Vulnerabilities of Government Websites in a Developing Country – the Case of Burkina Faso

Tegawendé F. Bissyandé[1,2(✉)], Jonathan Ouoba[3], Daouda Ahmat[4], Fréderic Ouédraogo[5], Cedric Béré[2], Moustapha Bikienga[5], Abdoulaye Sere[6], Mesmin Dandjinou[6], and Oumarou Sié[2]

[1] SnT, University of Luxembourg, Luxembourg, Luxembourg
tegawende.bissyande@uni.lu
[2] Université de Ouagadougou, Ouagadougou, Burkina Faso
{cedric.bere,oumarou.sie}@univ-ouaga.bf
[3] VTT Technical Research Center, Espoo, Finland
jonathan.ouoba@vtt.fi
[4] Université Virtuelle du Tchad, N'Djamena, Chad
daoudique@gmail.com
[5] Université de Koudougou, Koudougou, Burkina Faso
{frederic.ouedraogo,moustapha.bikienga}@univ-ouaga.bf
[6] Université Polytechnique de Bobo Dioulasso, Bobo-Dioulasso, Burkina Faso
{abdoulaye.sere,mesmin.dandjinou}@univ-ouaga.bf

Abstract. Slowly, but consistently, the digital gap between developing and developed countries is being closed. Everyday, there are initiatives towards relying on ICT to simplify the interaction between citizens and their governments in developing countries. E-government is thus becoming a reality: in Burkina Faso, all government bodies are taking part in this movement with web portals dedicated to serving the public. Unfortunately, in this rush to promote government actions within this trend of digitization, little regards is given to the security of such web sites. In many cases, government highly critical web sites are simply produced in a product line fashion using Content Management Systems which the webmasters do not quite master.

We discuss in this study our findings on empirically assessing the security of government websites in Burkina Faso. By systematically scanning these websites for simple and well-known vulnerabilities, we were able to discover issues that deserved urgent attention. As an example, we were able to crawl from temporary backup files in a government web site all information (hostname, login and password in clear) to read and write directly in the database and for impersonating the administrator of the website. We also found that around 50 % of the government websites are built on top of platforms suffering from 14 publicly known vulnerabilities, and thus can be readily attacked by *any* hacker.

Keywords: e-government · Websites · Security · Vulnerabilities · CMS · Developing countries

R. Glitho et al. (Eds.): AFRICOMM 2015, LNICST 171, pp. 123–135, 2016.
DOI: 10.1007/978-3-319-43696-8_14

1 Introduction

E-government is now a pillar of ICT4D initiatives to improve the life of citizens in developing countries. Generally, it is realized through a web portal (website) where citizens can readily collect information and interact with government officials in an effort for simplifying administrative processes. In their simplest form, government websites are designed to be a reliable data source for all citizens. These websites are thus sensitive sources of information and, as such, they should be resilient to most tampering attempts. Unfortunately, recent high-profile security mishaps across the world, and in particular in developing countries, show that the design and implementation of e-government portals leave security holes that are exploited by attackers.

The primary cause of the precarious situation in which most e-government portals are today, especially in the context of developing countries, is the lack of attention that developers give to the assessment of their installs. A secondary reason is the fact that website design is often outsourced and thus the resulting web site is built in a way that government agents, which are not IT professionals, can easily update text in the web. To that end, Content Management Systems (CMS) are heavily used.

From WordPress to Joomla! and beyond, businesses (such as newspapers or e-commerce companies) and institutions (such as schools or city halls) depend on CMS to maintain online content. Thus, these third-party platforms are everywhere, and like all software, they come with security concerns. Not surprisingly, the popularity of CMS has been an opportunity for malicious hackers, since CMS provide a much larger attack surface. Before the proliferation of CMS, hackers had to focus on finding a vulnerability in a single identified target (e.g., a bank) and then attacking it to compromise its services (e.g., Denial of Service attacks) or to steal data. Today, however, with the vast opportunities presented by CMS, hackers take the path of least resistance (i.e., no time and computing power is wasted on trying to find a vulnerability in a single strong target). Indeed, "use search engines to identify common security vulnerabilities in a CMS platform as a means to accomplish server takeover and data theft" [1]. Unfortunately, as we will show later in this paper, there are literally thousands of security vulnerabilities in CMS platforms. Once such weaknesses are identified, it is again easy to rely on a search engine to fingerprint websites that are built on top of CMS that are affected by the known vulnerability. Doing so, malicious hackers can exploit the vulnerability in multiple CMS in many businesses and institutions, and they can do so very fast.

