Flash vulnerabilities analysis of US educational websites

Joanne Kuzma*, Colin Price and
Richard Henson

Business School,
University of Worcester,
Worcester WR26AJ, UK
E-mail: j.kuzma@worc.ac.uk
E-mail: c.price@worc.ac.uk
E-mail: r.henson@worc.ac.uk
*Corresponding author

Abstract: With the increase in online and web learning, schools are building the number of web-based applications using media like Flash. However, sites that use Flash and other types of media encounter problems with security. Issues are raised with how to protect personal data that are entered via these sites. The purpose of this study is to determine if Flash-based web application at US educational institutions protect the personal data of their consumers, and what levels of security vulnerability are shown. The research also shows the main types of security problems that are shown in the schools sites. To mitigate these vulnerabilities and provide a higher level of security during development, technical, procedural and managerial recommendations are presented.

Keywords: security; Flash, web applications; education; universities; SwfScan.

Reference to this paper should be made as follows: Kuzma, J., Price, C. and Henson, R. (xxxx) 'Flash vulnerabilities analysis of US educational websites', *Int. J. Electronic Security and Digital Forensics*, Vol. x, No. x, pp.xx–xx.

Biographical notes: Joanne Kuzma is a Senior Lecturer in Computing at the University of Worcester and her main lecturing and research interests are in the areas of computer security, electronic commerce and web accessibility. She received her PhD in Information Systems from Nova Southeastern University (USA), and her MBA and BS from Pennsylvania State University and has her CISSP certification in computing security.

Colin Price is a Principal Lecturer in Computing at the University of Worcester and his main lecturing and research interests are in the areas of computer game and immersive environment development. He received his PhD in Electrotechnical Engineering from Katholieke Universiteit Leuven, Belgium, and a BA and an MA in Natural Sciences.

Richard Henson is a Senior Lecturer in Computing and a Knowledge Transfer Fellow in Information Security at the University of Worcester. His main lecturing and research interests are in the areas of information assurance and privacy, particularly in relation to SMEs and e-commerce. He received a BSc in Chemistry from Imperial College and an MSc in Computing Science from Staffordshire University.

1 Introduction

There are many aspects of security one needs to consider when developing web-based applications using media like Flash. News articles report daily security vulnerabilities and hacking attacks of online applications. Because of these, consumers are growing increasingly concerned about misuse of their personal information and are mistrustful of the security protection firms are employing. To protect consumers, federal and state agencies have enacted some legislation to attempt to secure systems and to protect consumer personal data, so it is mandatory that institutions understand legal issues related to security when developing their systems. In addition, institutional developers need to understand the technical framework when creating web-based applications, and review security of their systems before implementing on production servers.

However, despite both legal mandates and advice to fully test and develop secure systems; many educational institutions are not fully protecting their web users. This research analyses the level of security vulnerabilities among 250 US educational sites utilising Flash-based pages using a software tool, SwfScan, developed by Hewlett Packard (HP). It reviews the most common types of Flash-based vulnerabilities and recommends some methods to improve security.

2 Framework for web security

2.1 Legal aspects

There are a wide range of US federal and state laws which are aimed to protect personal consumer information collected and stored in computer systems. Most of these laws are dependent on one's industry, so it is imperative that application developers understand and adhere to specific legal mandates to protect the data of their consumers. The Health Insurance Portability and Accountability Act (HIPAA) covers securing personal data within the healthcare industry while the 1999 Gramm-Leach-Bliley (GLB) mandates protection for the financial industry (Johnson, 2005). Although the GLB is usually geared towards the financial industry, in 2002 the Federal Trade Commission (FTC) ruled that colleges and universities are covered under this law, and it requires institutions to have data-security systems in place ('Legislators try to shore up…', 2004).

Educational institutions including those in higher education must adhere to Family Education Rights and Privacy Act (FERPA). This federal privacy law prohibits institutions from disclosing personally identifiable information without a student's permission (Salomon et al., 2003). The authors explain that a significant technology risk is that the number of computerised records on databases have increased exponentially, thus increasing the likelihood of potential security vulnerabilities.

