An Analysis of Security and Performance Concerns in Mobile Web Application Development: Challenges and Open Issues

https://doi.org/10.3991/ijes.v5i3.7330

Douglas Kunda(✉), Mumbi Chishimba, Mwenge Mulenga, Victoria Chama
Mulungushi University (MU), Kabwe, Zambia
dkunda@mu.edu.zm

Abstract—The paper focuses on security and performance concerns in mobile web development. The approach used in the study involved surveying journal publications to identify security and performance concerns. The paper highlights some of the contemporary issues currently being faced by application developers as they create, update and maintain mobile web applications including Cross-Site Scripting, Cookie hijacking/theft, location hijacking, history theft, behavior analysis, session hijacking, API design, security and the type of web server used considered.

Keywords—Mobile Web Apps, Open Issues, Performance and Security

1 Introduction

According to [1] "Mobile communication networks provide us flexibility for working at any time, nearly anywhere and in many forms." It is with this in mind that mobile browsing has become more and more popular in recent years and as a result, the development of mobile web applications has also become more and more popular.

[2] Define a mobile web application as "an app that uses an embedded browser to access and display web content." On the other hand, [3] define standard web application as "applications designed to work on desktop computer browsers." [3] Differentiate between mobile web application and standard web applications by stating that a mobile web application usually "renders the user interface controls (such as buttons, selectors, and textboxes) in a way that's similar to a native (mobile) app."

As mobile web applications are being developed, performance and security need to be considered in the development process. [4] State that "The increase in the variety of security attacks is partly due to the emerging number of sophisticated tools for attack penetration in both web and mobile applications." While security is often not in the fore-front of concerns when designing mobile web applications, they still remain as major security concerns as noted by [5] who state that "software security vulnerabilities caused by damage and loss, may impact a company operation or even harm the whole society." [5] Identifies two software security vulnerability factor categories namely "(1) software processes defects to generate security vulnerabilities" and "(2)

operation environment prevention defects to lead to maintenance and operation environment security vulnerabilities." Noting that the cost of repairing software security after the application has been developed is a very expensive venture, [5] states that security should be seriously undertaken in the early stages of designing software.

This paper will focus on security and performance concerns in mobile web development. The paper will discuss the challenges and open issues of the following aspects of mobile web development:

1. Web Services
2. Mobile Web Browser
3. API security and design
4. Web servers

2 Web-Services

[1] Define a web service as "a software system designed to support interoperable machine-to-machine interaction over networks." Considering that mobile web apps are also made with technologies such as JavaScript, web applications are also, like native mobile applications, making use of web services to load data asynchronously. Commonly used HTTP Web Service architectural patterns include the Simple Object Access Protocol (SOAP) and Representative State Transfer (REST). In terms of format, the two most popular web service format are JSON web services and XML web services. In terms of mobile web applications, while both formats are fully compatible with mobile web applications, JSON is generally more preferable due to the fact that JSON is less verbose than XML which is a huge factor to consider in cases where mobile data is a concern for users of mobile devices. There are a number of open issues with web services which include mobile device limitations, web service security and web-service performance. Open performance and security concerns for mobile devices include:

Table 1. Open issues in web services

Ref	Issue	Performance	Security
[6]	Limited battery power and the ability to function in changing network conditions	Yes	No
[6]	Un-validated input	No	Yes
[6]	Weaknesses in the protocols used	No	Yes
[6]	Man-in-the-Middle (MITM) attacks	No	Yes
[6]	Spoofing	No	Yes
[6]	Stateless nature of web services	Yes	No
[6], [7]	Data transfer cost	Yes	No

2.1 Mobile Device Limitations

A key concern in designing secure web services observed by [6] is that the limitations of mobile devices need to be taken into account when designing secure web services. These limitations include lower processing power of mobile devices, limited battery power and the ability to function in changing network conditions. For instance, if a highly interactive web application such as Facebook and Twitter were to be accessed by a user, the user would have to interact frequently with the application which may need to access web services to display content to the user. This would mean that the user would not only need to have a stable connection to the internet to access the service but also that due to the JavaScript constantly accessing the web services, battery life becomes a huge concern especially from the users' end. For instance, aspects such live notifications in a social networking application would need a developer to time how often their web application checks for content on the internet.