Nevertheless, businesses and institutions can still defend themselves with some simple tactics. Unfortunately, they are often not aware of the vulnerabilities of such platforms. Our work is part of this effort to sensitize government officials and web developers in developing countries of the perils of overtrusting CMS. The contributions of this paper are as follows:

– We describe and survey the security vulnerabilities that are found in today's popular CMS platforms.

- We investigate government websites in Burkina Faso to study those that are built on top of CMS platforms.
- We develop `s2e-gov`, a framework for security testing of e-government portals. It exploits a database of known web vulnerabilities and other heuristics to assess the security of web sites.
- We provide guidelines for readily securing e-government websites against common security attacks. In particular we call for the teaching/training of web security basics to the younger generations as we previously did for bootstrapping software engineering teaching in developing countries [2].

The remainder of this paper is organized as follows. Section 2 motivates our work in the context of a developing country, namely Burkina Faso. Section 3 describes web vulnerabilities and presents an overview of their presence within popular CMS. We discuss the implementation of `s2e-gov`, our e-government security scanning tool, in Sect. 4. Section 5 then details the lessons that we have learned while experimenting with `s2e-gov`. We discuss related work in Sect. 6 before concluding in Sect. 7.

2 Motivation

During April 2015, a large number of government websites in Burkina Faso were hacked to deliver a single page with a message from a radical group. Major newspapers in the country made headline stories out of these incidents. Observers and readers of online content were led to believe that these attacks were retaliation due to the engagement of the country in the war against terrorism. Online forums have seen discussions on how websites in Burkina Faso were poorly developed with little consideration for security. At that time, our opinion was simply forged by the fact that many other websites in the country, and beyond, had fallen. Given the relatively limited strategic importance of Burkina Faso for hackers, as a researcher we immediately set to understand how such attacks could have been easily, and potentially blindly, performed without specific targets in mind. We then collected the dynamically generated web pages of a number of web sites to study the HTML code. A first review showed that these web pages contained default information on how they were built. In particular, they contained meta-information in HTML headers describing the CMS platform on top of which the web sites are built.

CMS Use in Government Websites in Burkina Faso. To evaluate the penetration of CMS usage in government websites we identify the top government bodies (ministries and departments) as well as institutions (research centers, universities, etc.) and scan their webpages. Our collected dataset includes 42 websites.

Similarly to CMS detectors existing on the web, we compiled a database of fingerprints of CMS versions using hash values of some reference files (e.g., configuration, License and Readme files). While scanning websites from our list, we search for such files and directories in default paths. When they exist, we

deduce that the website has been built via a specific CMS. Using the hash values of the reference files we can further identify the CMS version. Table 1 depicts the distribution of CMS usage in government websites. For each CMS version used we refer to the National Vulnerability Database (NVD) hosted by the US Government National Institute of Standards and Technologies (NIST) to determine the number of known vulnerability exposures (CVE) that have been reported for this CMS version.

This first investigation clearly shows that Government websites in Burkina Faso are largely built on top of vulnerable CMS versions. A large proportion of websites have been built with Joomla! 1.5 which was released in January 2008 and which is no longer supported since September 2012: this means that even if vulnerabilities are found today in this version, no Joomla! developer will officially work on releasing a patch for it. Despite the 14 vulnerabilities known for this Joomla! version, relevant government websites have not yet been updated as of June 2015. Yet, the content of the websites are still up to date, implying that they are still important tools of communication for the government.

Table 1. Distribution of CMS usage for government websites in Burkina Faso

CMS	Version	Nb sites	Nb vulnerabilities	CMS release date	End of life
Joomla!	1.5	19	14	22 Jan 2008	Sept 2012
Wordpress	3.5.1	1	13	24 Jan 2013	-
Drupal	-	1	(135)	-	-
Microsoft sharepoint	14.0.0	1	(35)	-	-
SPIP	3.0.17	1	0	August 2014	-
OpenCMS	-	1	(14)	-	-
Static / in-house PHP	-	12	-	-	-
Unreachable hosts	-	6	-	-	-

3 Vulnerabilities in CMS

Software *vulnerabilities* are program defects that provide the opportunity to malicious users to attack a system or application. In the literature, they are often referred to as security bugs [3] or software weaknesses. In this article, we use the terminology of the CVE[1] system to define vulnerabilities: "*An information security "vulnerability" is a mistake in software that can be directly used by a hacker to gain access to a system or network*"[2].