> "An institution may be particularly vulnerable to a FERPA violation if it can be shown that it was negligent in instituting procedures to protect against disclosure of electronic records. At the same time, however, continuous changes in the technology environment, combined with the often decentralized nature of institutions' networks, make it difficult to ensure that appropriate practices are in place. For example, the movement away from mainframe systems and toward distributed databases and servers has made it more difficult to have consistent and clear data access policies and procedures." (Salomon et al., 2003)

Thus, educational institutions must have clear policies related to data protection. However, even with clear policies, if there is a lack of training of the staff or application developers, the institution may still find itself at risk to data theft and security holes within their systems.

Some states have enacted their own laws to protect computerised students records. California was the first state to enact a security breach law – the 2003 California Database Security Breach 1386 (CA SB 1386), which requires prompt notification of any data breaches affecting California citizens, including students at educational institutions. This would also affect universities outside of California who have students who are California residents (Johnson, 2005). Thus, higher education officials throughout the USA would have to notify California students if data issues occurred, and it is to the benefit of these institutions to ensure that their computer applications are secure. Since California's implementation, 44 other states have/are considering acting laws that require businesses and other institutions that maintain personal consumer data to give consumers timely notice of a breach in the security of their personal information (Coakley, 2009). New York and Arizona limit the use and/or disclosure of Social Security numbers (SSNs) for educational institutions (Salomon et al., 2003). Thus, developers of computer applications need to be aware of a variety of state and federal laws when creating applications that use or store personal data.

2.2 University security breaches

There have been a myriad of computer security breaches at US universities, putting many staff and students at risk for personal data disclosure. According to Security Directors Report (2008), in USA there were 167 reported breaches in a variety of industries including: military, education, banking, medical and other businesses. Of these, educational institutions accounted for 25% of all breaches in the first three months of 2008, with 2.9% of compromised personal records in the total number of records stolen in these categories. According to Young (2008a), a report by Identity Theft Resource Center indicates that number of data-security incidents at colleges and universities across the country rose 47% between 2007 and 2008. These numbers include reported statistics, although there could be a sizable increase with the numbers of unreported or undiscovered breaches.

In February 2009, the University of Florida admitted a security breach exposed the names and SSNs of 97,200 students, faculty and staff who attended the school between 1996 and 2008. It also had other data breaches in October and November 2008. The latest breach was caused by a server configuration error (Vijayan, 2009). In 2006, Ohio University had five data breaches resulting in the theft of 173,000 SSNs and hackers using unsecured servers to launch attacks against outside systems. These breaches had such a negative impact on public relations that the school reported an 88% decline in donations once the breaches were reported (Culnan and Carlin, 2009).

2.3 Application security

Young (2008b) reports there are a myriad of computer security threats that universities face, but one of the top one includes malware, a large category which can include infected and unsecured code. According to Waters (2009), the rise of web-based applications is the number one avenue of malicious hacker attacks. The root of most problems is due to software issues, such as poor coding practices in the applications, and that it is difficult for developers to build secure systems. Waters also adds that educational institutions are finding more media-based applications, like Flash, containing security vulnerabilities. This is a growing problem for schools as more schools are putting online educational content online using media and web-based applications. Although application developers could easily become the scapegoat for web application systems, school administrators should realise that programmers may not have the knowledge or training or create secure applications, and project managers may value speedy development and functionality over secure systems (Waters, 2009).

In 2008, Culnan and Carlin (2009) conducted a privacy study of the top 236 schools from the US News and World Report 2004 list of best colleges. Although most of the study dealt with general privacy risk practices and notices, part of the study involved analysing privacy risk associated with web-based data collection forms (although not necessarily Flash-based forms). They found that of the 236 schools, all had at least one non-secure page with a data collection form, with an average of 424 such pages per school. Nearly all had at least one data collection form that used the HTTP GET method to submit the data with an average of 209 instances per school, a problem which could result in personal data to be made visible to unauthorised individuals. Their study suggested that nearly all institutions engaged in poor security practices that put personal data at risk.