2.2 Web Service Security

Web service security is also a large concern in terms of mobile web development. This is especially true in today's world where privacy concerns are growing day by day. A good mobile web-app developer needs to be cautious and thoughtful as to how they develop and implement security in their mobile web applications lest their applications become compromised. According to [6], malicious activities that need to be taken into account when designing web services include:

- Un-validated input which may cause attacks such as SQL injection
- Weaknesses in the protocols used especially in the protocols are outdated
- Man-in-the-Middle (MITM) attacks
- Spoofing
- Message altering
- Cross-Site Scripting and Cross-Site Forgery where a web application is used to attack the browser on the other end in order to steal token information and other content.

To mitigate these issues, a number of security mechanisms are often employed in web services. These mechanisms identified by [6] include Open Authorization (OAuth), OpenID, Transport Layer Security (TLS/SSL), Token Based Authentication (sessions) and HTTP Digest Authentication Scheme (DAS). These authentication methods all have their advantages and disadvantages. For instance, while TLS/SSL can mitigate message sniffing and altering, MITM attacks and encrypt that data as it is being transported, it relies on the use of certificates which are relatively hard to use and cause a huge overhead with each connection setup. This is especially not optimal on mobile devices where processing time greatly affects battery life. OpenID on the other hand provides a general authentication service for multiple web applications but the need to remember the URI being used means that it is not stateless.

2.3 Web Service Performance

One performance concern that was noted by [6] is that since RESTful web services are stateless i.e., the web servers to not keep the state of a communication with a host, "data tends to be sent in a rather repetitive way, because no history is saved, possibly resulting in a larger overhead than typical state-full alternatives." It can also be noted that given that mobile devices web applications are the topic of discussion, it is important to note that since mobile devices generally have the option of using metered connections to the internet, JSON based web-services are recommended over XML based web services because JSON web services use a smaller amount of text to represent the same data as an XML based service which is important in cases where end users may have to pay for their internet connectivity per megabyte. This point is also supported by [7] who state that "Using JavaScript Object Notation (JSON)-formatted data is a good idea as it tends to result in a smaller data payload compared with an equivalent XML payload, depending on how the XML is formatted."

Generally, web services are important especially in today's internet age. With this in mind, developers need to be smart about how they not only develop web services but use them in their mobile web applications. Also, it is important to note that the choice of web service and their security implementation is also important and needs to be considered especially when developing web applications.

3 Mobile Web Browser

[8] Defines a web browser as "a client-side application, and its basic function is to fetch the content from the web servers and display it in the browser's windows" or in the case of mobile applications, the browser screen. "Mobile phones used to be extremely limited devices that were best used for making phone calls and sending short text messages" [9]. [9] Continues by stating that previously, developers had to write mobile specific interfaces in languages such as the Wireless Application Protocol (WAP) but in recent times, Mobile Web browsers support more complex functionalities such as CSS. JavaScript, images, video and may other features for a better and more improved experience. With the use of these richer features in mobile web browsers, a number of concerns have been brought to the forefront. [10] For instance, note that with the advent of more advanced browser technologies for mobile devices, "many mobile sites are poorly optimized for energy use and rendering them in the browser takes more power than necessary." This is true especially considering that there are many smartphone options to choose from especially if an Android based device is used where the power and for factor is very wide. All else considered, there are a number of security concerns that a developer needs to consider when developing mobile web applications. These include JavaScript performance, cross-site scripting and cookie theft/hijacking: Open performance and security concerns in terms of mobile web browsers include:

Table 2. Open issues for Mobile Web Browser

Ref	Issue	Performance	Security
[11], [12]	Cross-Site Scripting (XSS).	No	Yes
[12]	Cookie/Session theft.	No	Yes
[8]	Location hijacking	No	Yes
[8]	History Sniffing	No	Yes
[8]	Behaviour tracking	No	Yes
[7]	JavaScript performance	Yes	No

3.1 Java Script Performance

In terms of performance issues, [7] presented a number of issues that affect the performance of web applications on mobile devices. The biggest issue highlighted in the paper is the performance of JavaScript by showing the following chart.