[1] Common Vulnerability Exposures.

[2] https://cve.mitre.org/about/terminology.html.

Common security breaches due to software vulnerabilities include sensitive information leakage, modification, and destruction. *Attacks* are successful exploitations of vulnerabilities. We refer to the CVE for our vulnerability counts: it is a system that provides a reference for all publicly known security vulnerabilities. This system is funded by the US government and is managed by the National Institute of Standard and Technology (NIST). To easily share data related to vulnerabilities, each identified and accepted vulnerability receives a unique identifier. Based on this identifier one can retrieve information about the vulnerability, including its description, the product concerned, the version of the product, the date of vulnerability record creation and some comments. In this paper, we use the CVE system as a dictionary for the CMS vulnerabilities that we study. In the same lines, we use the term exposure or exploitable to describe a software vulnerability that was advertised to the public and for which it exists ways to take advantage of it.

There are different types of vulnerabilities that can be found in web applications, including CMS platforms. We describe those vulnerabilities and provide statistics on their appearance in popular CMS platforms.

File Inclusion is a type of vulnerability which allows an attacker to include a file, usually through a script, on the web server. This often occurs due to the use of user-supplied input without proper validation.

Cross Site Request Forgery is a type of vulnerability that makes a malicious Web site, email, blog, instant message, or program cause a users Web browser to perform an unwanted action on a trusted site for which the user is currently authenticated. Attacks based on this vulnerability can result in an unauthorized transfer of funds, changing a password, or purchasing an item in the user's context.

Gain Privileges also known as *Privilege Escalation*, is a vulnerability that allows attackers to gain elevated access to resources that are normally protected from an application or user. This results in them being able to perform unauthorized actions via a vulnerable application.

Bypassing is a vulnerability type where authentication schemes can be bypassed by simply skipping the login page and directly calling an internal page that is supposed to be accessed only after authentication has been performed. Similarly, one can bypass authentication measures by tampering with requests and tricking applications into thinking that we're already authenticated. Bypassing can be accomplished either by modifying a given URL parameter or by manipulating the form or by counterfeiting sessions.

Directory Traversal (also known as *path traversal*, *directory climbing* and *backtracking*) is a vulnerability where insufficient security validation/sanitisation of user-supplied input file names can be exploited so that characters representing "traverse to parent directory" are passed through to the file APIs.

Code Execution refers to a vulnerability through which an attacker is able to execute any commands of his choice on a target machine or in a target process.

Cross Site Scripting (XSS) attacks where malicious scripts are injected into otherwise benign and trusted web sites. They occur when an attacker uses a web

application to send malicious code, generally in the form of a browser side script, to a different end user. XSS vulnerabilities occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

SQL Injection is a code injection technique in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection attacks exploit vulnerabilities where user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed.

Denial of Service (DoS) attacks are focused on making a resource (e.g., web site or server) unavailable for the purpose it was designed. Denial-of-service attacks significantly degrade the service quality experienced by legitimate users by introducing large response delays, excessive losses, and service interruptions.

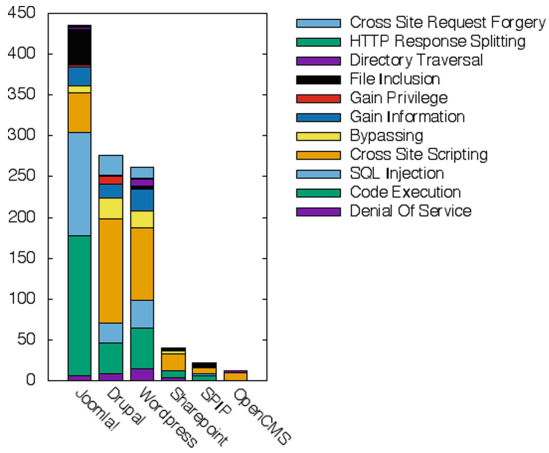

Fig. 1. Distribution of vulnerbaility types in popular CMS – data from NIST as of June 13, 2015

Vulnerability statistics depicted in Fig. 1 show that many CMS suffer from numerous vulnerabilities. Code Execution, SQL Injection and XSS vulnerabilities are the most widespread. Joomla! is the CMS with the most known vulnerabilities.

