第10部

ウェブアプリケーションのセキュリティ技術の研究

門林 雄基, Gregory Blanc

第１章　Scope

The SWAN WG carries out research in the field of Web 2.0 application security. SWAN stands for Security for Web 2.0 ApplicatioN. It was founded and started its activities in June 2010. It aims to bring forward Web 2.0 application security issues in the WIDE project as these issues are becoming more and more critical to services offered on top of the Internet.

第２章　Second Year Activities

From its inception, the SWAN WG has decided to concentrate on a rather precise objective in order to appeal to the community and launch projects as fast as possible. We launched a first project, the Web2Sec Testbed, a testbed to accommodate largescale practical Web 2.0 application security-related experiments. The project does not only intend to build a practical testbed on top of the WIDE cloud but also to develop tools to be used within the testbed. While building a testbed seems relatively easy given the efforts deployed in other WIDE WGs such as WIDE-Cloud or ds1, theWebSec Testbed is yet to be constructed. However, the SWAN WG has been concentrating on developing analysis tools, especially to analyze JS malware which is a hot topic in Web 2.0 security.

For that purpose, we designed and started developing a proxy-based solution to analyze JS malware. This led to several publications on how to detect and extract obfuscation in JS malware[61] and how to automate deobfuscation to provide static analysis of JS malware[62, 63].

The proxy-based solution, named (sak_mis), would sustain the load of realtime static analysis on JS candidates. The system is designed to work as follows:

1. (sak_mis) is a proxy that intercepts HTTP requestsissued by the end user and subsequent responses returned by the server;

2. upon reception of an HTTP response, the proxy kicks off the prefetching stage:

3. the requested web page is parsed to detect script inclusions, links and potential malicious locations (iframes, images, etc.). Script contents are then retrieved (prefetching) and inlined into the original web page;

4. if newly downloaded contents also contain links or inclusions to contents of interest, prefetching is also performed on these contents. This scenario also applies when new inclusions are uncovered after deobfuscation;

5. once prefetching is completed, the aggregated script page is sent to an external application server;

6. the application server is responsible for automating deobfuscation: obfuscated contents and decoding

routines are extracted first. The deobfuscation stage of the attack scenario is emulated by the server. The process is repeated in case the deobfuscation yielded obfuscated contents as well;

7. once scripting contents can be directly interpreted by the machine, the decision module applies static analysis to extract a model of the script's intents. This model is compared to a knowledge base in order to infer whether it is a model of malicious intents or not;

8. in case, the script is benign, the deobfuscated script is injected back into the original web page and served to the end user.

Key components of the proxy-based solution have been designed and implemented, proving its usefulness against several real-world JS samples. More detailed treatment of this subject can be found in the full report of SWAN WG.