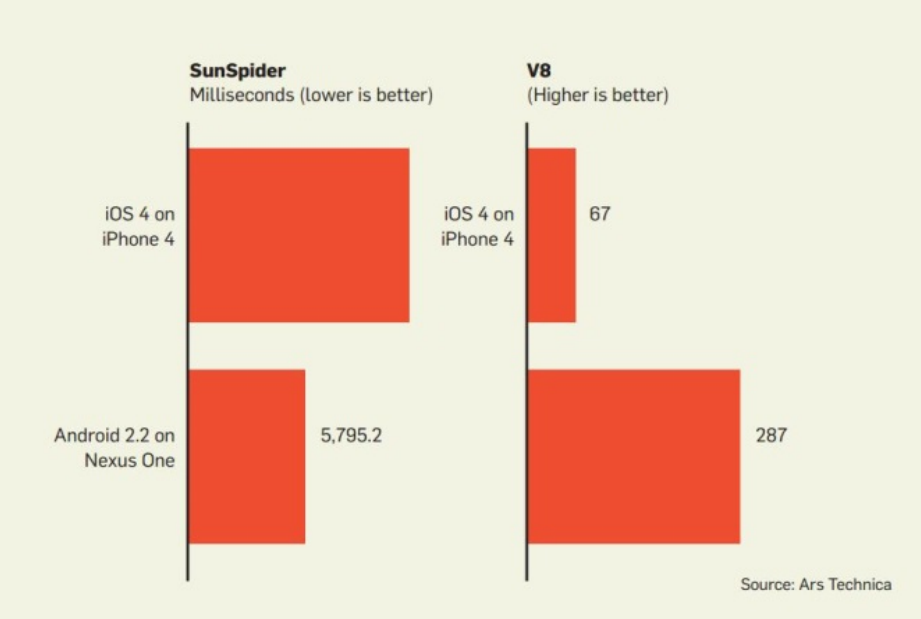

Fig. 1. Performance of JavaScript on an Android 2.2 device and an iOS 4 device.

Figure 1 shows the performance of JavaScript on an Android 2.2 device and an iOS 4 device. The results show that the Android device performed better than the iOS device. Though the figure 1 does not represent all possible use cases, it shows that JavaScript will perform different given a different hardware and software configuration. When developing an application that depends on JavaScript, a developer needs to take into consideration how they design their application and try to make it as fast as possible so that web applications perform well on as many devices as possible.

Another issue of concern of JavaScript is the issue of dynamic loading of content from a web service. For instance, when dynamically loading data from a web service, "XML data can make sense when returning HTML fragments that are to be inserted into a Web page rather than returning JSON-formatted data that, while smaller over the wire, needs to be converted to an HTML fragment using JavaScript" [7]. The process of converting JSON data to HTML elements may prove to be a significant overhead especially in cases where the application is heavily dependent on web services.

Aside from dynamic loading of data in a web application, [7] also note that once JavaScript code is downloaded by the web browser, the code still needs to be loaded into memory and then still needs to be parsed so that the browser can actually execute the code. This presents a significant overhead in cases where the developer is using a lot of JavaScript code in their application.

3.2 Cross-Site Scripting

One of the greatest threats to consider in the development of Mobile Web application is the aspect of Cross-Site Scripting (XSS). [11] "This vulnerability occurs when a web application uses inputs received from users in web pages without properly checking them." This type of attack allows an attacker to inject code that performs malicious actions when a browser visits a web page which has been exploited. These types of attacks may cause severe breaches in security such as Account Hijacking and Cookie/Session theft. [12] State that Cross-site scripting (XSS) is a "security vulnerability that affects web applications. It occurs due to improper or lack of sanitization of user inputs. The security vulnerability caused many problems for users and server applications." If a third party were to do simple reconnaissance on a Mobile Web Application and find out which cookies and login sessions are used, the attacker would find it relatively easy to hijack a user's session.

According to [8], a big reason why XSS is a big possibility is that "When a remote JavaScript code is included into a web page, it gets the same privileges as any other code originally inlined in the page." [8] States that "JavaScript program is executed by the JavaScript interpreter of the web browser and can access a set of available APIs implemented by the browser. These APIs allow the script to communicate with other elements of the page (by DOM APIs), local browser data (such as cookies) or remote servers (by several communication APIs), and to manipulate the web page elements and other browser events". This means that if Google's JavaScript ran on the same web page other more malicious code, the malicious code will have the same level of permissions as Google's JavaScript code. This would allow the malicious code to steal cookies and other site data. [8], also impresses on this vulnerability by stating that as of 2013 among the top 10, 000 websites, 88.45% include remote JavaScript in their java code.