A specific study on Flash-based application security was conducted by HP in 2008. They downloaded and audited 4,000 Flash applications and found numerous insecure security vulnerabilities including:

- of 250 Flash applications that had a login form 15% had user names or passwords hard-coded inside the application code
- 16% of SWF applications targeting Flash Player 8 and earlier contained XSS vulnerabilities
- 35% of all SWF applications violated Adobe's security best practices
- 77% of SWF applications targeting Flash Player 9 and 10 contained developer debugging information and source code file references (HP Application Security Center Community, 2009).

A serious problem in university security is that professors, colleges, departments and even student organisations regularly create and maintain separate shadow systems. So even if the university does have secure core applications and specific security policies, these shadow systems could create security vulnerabilities (Titus, 2008). According to Culnan and Carlin (2009), academic departments may bypass core university systems and operate their own individual servers. In addition, individual faculty, students and school organisations have their own systems, bypassing corporate servers and policies. For example, a staff member could create a separate Flash application to collect miscellaneous user information and this application could be developed with minimal thought to security, or could bypass corporate security policies and development procedures. Another reason for security problems in higher education institutions is that staff may be unaware of legal regulations that apply to the industry (Johnson, 2005). Or, staff may be unaware of specific university policies regarding safeguarding personal data.

They may also have minimal coding skills when developing applications and could be unaware of safe development practices regarding securing data.

2.4 Flash security framework

Adobe (previously Macromedia) aims with its Flash application is to provide robust security by using existing security solutions, architecture and protocols to avoid vulnerabilities exposures. Because of an integrated approach, institutions should not have to employ additional security solutions outside the Flash architecture and platform (Ludwig, 2005). Ludwig explains that in order to prevent malicious code from being spread, Flash utilises a sandbox approach where the application code is separate from the host system. In addition, to prevent cross-site scripting (XSS) threats, Flash content is delivered as a series of instructions in binary format to Flash Player over web protocols in the SWF file format in comparison to string-based language solutions. Protection against unauthorised access to private user information can be controlled through security settings and permissions (Ludwig, 2005). Thus, the Flash framework itself can supply a high level of security to protect against disclosure of personal information and other security vulnerabilities. However, even though Adobe has implemented secure features within the architecture, it is still up to developers to implement safe coding within the context of the framework. If they do not, security problems could still affect web applications. According to Fukami and Fuhrmanne (2008) one issue with Flash security is that most developers only superficially test Flash pages and in most cases, no testing of ActionScript functions happens. Many companies view Flash as a way to show pictures on sites, and do not realise that Flash can carry full applications. The authors give an analysis on two common methods of Flash attacks. First, HTTP GET requests could result in redirection attacks. Also, privacy concerns could results with so-called local shared objects (Flash cookies).

3 Methodology

The research was accomplished through completing an analysis of 250 US education sites containing Flash SWF applications to determine the most prevalent security problems for these sites. The project consisted of four phases:

1 choosing an online testing tool

2 choosing a list of education sites to test

3 run a software analysis

4 perform an in-depth analysis of the results.

3.1 Choosing a testing tool

The first phase of the study was to choose a tool to test for Flash application vulnerabilities. For this study, a software product from HP, called SwfScan, was chosen. This is a relatively new product which was released in March 2009 to allow web developers to test Adobe Flash applications for various types of security vulnerabilities. According to HP, it works in the following manner:

- decompiles applications built on the Adobe Flash platform to extract the ActionScript code and statically analyses it to identify security issues like information disclosure
- identifies and reports insecure programming and deployment practices and suggests solutions
- enables auditing of third party applications without requiring access to the source code
- audit the code for over 60 vulnerabilities including exposure of confidential data, XSS and cross-domain privilege escalation
- validate Flash application adherence to Adobe Best Practice (HP, 2009; HP Application Security Center Community, 2009).