4 S2e-Gov: Security Scans for E-Government

In this section we propose a framework for security testing of websites. In `s2e-gov` we automate the process of identifying security concerns, whether known vulnerabilities or misconfigurations, that targeted e-government websites may contain. Figure 2 illustrates the inputs of outputs of `s2e-gov`. We detail the key aspects of its implementation in this section.

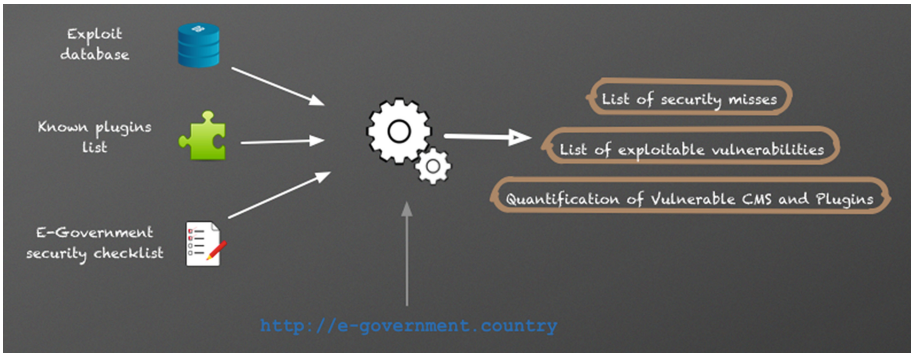

Fig. 2. Schematization of `s2e-gov` process for security testing of e-government websites

4.1 Default Configuration and Directory Listing

The first feature implemented in `s2e-gov` is about directory listing: after verification of the HTML code extracted from the index page, we proceed to check default configuration files and directories in popular CMS. Because a CMS often includes configuration files where credentials are stored it is important that such files are not readily accessible for reading. CMS developers hide such information in PHP files which cannot be read by client browsers. However, when editing those files with editors such as Vim, Nano, Notepad or Emacs, temporary backup files are often automatically created. If these are not deleted, all configuration information are left and become accessible in plain text format.

`s2e-gov` thus systematically tries to detect unprotected configuration files which are available on the website. These files usually contain clear text credentials to the login page of the website, or credentials for accessing the backend databases of CMS platforms.

4.2 Common Security Guards

The second feature implemented in `s2e-gov` is the verification of common security protocols in websites. For example, transaction data between user web browsers and government websites must be moved securely across the internet without any body being able to eavesdrop or tamper with the data. This security is commonly implemented through encryption offered by the Secure Socket Layer (SSL) to support HTTPS connections. Website administrators must therefore make available a certificate (with their public key) to allow users browsers to encrypt their requests.

Similarly, the X-Frame-Options HTTP header is commonly known as a good means to protect websites from Clickjacking attacks which involve fooling users into clicking on seemingly harmless buttons or links[3]. The vulnerability comes

[3] http://jeremiahgrossman.blogspot.de/2008/10/clickjacking-web-pages-can-see-and-hear.html.

from the possibility to place the targeted web page (e.g., the Adobe Flash settings panel, a Facebook like button or a shopping cart) inside an Iframe and overlay it with a completely different webpage. Such an overlay might then entice users to click on certain buttons or links, leading to Clickjacking, double-click jacking, like-jacking, cursor-jacking attacks. It is possible however to prevent such attacks by simply parameterizing the X-Frame-Options:

* X-Frame-Options: `DENY`: "wont allow the website to be framed by anyone"
* X-Frame-Options: `SAMEORIGIN`: "No one can frame except for sites from same origin"
* X-Frame-Options: `ALLOW-FROM %uri%`: "Only this one URI can frame. No one else"

`s2e-gov` checks for such common security guards in the websites and ensures that they properly set.

