matched scripts with BL>WL){
Print the site as BL
Print warning message to the user
}
```

Else if(number of matched scripts equals both BL && WL){
Print check this site status later
}

Fig. 2. Description of the E-XSSGuard algorithm.

The Enhanced XSS guard algorithm works as follows. Initially, we set whitelist, blacklist and graylist to empty in our experiments. Then we manually enter some known scripts of both white and black type in to the database. We refer to all the webpages basing on the scripts they contain i.e if the website contain more white scripts , then it is listed as white and if the website contains more number of black scripts, then we list it as black. When we encounter a new website which is not in both the lists, then the algorithm of E-*XSS Guard*, scans the page source of the grey site and prints the total number of scripts in it. Then it checks the scripts with the whitelisted and blacklisted scripts and its judges the website on the basis of number of scripts matches with white and black list sites. If the number of script matched with white list are more, then we add that site to white list. If the number of script matched with black list are more, then we add that site to the list of black. If there are equal number of scripts matched with both white and black list, then we add that site to the new list named Grey list, which is left un-judged. After a few entries of new websites, this website will be examined once again. And based on the result we list the website accordingly.

Initially, when we check the websites we recognize the scripts in them and list them on the basis of Enhanced XSS guard algorithm. Every time we visit the webpage the XSS algorithm runs and checks whether any new scripts are added to that webpage or not. If the number of scripts remains same, then there will be no change in the listing we made. If there are additional scripts in the webpage, then we consult the owner of that site for the information regarding on the additional scripts. If the owner did not add any scripts, then we can consider that the webpage is hacked by someone and we have to take appropriate action according to the algorithm of Enhanced XSS guard.

The following Figures shows the clear picture of the main routine of Enhanced XSS Guard and graylist implementation.

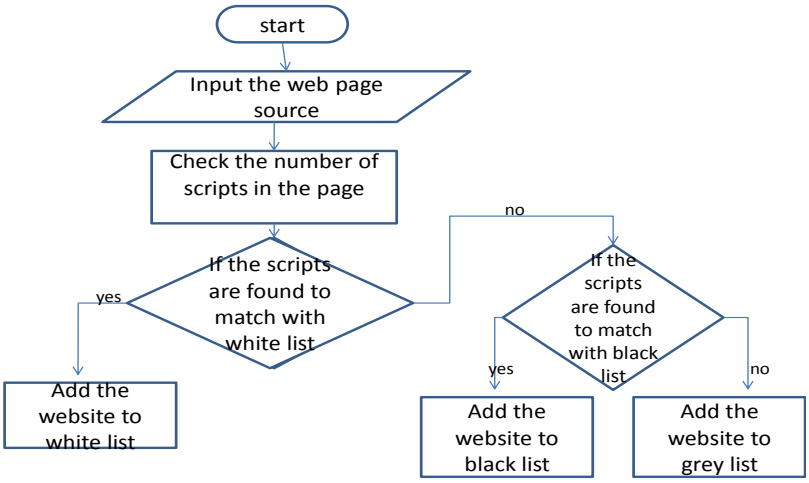

Fig. 3. Flowchart of working of EXSS GUARD

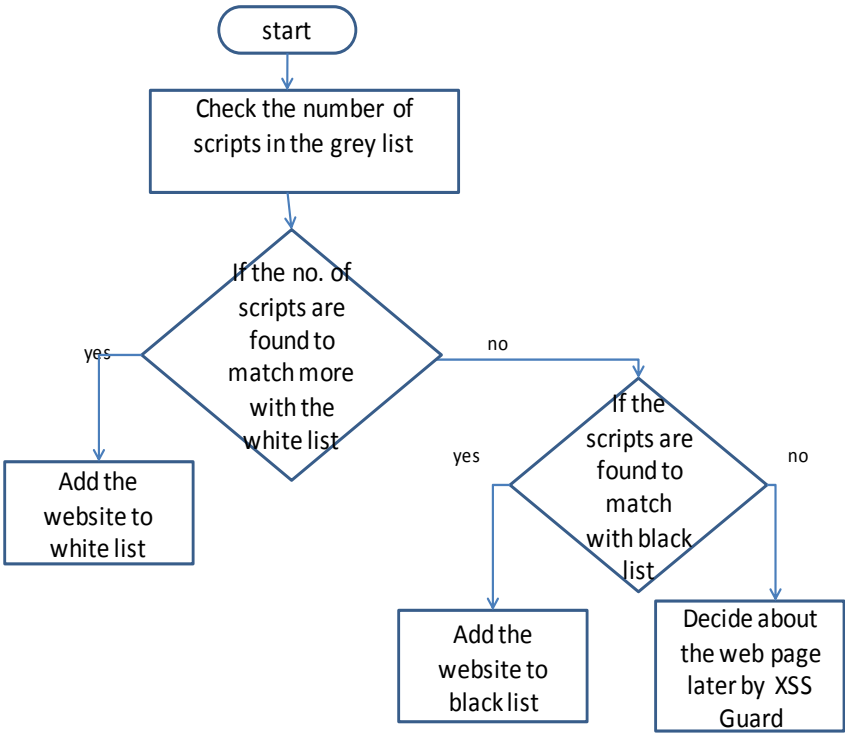

Fig. 4. Flowchart of working of EXSS GUARD

5.2 Handling of False positives and false negatives

Since Enhanced XSS guard is a rule-based heuristic algorithm, it will not cause false positives [6] (i.e., treat non-XSS site as XSS site) and may contain some false negatives (i.e., treat XSS site as non-XSS site). In what follows, we show that Enhanced XSS guard may result in no false positives but is very unlikely to cause some false negatives.