Although a variety of Flash testing tools do exist, there were several reasons why this tool was chosen compared to other products, such as cost and testing of ActionScript versions 2.0 and 3.0. For example, a product by AsUnit, SWF UI, only tests ActionScript 2.0 (AsUnit, 2009). IBM's Rational AppScan software is a very comprehensive software product testing Flash and other web applications for security vulnerabilities (IBM, 2009a). However, this product cost ranged from $17,550 to $32,000, with a trial version available. However, the free trial was limited to 30 day and was limited in functionality testing (IBM, 2009b). Another software package reviewed was SWFIntruder developed by Open Web Application Security Project (OWASP). However, although this product is open source and free, one significant issue with using it is that it requires downloading the software on a server in order to test Flash applications and these applications can only be tested on the Firefox browser (OWASP, 2008). Finally, the 'erlswf' toolkit allows disassembly of SWF code and was designed to test security. However, it is only designed for ActionScript 2 tags and it is still missing some testing abilities, for example full AVM2/ActionScript 3 support and support for ActionScript 2 tags other than DoAction and DoInitAction (Fukami and Fuhrmanne, 2008) Therefore, based on an analysis of functionality as well as the free cost to use HP's product, SwfScan was chosen.

3.2 Choosing testing sites

The second phase of this project involved choosing a list of education sites in order to analyse for Flash vulnerabilities. To accomplish this, an advanced search of Google was performed with the following search criteria: 'login site:.edu filetype:swf'. The first parameter was a keyword search for the term 'login'. This was chosen in order to obtain as many Flash applications as possible that would have login capabilities or login screens, thus necessitating the need for greater security in security personal data in the login. It should be noted that although this keyword was used in the search, the results would not necessarily guarantee that a login screen would be a result of a search. If the text 'login' was used anywhere in the Flash application, whether it was an actual login screen or not, this would result in the application site being included within the search result. However, using this search term in Google would result in a higher probability of a login application. The second parameter was '.edu' which would only bring up sites with an US education suffix within the uniform resource locator (URL). The final Google search parameter was '.swf', which would only produce sites that contain Flash-based applications.

The top 250 results from this Google search contained a vast majority of US university educational institutions. There were a variety of different functional applications. One example was login screens which would require users to enter sensitive information such as user IDs and passwords. Other site results were purely informational web-based Flash applications not requiring users to enter information into the site.

3.3 Running the software analysis

The product software is downloaded from the HP site and can be installed on a stand-alone computer system with access to the internet. The software is started and the designer can choose whether or not to test 70 different error checks. Each specific error check is divided into a criticality ranking (critical, high, medium and low).

- *Critical*: severe impact problems that have serious repercussions on user security and lead to an extreme probability of risk and damage potential. These should be corrected as soon as possible.
- *High*: serious impact problems that can pose a significant level of security risk to users, and should also be addressed promptly.
- *Medium*: moderate problems that could pose some level of risk to users. These should be corrected, but with less urgency than the critical and highly ranked errors.
- *Low*: minor issues that pose slight or harmless risk to users, but should be reviewed by application designers. These may or may not need to be addressed based on the effort and cost to correct based upon the level of risk.

This phase of the study was completed in April 2009, and involved using SwfScan to determine the main types of Flash checkpoint errors and the quantity of each error. Although the software allows one to choose which of the 70 available Flash errors to test, for this study all 70 error checks were chosen to test for each of the 250 web applications. The URL of each of the 250 websites to be tested was typed into a selection box and each individual test was compiled. For each URL run, an individual error report was produced listing which specific Flash-based errors were found for that URL. In addition, each error was listed by the criticality of the specific error. The next phase of this project was to take the raw data from the SwfScan results and compile it into tabular format and analyse the results.

4 Analysis of results

Results in Table 1 show the number and types of Flash errors reported within the 250 university sites tested. The first column of the table shows the specific type of error, while the second column is the level of severity for that error (critical, high, medium and low). Column three contains the number of errors that were reported across 250 sites, while the fourth column is the calculated percentage of the type of error for the total number of sites. The final column lists the potential application or coding fix suggested by HP in order to ensure this type of error does not occur.