A way to tackle XSS as noted by [8] is through the use of a security policy called the Same-Origin policy (SOP). The Same-Origin policy is a policy which basically states that JavaScript code can only read HTML elements that have the same origin as the JavaScript code. For instance, JavaScript from Google can only read elements

from the Google domain even if elements from a domain like Wikipedia or Amazon were included via an iFrame. As novel as this solution appears, the author is quick to point out that SOP, while an effective tool, does not prevent the leakage of sensitive user information since once malicious JavaScript accesses this information, the malicious code can transmit the data to the attackers machine.

3.3 Cookie Theft/Hijacking

Another security threat on the part of browsers is cookie theft/hijacking. In a 2016 in-depth assessment of a diverse set of major websites and explore what functionality and information is exposed to attackers that have hijacked a user's HTTP cookies by [13], it was noted that "38% of mobile users have experienced mobile cybercrime in past 12 months (of 2016)."

3.4 Other Threats

Other identified threats include:

- Location hijacking where "an untrusted JavaScript code can either influence the document's location directly or the string variables that are forming a URL" [8] which can cause a web page a user has opened to navigate to an untrusted page of the attackers' choice. This is a big security breach in that privacy is a very big issue in today's computing world.
- History sniffing which, as the name suggests, allows an attacker to "check whether the user has ever visited a specific URL" [8]. Like location hijacking, this is a big security issue especially if a user is particular about privacy or in cases where a user is performing sensitive operations such as purchasing items on the internet. An attacker can easily use a user's browsing history to profile a user's behavior.
- Behavior tracking where "A script running on a web site can gather precise information about the user's mouse clicks and movements, scrolling behavior, what parts of the page are highlighted, and clipboard content" [8]. With this type of breach, an attacker can study a user's behavior to profile a user and find out when the user is most vulnerable to attack.

Generally, a mobile web application has to consider a number of issues when it comes to developing a web application. Aside from considering that users use different types of browsers having different design and behavior implementations, the users need to also consider how the different browsers implement security. We have discussed threats like Cross-Site scripting, Cookie Theft/Hijacking and a number of other security issues such as location hijacking and behavior tracking. Overall, a developer needs to be able to choose how much security they will put in in their application to optimize performance and security.

4 API design and security

"Modern-day software development is inseparable from the use of Application Programming Interfaces (APIs)." [14]. [15] state that "Web APIs provide a systematic and extensible approach for application-to-application interaction." They go further to state that a large number of mobile applications use web application programming interfaces (APIs) and that in this modern age of software development, "Software developers access APIs as interfaces for code libraries, frameworks or sources of data, to free themselves from low level programming tasks or speed up development." From these statements, we are able to understand how important web APIs are to the development of some software. Commonly used APIs include Etsy, Freeagent, Dropbox, Google Documents, Slideshare, Soundcloud, Foursquare, Google translate, Facebook, Instagram, Tumblr, YouTube and many others. Some open performance and security API issues include:

Table 3. Open issues in API design and security

Ref	Issue	Performance	Security
[16]	APIs are consistantly modified to fix bugs and to patch up security flaws	Yes	Yes
[16]	Announcement/feedback rate	Yes	Yes
[16]	Class templates used	Yes	Yes
[16]	CRUD operations provided by the API	Yes	No
[16]	Form validation	No	Yes
[16]	Import/export of data	Yes	No
[16]	User Authentication	No	Yes

4.1 Challenges in API usage

As good as web services may be, [16] admit that the process of a developer integrating a Web API into an application can be challenging. The biggest difficulty identified by the authors is that web APIs are consistently modified to fix bugs and to patch up security flaws. This also means that developers need to update their application code to match the design specifications of the original API creator. In this scenario, it is the API creator that sets the pace for the API creators. API creators such as Facebook offer a grace period to developers after which their applications will simply not be able to use the web API. This is also compounded by the fact that if the return on investment for migrating to a new API is not justified; most client developers will not migrate to the new API.

