FlashOver: Automated Discovery of Cross-site Scripting Vulnerabilities in Rich Internet Applications

Steven Van Acker, Nick Nikiforakis, Lieven Desmet, Wouter Joosen, Frank Piessens
IBBT-Distrinet, Katholieke Universiteit Leuven, 3001 Leuven, Belgium
Steven.VanAcker@cs.kuleuven.be

1. EXTENDED ABSTRACT

The last fifteen years have transformed the Web in ways that would seem unimaginable to anyone of the "few" Internet users of the year 1995 [8]. What began as a simple set of protocols and mechanisms facilitating the exchange of static documents between remote computers is now an everyday part of billions' of users life, technical and non-technical alike. The sum of a user's daily experience is composed of open standards, such as HTML, JavaScript and Cascading Style Sheets as well as proprietary plugins, such as Adobe's Flash [1] and Microsoft's Silverlight [6].

Adobe's Flash is the most common way of delivering Rich Internet Applications to desktop users, with the latest statistics revealing an almost complete market penetration of Flash on desktop computers [3, 7]. While some have claimed that the new version of HTML, HTML5 [4], contains enough functionality to render the use of Flash obsolete, the reality is that today most Rich Internet Content, ranging from advertising banners and video players to interactive photo galleries and online games, is served and consumed by the Flash platform.

This rapid evolution of the Web was not left unnoticed by attackers. Traditionally, attackers preferred attacking the server-side of the Internet infrastructure, such as Web servers [5] and mail servers, since that gave them access to powerful hosts with plenty of bandwidth and disk space as well as a foothold in a company's internal network. Nowadays however, the attacks are targeting the client-side of the Internet infrastructure. This can be the Web application, as rendered in a browser, the software of the browser itself or even the user sitting behind the browser. The result of client-side attacks is usually the theft of user credentials or the download of malware that makes the user's computer an unwilling part of a botnet [2].

Since Flash is part of all the technologies that shape the every day experience of Web users, it is also part of this new attack surface. Attacks against Flash target either vulnerabilities in the code of the Flash platform itself, or the insecure practices of developers of Flash applications. In this second category falls the problem of Cross-site Scripting (XSS) [9]. While XSS in Web applications is a well-known and extensively researched problem, the problem of performing Cross-site Scripting attacks through vulnerable Flash applications has received much less attention. A Flash application can interact with the DOM (Document Object Model) of the page that embeds it or even with the browser itself. This allows Flash developers to read information from the page that embeds them, write information to the DOM or redirect the user to a desired page, such as the redirection that happens when a user clicks on a Flash advertisement banner. If these interactions are not protected adequately, an attacker can inject arbitrary JavaScript code that will be executed by a victim's browser in the context of the website hosting the vulnerable Flash application. Such code can, among others, steal a user's session identifier, access the website's local storage on a victim's browser or, in some cases, read the victim's geolocation information.

In this paper we present FlashOver[1], a system capable of automated detection of Cross-site Scripting vulnerabilities in Flash applications. As the name of our system implies, its goal is to discover ways to perform malicious interactions between a Flash application and the rendering browser, that were never intended by the programmer of the vulnerable application. Given a Flash application, FlashOver performs static analysis in order to automatically identify ActionScript variables that can be initialized with user-input and are also used in operations that are commonly prone to code injection attacks. The identified variables are then tested dynamically in order to discover actual vulnerabilities present in the audited Flash application.

More specifically, our FlashOver prototype first decompiles the byte-code representation of ActionScript (the scripting language of the Flash platform) and then performs static analysis on the source code of the application, in search for commonly misused function calls that are responsible for *Flash-to-DOM* and *Flash-to-Browser* communication. Once these functions are located, our system then tracks the arguments of these function calls back to their initialization. When this process is complete, the static-analysis component FlashOver produces a list of variables which are utilized in commonly misused ActionScript API calls and are initialized using user-input. This list of *potentially exploitable variables* is then used by the dynamic-analysis component of our system, which renders the Flash application in the Firefox browser and initializes the variables in many pos-

ASIACCS '12, May 2–4, 2012, Seoul, Korea.