Table 1 Flash error statistics and fixes

Type of error	*Criticality*	*# errors*	*Percent*	*Fix*
Insecure Security.AllowDomain()	Critical	6	2.4	Avoid using a wildcard as parameter in a Security.allowDomain() method call. Instead specify a list of trusted domains in the allowDomain call
FlashVars Cross Scripting	High	5	2.0	Properly validating user supplied input via white listing
FlashVars Cross Scripting/Request Forgery	High	5	2.0	Properly validating user supplied input via white listing
Insecure Flash Storage Object	High	1	0.4	Configure the localPath of the shared object to allow only trusted SWFs at specific URL paths to gain access to the shared object
Possible Cross-site scripting in getURL using 'GET'	High	5	2.0	Perform proper validation on the data before making a request to the URL via a getURL call
Possible FlashVars Cross-Site Scripting in HTMLtext property bound to an Uninitiialised Variable	High	6	2.4	Avoid using undefined or uninitialised variables and validate values obtained from user through FlashVars
Debug Messaging	Medium	31	12.4	Set 'Omit Trace Actions' to 'true'
MD5 Hash Detected	Medium	2	0.8	Only use cryptographically secure hashing algorithms
Potential User Account Information Disclosure	Medium	51	20.4	Ensure information of potential value to an attacker is not being left in application code
Possible Application Information Disclosure	Low	13	5.2	Ensure information of potential value to an attacker is not being left in application code
Possible Commerce Information	Low	1	0.4	Ensure information of potential value to an attacker is not being left in application code
Possible Cryptographic Data	Low	1	0.4	Ensure information of potential value to an attacker is not being left in application code
Potential Interesting Name Encountered	Low	4	1.6	Ensure information of potential value to an attacker is not being left in application code
Potential Personal Information Disclosure	Low	27	10.8	Ensure information of potential value to an attacker is not being left in application code
Shared Flash Storage Object	Low	8	3.2	Do not store sensitive information in the local stored objects
Suggested Security Controls for Embedding SWF Files in HTML	Low	248	99.2	When embedding SWF in HTML page, set the Allow Networking Access flag to internal

Of the 250 sites tested, only 2 (0.8%) did not have any reported Flash issues. The vast majority of sites had the error 'Suggested Security Controls for Embedding SWF Files in HTML' reported. According to HP documentation this indicates that the SWFScan examined the Flash ActionScript and revealed that it did not utilise any browser or network communication functionality (HP, 2009).

Table 1 results show that there were six other low-criticality errors. The second most common of this category was 'Potential Personal Information Disclosure', which was found in 27 sites (10.8%). This error indicates that there is a potential vulnerability due to the presence of a 'keyword' that would allow personal information to be disclosed. To fix the problem, application designers or coders should ensure that this potential personal information is not left within the application code. Five other low-criticality errors existed where similar information could be viewed within the application code. These included: 'Possible Application Information Disclosure', found in 13 sites (5.2%), 'Possible Commerce Information' (1 site and 0.4%), 'Possible Cryptographic Data' (1 site and 0.4%), 'Potential Interesting Name Encountered' (4 sites and 1.6%) and 'Potential Personal Information Disclosure' (27 sites and 10.8%). The final low-severity problem was 'Shared Flash Storage Object', which means that the data stored in shared objects are not encrypted and could be prone to unauthorised access depending on the access restrictions specified by the developer (HP, 2009). This problem was found in eight sites (3.2%).