In their study on common issues encountered by developers in APIs, [16] discovered that some of the common issues include the announcement/feedback rate, authorization/authentication, the display styling of the APIs, class templates used, CRUD operations provided by the API, form validation, import/export of data, Subscription and User Authentication just to name a few.

In a study of various APIs, [14], analyzed a number of web APIs namely the Google Maps, the Twitter APIs, Facebook and Netflix and observed a number of interesting findings. For instance, the Google Maps API allows clients to display Google provided maps in their application. The biggest advantage this API has is that the API offers extensive backwards compatibility. Users of this API are given a generous 1-year grace period for client developers to update their API usage with few exceptions. The Twitter API on the other hand has no official deprecation policy but instead opts for a 6-month announcement period. Extreme instances of changes to the Twitter API identified by the authors include being forced to authenticate and the abandonment of XML support in favor of JavaScript Object Notation (JSON). The Facebook web API according to the authors allows for client applications to change a number of features to the client developer. The one gripe that the author discovered about the Facebook API is that that Facebook API enforced a strict 90-day window for client developers. The Netflix API on the other hand, like Twitter, has no official deprication policy. With these APIs, we have a simplified idea of how API updates by major organizations are done.

4.2 API security

In terms on API security, "the modern Web API provides a key role in the mobile cloud computing infrastructure" [17]. [17] go on to state that the Web API technology has seen a lot of growth and adoption in the enterprise world due to increasing demands in the field of distributed computing. With the distribution of the computing, there is also a need for Web APIs to be secured especially for organizations that deal in sensitive information. [17], highlight the importance of protecting Web APIs by highlighting the incident where a Twitter security breach where 250, 000 accounts were exposed and an estimated 42% of Web API consumers encountered security issues.

In terms of Web API, the SOAP web APIs are more secure than the REST variety. However, [17] are quick to point out that REST web service APIs have become more mainstream due to their "simplicity and ubiquity." SOAP web APIs use the WS-* security infrastructure while the REST Web APIs use a more simplified method. REST web APIs are generally more vulnerable to injection attacks and other types of web service attacks due to its weak built in security features.

[17] Identify two types of API security paradigms which are weak API security and strong API security. Weak API security and its resources, according to the authors, are protected using methods such as basic HTTP authentication and SSL security. Methods under the HTTP/SSL method include the traditional TLS security method, OAuth2 authentication and authorization, OpenID authentication, Security Token Service (STS) service, sensitive data encryption and security monitoring and attack prevention. The authors also state that "Username and password pair based on weak API security is vulnerable, since the passwords have been stolen, cracked, phished, guessed, sniffed, captured, and leaked." Given the points presented, it is possible to conclude that applications that deal with sensitive data should generally avoid using weak API security while applications that to not deal in overly sensitive data have the

option of weak API security to improve on performance. Strong API security on the other hand offers better security infrastructure than weak API security. Strong API security protects its resources using a three-factor security mechanism which makes it more secure than weak API security. The three factors are:

- "Basic authentication which includes API user registration, SSL, and strong password protection"
- "Adopt modern standard-based security mechanism specified in API Security" which integrates the Identity Management (IdM) and (Lightweight Directory Access Protocol) LDAP and provide REST JSON message level security, such as JSON Web Token (JWT), JSON Web Signature (JWS), and JSON Web Encryption (JWE)
- "Adopt security mechanism between API and its backend services as the third security factor" for protecting the backend services.

Generally, APIs are a good and popular way of enhancing the performance and the look-and-feel of the mobile web application. For instance, as opposed to creating a whole new maps API, a developer can simply make use of other APIs such as the Google Maps API to speed up application development and the look and feel of their web application. At the same time though, the developers need to also consider which features are provided by the API they are using by not only considering the methods and operations provided by the API but also the security features that are offered by the API.

5 Web Servers

5.1 Web-Server Security

[22] Define a web server as "a program that serves a content request using the HTTP protocol." They go further by stating that a web server is an indispensable server-side component of the web-based application architecture. The authors identify two types of resources that can comprise an HTTP request. The first of these request types are static resources which, no matter how many times they are requested, do not change. These include resources such as videos, images and static web pages amongst many others. The authors state that for these types of resources, the web server simply needs to fetch the content and deliver it to users. The advantage of web servers that only deliver static content is that the web servers do not use extra resources to process the request. The other type of request is a dynamic web request which, as the name suggests, delivers dynamic resources to the user. Users will pass data in the form of parameters which may then be processed by a web server to present dynamic content to a user. Current popular web servers include Apache (PHP), Apache Tomcat, Microsoft IIS, Nginx and Google GWS. Factors that may influence the choice of web server include familiarity with the web server, scope of project being developed and the budgeting.