4.3 Known Attacks

Although the NVD database shows that the core of all CMS contained at some point some vulnerabilties, security bulletins recurrently point out more security holes in plugins. `s2e-gov` thus implements a plugin detection for popular CMS to collect the list of plugins added to the CMS core. Once this list is known, `s2e-gov` checks whether they contain known vulnerabilities.

Plugin Detection in CMS – Vulnerabilities in CMS are mostly found in extensions code such as plugins, rather than the CMS core code. We first build a list of plugins available on the Internet for the different plugins supported by our tool `s2e-gov`. Then, for each CMS install, `s2e-gov` identifies the included plugins by scanning the web directory: for each plugin, `s2e-gov` attempts to find out whether there exist some known vulnerabilities to report. To this end, we rely on the Exploit database.

Database of Exploits – the Exploit Database[4] is an established archive of attacks that exploited CVE compliant vulnerabilities in various software. We rely on this database to search for vulnerabilities in plugins that have been successfully exploited, either in testing scenario or in harmful hacks.

WordPress TimThumb Exploitation – Website themes in CMS come as add-ons that extend the core layout changes. Often, themes contain variables that refer to dynamic elements such as images. In the popular Wordpress CMS, it has been reported that themes often come with insecure PHP files which are used for caching and or resizing images. The recent "TimThumb" exploitation refers to a project which essentially caches even remote files locally, without doing the necessary proper sanitization: timthumb.php only checked whether the target

[4] https://www.exploit-db.com.

file is actually an image or not. This PHP file is used in many themes, with different names.

The Timthumb exploitation is realized simply by tricking TimThumb into believing that a remotely stored file, which may contain malicious PHP for example, is an actual image. Attacks found this trick very simple to perform and many attacks based on the TimThumb exploit were recorded in forums[5].

Joomla Content Editor Bots – JCE is popular component that can be found enabled in most Joomla! sites as a fancy content editor. Unfortunately in its development course, the JCE had once a well known security hole that has since been fixed in Joomla! versions higher than 1.5.x series. This security hole allows anyone to upload arbitrary files to a server. One can easily find a working exploit on the Internet for such a vulnerability. A typical exploitation scenario consists in 3 steps where the attacker scans Joomla! installs looking for vulnerable JCE, then exploits the bug in the JCE image manager to upload a PHP file with an image extension (e.g., .gif) to the images/stories directory. Finally, the hacker can simply rely on a JSON comand to change the file extension to .php. At this point, the attacker has a backdoor to the website and do whatever it wants with the site. Usually, attackers use this strategy to build their botnets.

4.4 E-Government Requirements

More than any other web site, e-Government portals are required to offer reliable services that make the information available to those who need it in a trustable setting. To realize this e-Government portals must setup security policies and enforce security practices during the design, development and production of their services.

Policy – While it is importance to take steps to be secure, it is even better to be sure that one are secure. Security policies are documents that outline the long-term strategy in an organisation to ensure that, when it is respected, information and infrastructure is secured. For example, policy indicates the acceptable use for users and the guidelines for reacting to a web site compromise.

Practice – Security practices, which play a key role for ensure e-Government security, are checklist for actions and advice on how to keep systems secure. Most prominent examples for security practices in e-Government include:

- Authenticate all accounts and make sure that the passwords are difficult to guess. Preferably use One-Time-Password to ensure that no masquerade attack can be performed even if an attacker uses a sniffer to collect passwords.
- Check for resource and sofware integrity regularly by maintaining a list of their MD5 checksums
- Check logs regularly to audit data and early detect anomalies. Possibly, install firewalls to drop unsollicited packets.

[5] https://www.exploit-db.com/wordpress-timthumb-exploitation/.

– Always update to the latest software available from the vendor with recent upgrades and patches

Finally, our experience in assessing government websites has shown that their design and implementation require: (1) some degree of obfuscation (e.g., which CMS is used) to complicate the work of attackers; (2) some level of resilience to DoS attacks to ensure availability; (3) a tight control on resource access (e.g., no dynamic loading) to preserve integrity; and (4) a certification/authentication scheme via a Public Key Infrastructure to guarantee confidentiality and integrity in information exchange.

5 Security Issues in Burkina Faso Government Websites

Our experiments with `s2e-gov` on the government websites have highlighted a substantial number of security holes that make it easy for any hacker, even with limited skills, to take over the administration of the web sites.