There were three errors of medium level severity found in these results. A sizable number of sites were found with the error 'Potential User Account Information Disclosure' (51 sites, 20.4%). This error is similar in function to the five low-severity errors where information is displayed via application code, although this specific error is of a more serious nature. Another medium-ranked problem was 'Debug Messaging' with 31 sites (12.4%) containing this issue. According to HP (2009) this specific fault indicates that the trace() function is being utilised and was detected due to the presence of Debug Messaging. This could indicate a serious security concern as path names and other information could be revealed to hackers. Finally, the third error in this category was 'MD5 Hash Detected', which indicates that a string of hexadecimal digits matching the length of a cryptographic hash from the MD family was detected. These hashes are used to protect passwords and other sensitive personal data. Certain hashes such as MD5, MD4 and MD2 are especially susceptible to attacks, and should not be considered secured. Therefore, they should only be used in certain situations.

There were five error types within the 'high' ranking category, with the number of sites ranging from one error up to the maximum of six sites displaying a specific error. The most common error in this category was 'Possible FlashVars Cross-Site Scripting in HTMLtext property bound to an Uninitialised Variable' (six sites, 2.4%). These types of XSS vulnerabilities can be manipulated by hackers to steal cookies, create requests that could be mistaken from those of a valid user, compromise confidential information or execute malicious code on client systems (HP, 2009). Three other specific scripting errors were found in five sites each (2.0%). These were:

1 'FlashVars Cross Scripting'

2 'FlashVars Cross Scripting/Request Forgery'

3 'Possible Cross-site scripting in getURL using 'GET''.

In addition to compromising confidential information and executing malicious code, if JavaScript is used within the application, appended malicious text can be used to execute arbitrary JavaScript code leading to a XSS attack (HP, 2009). The fifth type of high-ranking error encountered was 'Insecure Flash Storage Object', displayed by one site (0.4%). This error indicates that if designers incorrectly set a localPath value, all SWF files on that domain, including multiple parties, could therefore have unauthorised data access.

There was one critical error in the study results, which was 'Insecure Security AllowDomain()', found in six sites (2.4%). To provide secure SWF applications, it is important to restrict loading of remote SWFs to trusted domains only by using the System.security.allowDomain() method. However, if an incorrect parameter is passed to a Security.allowDomain method call, the developer can have an overly permissive cross-domain setting which could lead to threats such as cross-domain privilege escalation, data injection and unauthorised data access (HP, 2009).

5 Implications

Evaluation results show that the majority of university Flash applications have some type of security issues. The vast majority of sites had the low-criticality problem 'Suggested Security Controls for Embedding SWF Files in HTML' reported. Recommendations to fix the problem include setting the AllowNetworkingAccess flag to none. This will implicitly disable the SWF applications communication ability to the browser or network (HP, 2009). However, it should be noted that this is a low priority issue. Based on the cost to fix and other business factors, a fix may or may not be appropriate. Of the six other low-severity errors, five were very similar in the types of information that would be disclosed including:

1 Possible Application Information Disclosure.

2 Possible Commerce Information.

3 Possible Cryptographic Data.

4 Potential Interesting Name Encountered.

5 Potential Personal Information Disclosure.

For each of these errors, the similarity in the functionality resulted in HP recommending a similar fix, ensuring information of potential value to an attacker is not being left in application code and to remove that information from the production server (HP, 2009). The final low-priority error was 'Shared Flash Storage Object', and the recommended fix was not to store sensitive information in the local stored objects (HP, 2009).

There were three medium-priority errors, with the most common being 'Potential User Account Information Disclosure'. The fix for this error was the same as the fix for the majority of low-ranked errors where personal information was revealed, ensuing information is not being left in application code. The next most common medium-ranked error dealt with displaying debug messaging, and HP's recommended fix was to remove all debugging messaging from the application code before it is moved to the production servers. Designers should also set the 'Omit Trace Actions' to 'true', which will make the published SWF smaller and will remove any excess information or actions from the

SWF (HP, 2009). The MD5 Hash error occurs because this hash function is especially susceptible to attacks, and this MD5 hash should not be used. Instead, the application should use more secure cryptographical hash algorithms, such as SHA-224, SHA-256, SHA-384 or SHA-512 (HP, 2009).