In this paper, we study the common procedure of XSS attacks and review possible approaches. We then focus on end-host based anti-XSS approach. Enhanced XSS guard is character-based, it can detect and prevent not only known XSS attacks but also unknown ones. We have implemented Enhanced XSS guard in Windows XP, and our experiments indicate that Enhanced XSS guard is light-weighted in that it consumes very little memory and CPU circles, and most importantly, it is very effective in detecting XSS attacks with minimal false negatives. Enhanced XSS guard detects about 96% of XSS archives provided by APWG without knowing any signatures of the attacks.

6. IMPLEMENTATION AND VERIFICATION OF ENHANCED XSS GUARD

We have implemented the E-XSS guard algorithm (in short we call E-Guard) in Windows XP. E-XSS guard executive consists grey list along with blacklist and whitelist.

Enhanced XSS guard is the key component of the implementation. It is a standalone windows program with GUI (graphic user interface).). It's composed of 4 parts as illustrated in Fig. 7: Analyzer, Alerter, Logger, and Database.

Implementation of Enhanced XSS guard is as follows:

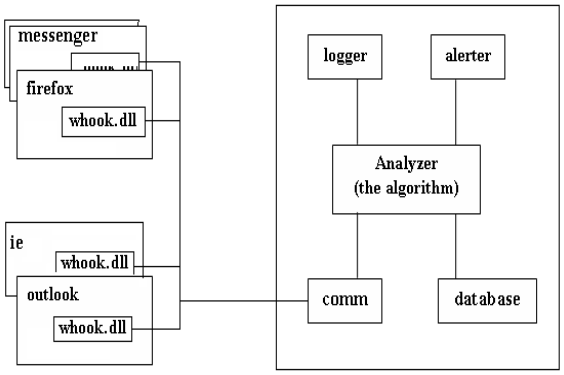

Fig.5 Architecture diagram of EXSS Guard

Database: Store the whitelist, blacklist, graylist and the user input URLs.

Analyzer: It is the key component of Enhanced XSS guard, which implements the Enhanced XSS guard algorithm,. It uses

data provided by Comm and Database, and sends the results to the Alert and Logger modules.

Alerter: When receiving a warning messages from Analyzer, it shows the related information to alert the users and send back the reactions of the user back to the Analyzer.

Logger: Archive the history information, such as user events, alert information, for future use. After implemented the Enhanced XSS guard system, we have designed experiments to verify the effectiveness of our algorithm.

7. RESULTS

Since we are interested in testing Enhanced XSS guard's ability to detect unknown XSS attacks, we set whitelist, blacklist and graylist to empty in our experiments. Our experiments showed that Enhanced XSS guard can detect about 96% XSS attacks. Our experiment also showed that our implementation uses small amount of CPU time and memory space of the system. In a computer with 1.6G Pentium CPU and 512MB memory, our implementation consumes less than 1% CPU time and its memory footprint is less than 7MB. We are planning to use Enhanced XSS guard in daily life to further evaluate and validate its effectiveness. Since we believe that a hybrid approach may be more effective for XSS defense, we are also planning to include a mechanism to update the blacklist and whitelist in real-time and also implementing graylist in its fullness.

8. CONCLUSION

It is becoming increasingly common to tune in to the news or load your favorite news Web site and read about yet another Internet scam. So XSS has becoming a serious network security problem, causing financial lose of billions of dollars to both consumers and e-commerce companies. And perhaps more fundamentally, XSS has made e-commerce distrusted and less attractive to normal consumers. So We designed an XSS algorithm, Enhanced XSS guard, based on the derived characteristics. Since XSS -Guard is characteristic based, it can not only detect known attacks, but it is also effective to the unknown ones. We have implemented Enhanced XSS guard for Windows XP. Our experiment showed that Enhanced XSS guard is light-weighted and can detect up to 96% unknown XSS attacks in real-time.

We believe that the Enhanced XSS guard will be useful for detecting XSS attacks in Web pages and eventually build confidence to use e-commerce without any fear of threat.

8.1 Future Scope

As this application is portable, flexible and light weighted we can provide this to the Internet Service Provider's(ISP's) so that the problem of XSS can be handled at the ground level itself. Also that if there is enough memory in the modem's and if it is technically feasible to implant applications in them, then we can provide these modem's with this Enhanced XSS guard application.

9. REFERENCES

[1] S. Christey and R.A Martin. Vulnerability type distributions in cve, version 1.1. [online], http://cwe.mitre. Org/documents/vuln-trends/index.html,(09/11/07), may 2007.

[2] A. Klien. Cross site scripting explained. White paper, sanctum security group, http:// crypto.stanford.edu/cs155/css.pdf, june 2002.

[3] Cross site scripting techniques and mitigation by CESG revision 1.0, October 2007.

[4] Blwood. Multiple xss vulnerabilities in tikiwiki 1.9.x.mailing list BUgtraq, http://www.security focus.com/archive/1/435137/30/120/threaded, may 2006.\

[5] SC magazine on the article Redditt Succumbs then cleans up from XSS Attack by Dan Kaplan dated September 28,2009.

[6] False Positive defined at Virus list.com .

[7] How serious are XSS threats http://doteduguru.com/id3067.