5.1 Vulnerability Highlights

– 23 out of 42 (i.e., 54 %) of government websites are built on top of vulnerable CMS platforms
– 4 (i.e., 12 %) websites had accessible temp files with login credentials written in clear
– No government website were SSL compliant leading to insecure connections
– 19 (i.e., 45 %) websites were vulnerable to JCE attacks
– In 5 cases, we found that the database servers were directly accessible (from the internet).

5.2 Lessons Learned

Overall, in developing countries, more than anywhere else, there is a need to strengthen security training for government technicians. We now enumerate the lessons learned from these experiments before providing general advice for e-Government in developing countries.

Harden Websites – The first lesson learned from our study is the need for web maintainers to harden their installs. In particular, trivial steps can be taken to avoid a government web site to be randomnly found attackable. Concretely, a CMS should never be run in its default configuration for a government website. Maintainers should rename directories and tune the settings. For example, by default, Drupal is the only CMS that will lockout user accounts after a certain number of failed attempts. This means that unless a specific security plugin is installed, for other popular CMS such as Joomla! or Wordpress, hackers are free to brute force their login forms.

Always update Core CMS – The second lesson learned is on the need to update to newer versions of CMS, since CMS developers often try to patch vulnerabilities regularly.

Monitor website usage / baseline against abnormal usage – Most attacks, in particularly DoS attacks, can be stopped while they are being performed if their is a monitoring systems that raises alarm when an abnormal behaviour is detected.

Consider adding a firewall to virtually patch vulnerabilities internally – A firewall is a requirement in a government system to filter packets and enforce the accepted usage behaviour.

Separation of concerns: database servers should not be accessible from outside the network domain – Finally, we have found out that in many cases, by obtaining information on the database hosts we were able to directly attack it. It is very easy to address such issues by disallowing requests coming outside the government network domain.

Edition and customization artefacts, typically backup files – By simply scanning web directories for configuration files backed up during edition, a hacker can collect credentials for taking over the administration of a website. It is therefore important to clean web directory regularly during customization.

Taming Security holes This study and others somehow demonstrate that the security threat landscape is large. However, our findings also suggest that will simple tactics e-government portals can defend themselves. The essential of the defense system lies in awareness. In particular, it is important in developing countries that web maintainers understand that by raising the bar on protecting their websites, those will be safe from the attacks of today's industrialsed hacker who is only looking for the weakest web sites.

The second important means to tame security holes is to carefully monitor web services. Indeed, it is important to regularly review server logs to have real-time alert when an abnormal behaviour of the web service is detected so that maintainers can promptly investigate them.

Finally, e-government maintainers should always assume that any third-party code, including the CMS their website is based on, has numerous security vulnerabilities (we have shown it based on NVD data). To address such threats, it is necessary to deploy a security solution like a firewall for virtually patching vulnerabilities and mitigating new attack risks when they arise.

5.3 Discussion and Future Work

In this study all government websites were considered as equally important with regards to security needs. It is possible that some websites appear to be more critical as they allow real interactions through forms and official reports. Others however might have been setup to display information only. In future work we plan to investigate the critical resource in each website to evaluate the potential cost of attack.

Similarly, we plan to investigate in the future the reasons why such vulnerabilities remain unpatched. Typically, we will correlate the number of vulnerabilities with the IT budget of each department.

Finally, in continuity with our previous work [4,5] on identifying safety holes in operating systems, we will invest in the study of popular web programming

languages [6] to better understand how vulnerabilities are distributed across them.

6 Related Work

ICT4D research can no longer focus only on getting millions of people out of poverty [7]. Researchers must take into account the security issues that threaten an extended use of ICT. In particular the extended use of open source software comes with the responsibility of understanding that malicious developers have access to the source code, and thus can discover the exploitable vulnerabilities.

Vulnerabilities of E-Government websites across the world have been a concern for years. In 2007, Moen et al. [8] have investigated such websites in 212 countries and found that 81.6 % were vulnerable to XSS and SQL injection. Yet, such simple and well-known web application vulnerabilities can be avoided with well-known techniques. Today, in developed countries, much effort has been put into protecting government web sites. Our study, in particular in the case of Burkina Faso, shows that developing countries are still behind.