For the four high-priority scripting errors, HP (2009) recommends the following prevention techniques to minimise XSS attacks in Flash applications:

- set appropriate allowScriptAccess and allowNetworking parameters within the HTML code
- perform data validation on variables sent to URL functions to ensure only http:// and https:// protocols are allowed; validate that the URL is for an allowed domain or use relative URLs
- escape special characters placed within HTML text fields
- do not use HTML text fields unless HTML support is needed
- compile the SWF for more recent Flash Player versions
- encourage users to have the latest version of Flash Player to view your content.

For the high-priority error 'Insecure Flash Storage Object', an incorrect setting of the localPath function could lead to vulnerabilities. It is recommended that the localPath be restricted to specific URL paths of trusted SWFs to gain access to the shared object. The critical error 'Insecure Security.AllowDomain()' could be avoided by not using a wildcard as parameter in the method call, and instead specifying a list of trusted domains in the call (HP, 2009).

Fukami and Fuhrmanne (2008) give some specific technical advice for thwarting Flash-based attacks. One approach is to use run-time or dynamic code analysis to analyse reoccurring behaviours such as file downloads while residing in a safe testing environment. For security reviews outside the testing environment, they suggest using static code analysis, where the SWF file is disassembled in order to check for analysation patterns recognising various attack methods.

Although technical recommendations aimed at the programmers for securing their Flash applications are most relevant for the purpose of this study, it should be realised that technical security measures addressing developers should be only one part of a multi-phase security agenda in educational institutions. First, educational entities should analyse applicable federal, state and local ordinances with regards to security and personal data safeguards. Many laws have been already passed, and others are pending, so it would be wise for schools to coordinate with their legal advisors to properly assess and analyse applicable and upcoming legislation (Salomon et al., 2003). In addition, mandatory workshops or training for staff and developers should be held in order to update them on their legal obligations to provide secure systems and safeguarding consumer data. Staff should understand how to properly handle data, and application developers should be taught how improper security development could have an adverse affect on the school from a legal standpoint.

A second practical solution that schools should implement is to periodically assess information security vulnerabilities and risks (Salomon et al., 2003). With tools such as SwfScan, schools could periodically run tests on their Flash (and other) applications to assess any security holes and risks. However, it should be noted that once a risk assessment is completed, the school needs to take appropriate action to correct the problems. According to Salomon et al. (2003), if an institution identifies vulnerabilities and then fails to institute appropriate measures to remedy those issues, the institution could open itself up to potential liability if a breach should occur, especially if this information is later obtained by the public under a Freedom of information or 'open records' law request.

A third tactic the institution should implement is to review and update secure policies and procedures. A school should develop a policy in line with the nature and magnitude of usual threats and respond with an appropriate level to ensure their information is being protected (Salomon et al., 2003). Along with developing these policies, it is important to educate staff as to the nature of the policies and to have some consequence in place if policies are not followed. In dealing with web application development, policies could exist where developers would have to test their applications for such Flash security problems as 'Potential Personal Information Disclosure' or 'MD5 Hash Detected.' The policy could state that approvals would have to be obtained before applications could be moved to a production server.

Finally, secure application development needs to be a priority in project management. Waters (2009) indicates that ultimately, it is the developers who are responsible for security. However, instead of emphasising rapid development and functionality, project managers and institutional administrators also need to put a high priority on security.

6 Conclusion

This research shows that a preponderance of US educational sites using web-based Flash applications do not adequately secure these pages. Almost all pages showed at least low-level security vulnerabilities, while over 20% of them had medium-level security issues where personal information could be disclosed to attackers. This indicates a serious concern for education sites, who are increasing the number of their Flash-based pages and applications, especially due to the growth of online learning. It may be impossible to make a web application 100% secure, but institutions need to implement a variety of policies in order to better secure their sites. There are a variety of technical, legal and procedural methods that institutions could effectively implement to provide a better level of consumer protection.

References

Adobe Systems (2008) *Adobe Flash Player 9 Security White Paper*, Available at: http://www.adobe.com/devnet/flashplayer/articles/flash_player_9_security.pdf.