For mobile web applications, the greater demand is on dynamic content. This is due to an increase in the dynamism of user behavior especially with the advent of social networking, blogs, wikis and e-commerce [19]. Another factor to consider for performance is that the web server needs to take advantage of the architecture of the machine it is using. For example, if the web server hardware has a multi-core setup, the web server software, in order to perform better, needs to have multi-core support in order to take full advantage of the hardware. [20], in support of this point state that "In order to improve performance of handling user requests, most of web servers adopt multi-core processors." [20], further state that web servers that use scheduling algorithms such as First-Come-First-Served do not fully take advantage of multi-core architecture and as a result would slow down performance for client users.

5.2 API security

Web server security is also an important consideration to take though the extent to which security is implemented is entirely dependent on the type of application being developed. While some web servers may be generally considered more secure than other web servers, the true extent of security is entirely dependent on the web-server developers and how often they patch up security flaws. In a study on assessing web server security, [23] analyzed five open-source web servers including Apache HTTPD2, Apache Tomcat 6.0.13, Apache Tomcat 6.0.16 and Apache Tomcat 6.0.13. They analyzed factors such as security policy, access control, communication and operation management, human resource security, information systems acquisition, development and maintenance and physical and environmental security. After applying these metrics to the web servers by measuring the rate of tests failed versus the rate pa tests passed, the authors discovered that Apache Tomcat 6.0.13 provided better security. While this was the case at the time the authors conducted the experiment, the developers of the web-servers could have patched the web servers in higher versions and sealed up the security flaws.

In terms of web servers, a mobile web application developer needs to take into consideration when type of application that they will be using and what type of web server to use in their application. Factors such as cost and skill with the web server are factor that may affect which web server a developer uses to provide functionality to their mobile web application.

Another security issue for developers in terms of web servers is the issue of Denial of Service (DOS) attacks. [21] Define a Denial of Service attack as "an attempt to make a service unavailable to legitimate users." Another danger of DoS attacks, highlighted by [24] is that "The DOS attacks can cause severe damage to the interconnected systems such as web servers, database servers, cloud computing servers, etc." For a mobile web application developer who, under usual circumstances, uses a web server and a database at minimum to run their application, DoS attacks are an especially bigger threat. A type of Denial of Service attack according to [25] is an application layer DDoS attack because Application Layer DDoS attacks can be initiated with minimal computational resources by the attacker. [25] Offer some obvious DDoS mitigation techniques such as keeping an attacker oblivious of the DDoS mitigation

techniques being used and creation of a decoy server where requests that are deemed to be malicious are redirected to.

6 Conclusion

In conclusion, this paper has undertaken an analysis of challenges and open issues in security and performance concerns in mobile web design. The first issue tackles were web-services. A number of design and security aspects such as the choice of web service format such as whether or not to use JSON or XML for design and the type of authentication used in terms of security.

In terms of mobile web browsers, it was discussed that in terms of design, most modern day mobile devices have up-to-date web browsers which, for the most part, have the same functionality as the desktop web-browser counterparts. The greatest concern for mobile web application design in terms of the web browser is that web applications may look different on devices having different screen sizes and a developer needs to keep this in mind when designing a web application. The greatest security threats for mobile web browsers are JavaScript based. For instance, Cross-Site Scripting, Cookie hijacking/theft, location hijacking, history theft, behavior analysis and session hijacking are a considerable threat against users.

In terms of API design and security, a client developer needs to consider a number of factors when selecting which API they will be using in their mobile web applications. Common issues in selection of APIs to be used in mobile web applications include the announcement/feedback rate by the API developer, authorization/authentication methods offered by the API, the display styling of the APIs elements, class templates used in the API, CRUD operations provided by the API, form validation, import/export of data, Subscription options and User Authentication options just to name a few.