[1] *flashover*: An unintended electric arc, as between two pieces of apparatus

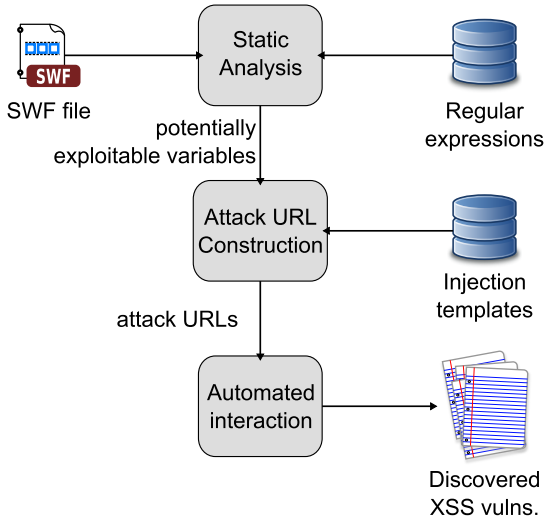

Figure 1: Schematic overview of our FlashOver prototype.

Variable Name	Instances found	Percentage
clicktag	101	35.31%
pageurl	97	33.92%
click	26	9.10%
counturl	10	3.50%
gameinfo	8	2.80%
link1	7	2.44%
url	3	1.05%
link04	2	0.70%
downloadaddress	2	0.70%

Figure 2: Top ten most commonly-named vulnerable variables found in our experiment

sible ways, always mimicking the methodology of attackers who would lure victims in a page under their control. In the last phase, the automatic clicking module of FlashOver clicks thousands of times on the rendered application, with the intent of triggering the vulnerable API call. If our system detects the execution of the injected JavaScript, then the Flash application is flagged as vulnerable. For our prototype, we purposefully chose to implement a minimalistic version of FlashOver to investigate the level of effort and skill required by an attacker to automatically detect XSS vulnerabilities in SWF files.

To evaluate FlashOver, we obtained a partial list of Flash applications hosted on the top 1,000 sites of the Internet, which we downloaded and provided as input to our prototype. Each of the supplied Flash applications was analyzed multiple times by our FlashOver prototype, so that false-negatives due to randomization issues could be minimized. The whole experiment lasted five days. At the end of the experiment, FlashOver successfully detected exploitable XSS vulnerabilities in Flash applications hosted on 64 domains of the top 1,000 Alexa domains, including many well-known websites, including `ebay.com`, `skype.com`, `mozilla.org` and `apple.com`.

Figure 2 shows the ten most commonly named vulnerable variables that we discovered in our analysis. Interestingly, the two most commonly vulnerable variables are responsible for more than 69% of all vulnerabilities found.

The results of our experiment show that the required effort and skill to automatically discover XSS vulnerabilities in Flash applications is limited: our FlashOver prototype uses suboptimal static analysis and randomized clicking to simulate a user. A determined attacker can easily uncover additional vulnerabilities using a manual static analysis. Likewise, the randomized clicker is lacking the cognitive ability of an actual human user. Therefore, the amount of vulnerable Flash applications detected in this experiment is a lower bound: the actual amount of vulnerable applications is most likely higher, making the security threat an even bigger issue. An interesting property of FlashOver is that it detects successful JavaScript injection by actually simulating a victim who triggers the use of the injected JavaScript code in one or more potentially exploitable variables. Thus, while FlashOver may miss some vulnerabilities (false negatives), it has practically zero false positives.

The main contributions of this paper are the following:

- Detailed analysis of an XSS attack vector that is commonly overlooked in Web application development
- Design and implementation of FlashOver, a fully automated system which uses a combination of static and dynamic analysis in order to identify Flash applications vulnerable to code injection attacks
- Evaluation of our system using Flash applications of the top Internet websites, showing the prevalence of the aforementioned vulnerability as well as our system's ability of detecting it

2. REFERENCES

[1] Flash Player | Adobe Flash Player 11 | Overview. `http://www.adobe.com/products/flashplayer.html`.

[2] M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In *Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '09, pages 88–106, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] Pc penetration | statistics | adobe flash platform runtimes. `http://www.adobe.com/products/flashplatformruntimes/statistics.html`.

[4] HTML5. `http://dev.w3.org/html5/spec/Overview.html`.

[5] JoMo-kun. m0j0.j0j0 Guide to IIS Hacking. `http://www.foofus.net/~jmk/iis.html`.

[6] Microsoft Silverlight. `http://www.microsoft.com/silverlight/`.

[7] Rich internet application (ria) market share. `http://www.statowl.com/custom_ria_market_penetration.php`.

[8] C. Stoll. The internet? bah! `http://www.thedailybeast.com/newsweek/1995/02/26/the-internet-bah.html`, 1995.

[9] The Cross-site Scripting FAQ. `http://www.cgisecurity.com/xss-faq.html`.