AsUnit (2009) *AsUnit Home Page*, Available at: http://asunit.org/.

Coakley, M. (2009) 'Privacy protection, safety and security: a state law enforcement perspective', *The Computer and Internet Lawyer*, Vol. 26, No. 4. pp.1–9.

Culnan, M. and Carlin, T. (2009) 'Online privacy practices in higher education: making the grade?', *Communications of the ACM*, Vol. 52, No. 2., pp.126–130.

Fukami, S. and Fuhrmanne, B. (2008) 'SWF and the malware tragedy, detecting malicious adobe flash files', *Application Security Conference*, May 2008, Ghent, Belgium, Available at: http://www.owasp.org/images/1/10/OWASP-AppSecEU08-Fukami.pdf.

AU: Adobe Systems (2008) and Jackson et al. (2009) are given in the list but not cited in the text. Please check.

HP (2009) *Register to Download SwfScan Software*. SwfScan Ver. 1.0, Program Documentation, Available at: https://h30406.www3.hp.com/campaigns/2009/wwcampaign/1-5TUVE/index.php?key=swf&jumpid=go/ swfscan.

HP Application Security Center Community (2009) *Exposing Flash Application Vulnerabilities with SWFScan*, Available at: http://www.communities.hp.com/securitysoftware/blogs/spilabs/archive/2009/03/20/exposing-flash-application-vulnerabilities-with-swfscan.aspx.

IBM (2009a) *Rational AppScan Standard Edition*, Available at: http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=en_US&synkey=T864089C20170U90.

IBM (2009b) *View Pricing and Buy*, Available at: https://www-112.ibm.com/software/howtobuy/buyingtools/paexpress/Express?P0=E1&part_number=D040CLL,D040DLL,D61SYLL,D61T0LL,&catalogLocale=en_US&Locale=en_US&country=USA&S_TACT=none&S_CMP=none&PT=html.

Jackson, C., Barth, A., Bortz, A., Shao, W. and Boneh, D. (2009) 'Protecting browsers from DNS rebinding attacks', *ACM Transactions on the Web*, Vol. 3, No. 1, pp.2–26.

Johnson, J. (2005) 'The privacy challenge', *Network World*, 30 May 2005, p.29.

'Legislators try to shore up campus data security holes' (2004) *Recruitment and Retention in Higher Education*, September, pp.5–6.

Ludwig, A. (2005) 'Macromedia flash platform security and macromedia enterprise solutions', *Macromedia White Paper*. Available at: http://security.hoffmanmarcom.com/docs/Network_Security_Authentication_Access_Control.pdf.

Moshchuk, A., Gribble, S. and Levy, H. (2008) 'Flashproxy: transparently enabling rich web content via remote execution', *ACM International Conference on Mobile Systems, Applications and Services Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services*, Breckenridge, CO, USA, pp.81–93.

OWASP (2008) *SWFIntruder Overview*, Available at: https://www.owasp.org/index.php/Category:SWFIntruder.

Salomon, K., Cassat, P. and Thibeau, B. (2003) 'IT security for higher education: a legal perspective', *EDUCAUSE,* 20 March 2003, Available at: http://net.educause.edu/ir/library/pdf/CSD2746.pdf.

Security Directors Report (2008) 'General business and universities report most data breaches in first quarter', *Ioma.com*, June, p.2208.

Titus, A. (2008) '5 key ways your electronic data may be at risk', *Chronicle of Higher Education*, 24 October 2008, Vol. 55, No. 9.

Vijayan, J. (2009) 'University admits to third recent breach', *Computerworld*, 2 March 2009, p.7.

Waters, J. (2009) 'IT security: target: the web', *T.H.E. Journal*, February, Vol. 36, No. 2.

Young, J. (2008a) 'Number of data breaches at colleges is still rising', *Chronicle of Higher Education*, 23 January, Vol. 55, No. 20.

Young, J. (2008b) 'Top 10 threats to computer systems include professors and students', *Chronicle of Higher Education*, 19 December, Vol. 55, No. 17.