Finally, in this paper, the aspect of web servers was discussed. In this section of the paper, it was discussed that the type of content that will be offered through the web server, the relative skill of the developer in working with the web server and the cost willing to be invested in the web server is a factor to consider when choosing a web server. It was also discussed that in terms of security, if a developer is working on a mobile web application that requires a greater level of security such as e-commerce applications, it is important for a developer to be well informed about the security levels offered by the web server. This can be done by simply checking the change log for the web server and reading about flaws that have been found about the web server.

A developer of mobile web applications needs to be relatively informed about a number of issues in mobile web development such as web-services that are available for the web application, the mobile web browsers to be used by the users of the mobile web application, APIs that are available for the mobile web application and the web server that will be used to host the web application.

7 References

[1] J. Zhang, Y. Wang and V. Varadharajan, "A New Security Scheme for Integration of Mobile Agents and Web Services," Second International Conference on Internet and Web Applications and Services, 2007. https://doi.org/10.1109/ICIW.2007.5

[2] P. Mutchler, A. Doupe, J. Mitchell, C. Kruegelz and G. Vignaz, "A Large-Scale Study of Mobile Web App Security," In Proceedings of the Mobile Security Technologies Workshop (MoST), 2015.

[3] N. Serrano, J. Hernantes and G. Gallardo, "Mobile Web Apps," Software Technology, pp. 22-27, 2013.

[4] D. Nyambo, Z. Yonah and C. Tarimo, "Security Frameworks in the Converged Web and Mobile Applications: A Review," Pan African International Conference on Science, Computing and Telecommunications, pp. 29-34, 2014. https://doi.org/10.1109/SCAT.2014.7055131

[5] S.-T. Lai, "An Interface Design Secure Measurement Model for Improving Web App Security," International Conference on Broadband and Wireless Computing, Communication and Applications, pp. 422-427, 2011.

[6] F. D. Backere, B. Hanssens, R. Heynssens, R. Houthooft and A. Zuliani, "Design of a Security Mechanism for RESTful Web Service Communication through Mobile Clients," IEEE, 2014.

[7] A. Charland and B. LeRoux, "Mobile application development: Web vs. native," CommunICatIons of the ACM, vol. 54, no. 5, pp. 49-53, 2011. https://doi.org/10.1145/1941487.1941504

[8] N. Bielova, "Survey on JavaScript security policies and their enforcement mechanisms in a web browser," The Journal of Logic and Algebraic Programming, p. 243–262, 2013. https://doi.org/10.1016/j.jlap.2013.05.001

[9] N. C. Zakas, "The Evolution of Web Development for Mobile Devices," ACM, pp. 1-10, 2013.

[10] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh and J. Singh, "Who Killed My Battery: Analyzing Mobile Browser Energy Consumption," Session: Mobile Web Performance, pp. 41-50, 2012.

[11] L. Shar and H. Tan, "Automated removal of cross site scripting vulnerabilities in web applications," Information and Software Technology, p. 467–478, 2012. https://doi.org/10.1016/j.infsof.2011.12.006

[12] I. Hydara, A. B. M. S, H. Zulzalil and N. Admodisastro, "Current state of research on cross-site scripting (XSS) – A systematic literature review," Information and Software Technology, p. 170–186, 2015. https://doi.org/10.1016/j.infsof.2014.07.010

[13] S. Sivakorn, I. Polakis and A. D. Keromytis, "The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information," Symposium on Security and Privacy, pp. 724-742, 2016. https://doi.org/10.1109/SP.2016.49

[14] T. Espinha, A. Zaidman and H.-G. Gross, "Web API growing pains: Loosely coupled yet strongly tied," The Journal of Systems and Software, pp. 27-43, 2015. https://doi.org/10.1016/j.jss.2014.10.014

[15] T. Espinah, A. Zaidman and H.-G. Gross, "Web API Fragility: How Robust Is Your Mobile Application?," ACM International Conference on Mobile Software Engineering and Systems, pp. 12-21, 2015.

[16] P. K. Venkatesh, S. Wang, F. Zhang, Y. Zou and A. E. Hassan, "What Do Client Developers Concern When Using Web APIs? An Empirical Study on Developer Forums and Stack Overflow," IEEE International Conference on Web Services, pp. 131-138, 2016.