Although we have focused on technical aspects, including firewalls and website hardening, to provide the appropriate levels of security, previous work has shown that the human factor is also of high importance. A 2011 study by Bowen et al. have shown how the human factor influences cybersecurity policies and how that work could be used to train government employees to improve the security posture of government departments and agencies.

The literature contains a large body of related work on vulnerabilities. Wang et al. have proposed a mathematical model to calculate the severity and risk of a vulnerability. Their model is time dependent, taking into account exploitability, remediation level, and report confidence attributes [9]. Paleari et al. have focused their investigation on race vulnerabilities for web applications [10]. There have also been a number of research works on the challenges for mitigating program security vulnerabilities [11]. Finally, researchers have focused on proposing practical approaches to detect vulnerabilities in web applications. In this context, Ciampa et al. have focused on SQL injections [12].

7 Conclusion

In this paper we have investigated the security of government websites in a developing country, namely Burkina Faso. We have discussed the development of a E-government security testing framework, `s2e-gov`, which uncovered numerous vulnerabilities in government websites. In particular, we show how extensive and default configuration of CMS platforms contribute largely to making E-government websites vulnerable.

References

1. Shteiman, B.: How your CMS could be breeding security vulnerabilities (2013). http://www.itproportal.com/2013/10/08/how-your-cms-could-be-breeding-security-vulnerabilities/
2. Bissyandé, T.F., Ouoba, J., Ahmat, D., Sawadogo, A.D., Sawadogo, Z.: Bootstrapping software engineering training in developing countries. In: Nungu, A., Pehrson, B., Sansa-Otim, J. (eds.) AFRICOMM 2014. LNICSSITE, vol. 147, pp. 261–268. Springer, Heidelberg (2015). doi:10.1007/978-3-319-16886-9_27
3. Tan, L., Liu, C., Li, Z., Wang, X., Zhou, Y., Zhai, C.: Bug characteristics in open source software. Emp. Softw. Eng. **19**(6), 1665–1705 (2014)
4. Bissyandé, T.F., Réveillère, L., Lawall, J.L., Muller, G.: Diagnosys: automatic generation of a debugging interface to the linux kernel. In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineerinh, ASE 2012 (2012)
5. Bissyandé, T.F., Réveillère, L., Lawall, J.L., Muller, G.: Ahead of time static analysis for automatic generation of debugging interfaces to the linux kernel. Autom. Softw. Eng. **23**, 1–39 (2014)
6. Bissyandé, T.F., Thung, F., Lo, D., Jiang, L., Réveillere, L.: Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In: Proceedings of the 37th Annual International Computer Software & Applications Conference, COMPSAC 2013, pp. 1–10 (2013)
7. Bissyandé, T.F., Ahmat, D., Ouoba, J., Stam, G., Klein, J., Traon, Y.: Sustainable ICT4D in Africa: where do we go from here? In: Bissyandé, T.F., Stam, G. (eds.) AFRICOMM 2013. LNICSSITE, vol. 135, pp. 95–103. Springer, Heidelberg (2014). doi:10.1007/978-3-319-08368-1_11
8. Moen, V., Klingsheim, A.N., Simonsen, K.I.F., Hole, K.J.: Vulnerabilities in e-governments. Int. J. Electron. Secur. Digit. Forensic **1**(1), 89–100 (2007)
9. Wang, J.A., Zhang, F., Xia, M.: Temporal metrics for software vulnerabilities. In: Proceedings of the 4th Annual Workshop on Cyber Security, Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead, CSIIRW 2008, pp. 44:1–44:3. ACM, New York (2008). Observation of strains. Infect Dis Ther. 3(1), 35–43 (2011)
10. Paleari, R., Marrone, D., Bruschi, D., Monga, M.: On race vulnerabilities in web applications. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 126–142. Springer, Heidelberg (2008). doi:10.1007/978-3-540-70542-0_7
11. Shahriar, H., Zulkernine, M.: Mitigating program security vulnerabilities: approaches and challenges. ACM Comput. Surv. **44**(3), 11:1–11:46 (2012)
12. Ciampa, A., Visaggio, C.A., Di Penta, M.: A heuristic-based approach for detecting sql-injection vulnerabilities in web applications. In: Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, SESS 2010, pp. 43–49. ACM, New York (2010)