[17] L. Tang and L. Ouyang, "Multi-Factor Web API Security for Securing Mobile Cloud," International Conference on Fuzzy Systems and Knowledge Discovery, pp. 2163-2168, 2015. https://doi.org/10.1109/FSKD.2015.7382287
[18] Y. Li, L. Guo, Z.-H. Tian and T.-B. Lu, "A lightweight web server anomaly detection method based on transductive scheme and genetic algorithms," Computer Communications, p. 4018–4025, 2008. https://doi.org/10.1016/j.comcom.2008.08.009
[19] R. Peña-Ortiz, J. A. Gil, J. Sahuquillo and A. Pont, "Analyzing web server performance under dynamic user," Computer Communications, p. 386–395, 2013.
[20] G. You and Y. Zhao, "A Dynamic Weight-based Dynamic Requests Scheduling Model in Multi-core Web Server," International Conference on Service Sciences, pp. 112-117, 2014. https://doi.org/10.1109/ICSS.2014.20
[21] R. A. Oliveira, N. Laranjeiro and M. Vieira, "Assessing the security of web service frameworks against Denial of Service attacks," The Journal of Systems and Software, pp. 18-31, 2015. https://doi.org/10.1016/j.jss.2015.07.006
[22] X. Qiao, G. Nan, W. Tan, L. Guo, J. Chen, W. Quan and Y. Tu, "CCNxTomcat: An extended web server for Content-Centric Networking," Computer Networks, p. 276–296, 2014. https://doi.org/10.1016/j.comnet.2014.10.014
[23] N. Mendes, A. A. Neto, J. Durães, M. Vieira and H. Madeira, "Assessing and Comparing Security of Web Servers," 2008 14th IEEE Pacific Rim International Symposium on Dependable Computing, pp. 313-322, 2008. https://doi.org/10.1109/PRDC.2008.45
[24] A. P. Ranekar and A. R. B. Patil, "Survey of DOS Defense Mechanisms," IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems ICIIECS'15, 2015. https://doi.org/10.1109/ICIIECS.2015.7193058
[25] J. D. Ndibwile, A. Govardhan, K. Okada and Y. Kadobayashi, "Web Server Protection against Application Layer DDoS Attacks using Machine Learning and Traffic Authentication," IEEE 39th Annual International Computers, Software & Applications Conference, pp. 261-267, 2015.

8 Authors

Douglas Kunda is currently working as Dean of the School of Science, Engineering and Technology at Mulungushi University in Kabwe, one of the public Universities in Zambia. He is a Fellow Member and Past President of the Information and Communication Technology Society of Zambia (ICTSZ). Dr Kunda is a Member of Association for Computing Machinery (ACM). He worked as the Project Manager for the Integrated Financial Management Information System (IFMIS), a large ICT project for Government of the Republic of Zambia. He is a holder of Doctorate degree in Computer Science from the University of York, United Kingdom. Dr Kunda has more than 20 years' experience in Information Communication Technology (ICT) development and implementation; Software Engineering and Project Management including project/programme design, implementation and review of large ICT projects and multi-donor programmes. Dr Kunda has presented more than 20 papers at International Conference and published 20 papers in peer reviewed journals.

Mumbi Chishimba is a Master of Computer Science Research student in the School of Science, Engineering and Technology at Mulungushi University in Kabwe. He holds a Bachelor's Degree in Computer Science from Mulungushi University with

Merit. He works also works as tutor for Mulungushi University. He has software development experience in in Java, C++, PHP/ MySQL web platforms, Microsoft products including SQL Server databases, Oracle and Postgresql databases.

Mwenge Mulenga is Assistant Dean and Lecturer in the School of Science, Engineering and Technolog at Mulungushi University in Kabwe, Zambia. He holds a Master degree from St Petersburg, RUSSIA. He has vast experience in in Java, C++, PHP/ MySQL web platforms, Microsoft products including SQL Server databases, Oracle and Postgresql databases.

Victoria Chama is currently a Staff Development Fellow in the School of Science, Engineering and Technology at Mulungushi University in Kabwe. She holds a Bachelor's Degree in Computer Science from Mulungushi University with Distinction. She has experience in Spring Java, PHP, Oracle and MySQL databases. She has participated in a number of consultancies, for example in design and development of Student Information Systems. She has published in one journal paper.

Article submitted 22 June 2017. Published as resubmitted by the authors 21 July2017.