Finding a Safe Port: Cyber-Security Analysis for Open Source Digital Library Software

Alex Tudu
at239@students.waikato.ac.nz
University of Waikato
Hamilton, New Zealand

David Bainbridge
davidb@waikato.ac.nz
University of Waikato
Hamilton, New Zealand

Bill Rogers
coms0108@waikato.ac.nz
University of Waikato
Hamilton, New Zealand

ABSTRACT

This article presents the results of an investigation into how safe, from a cyber-security standpoint, our Open Source Digital Library (DL) systems are. The fact that these systems use open source software presents particular challenges in terms of securely running a web-based digital repository, as a malicious user has the added advantage that they can study the source code to the system to establish new vectors of attack, in addition to the many well documented black-box forms of web hacking. To scope the work reported we focused on two widely used digital library systems: DSpace and Greenstone, undertaking both Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST), in addition to more traditional port scans. We summarize the deficiencies found and detail how to make improvements to both systems to make them more secure. We conclude by reflecting more broadly on the forms of security concerns found, to help inform future development of DL software architectures.

CCS CONCEPTS

• **Information systems → Digital libraries and archives**; • **Networks → Network security**; • **Social and professional topics →** *Systems development*.

KEYWORDS

Cyber-security, Open Source Software, Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Digital Library Software

ACM Reference Format:
Alex Tudu, David Bainbridge, and Bill Rogers. 2020. Finding a Safe Port: Cyber-Security Analysis for Open Source Digital Library Software. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020 (JCDL '20), August 1–5, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3383583.3398612

1 INTRODUCTION

Cyber-security has come to prominence in recent years. Computers and other forms of electronic devices have undoubtedly made many of our daily tasks easier to accomplish, but the prevalence of the digital platforms that enable this has also increased our exposure to risks, with hackers continually seeking to exploit deficiencies in these systems to tamper with personal information, extort money, and engage in other malicious acts [9].

There is a need for every individual and organization to understand the importance of cyber-security. In 2019, for example, it was estimated that globally a business fell victim to a ransomware attack every 14 seconds—a frequency that is predicted to increase to every 11 seconds by 2021 [8]. Concerning breaches of personal data, in the same year, the cyber-security firm UpGuard reported on the discovery of a staggering 550 million Facebook records, totalling 146 GB in size, that had been negligently left exposed on an Amazon Cloud service by Cultura Colectiva, a 3rd party vendor to the social media giant [6].

In this article we focus on Digital Library software and assess, from a cyber-security perspective, the risks associated with this form of web-based system. In particular, we undertake a study of two widely used Open Source Digital Library systems: DSpace and Greenstone. We start the article with a review of relevant literature and cyber-security analysis tools. Section 3 details the results of port-scanning installed instances of the latest versions, at the time of writing, of DSpace and Greenstone. The sections following this move on to presenting the results of more intensive forms of analysis: Static Application Security Testing (SAST)—of particular concern because the DL systems are Open Source—and Dynamic Application Security Testing (DAST). In Section 6 we summarise our findings and conclude by reflecting more broadly on the forms of security concerns found, to help inform future development of DL software architectures.

2 BACKGROUND

There is published literature describing the design and technical capabilities of the two selected DL systems produced by their respective development teams: see for example [11], or [12] for DSpace, and [3], [5], or [14] for Greenstone. There are also review articles by others, such as [4], [10], and [13] that compare and contrast the strengths and weaknesses of different open source digital library systems. Notably lacking in these articles, however, are any specific details of the cyber-security risks associated with deploying such a system as a public facing web resource. Since a DL system operates in tandem with a web server, one could argue that in these articles there has been an unwritten assumption that the role of network security is handled by the web server; but, even if this was the reasoning at the time, it is insufficient for the on-line environments in which such repositories operate in today, and the key observation that motivated the work reported here.

ACM ISBN 978-1-4503-7585-6/20/08…$15.00
https://doi.org/10.1145/3383583.3398612

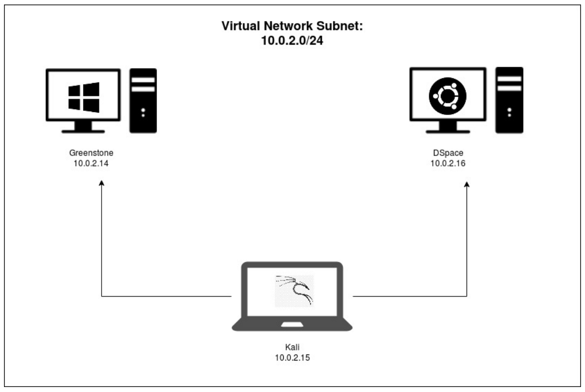

Figure 1: Setup of virtual machines and network map used for security assessments of Greenstone and DSpace.

More closely aligned with our work is that of Anwar and Shibli [2], where the topic of security is front and center. In their work, the DSpace software architecture is extended to include Attribute Based Access Control (ABAC) to deliver fine-grained authorized control to resources within the repository. Their use of the term security, while related, is different to our use. Their work is concerned with extending the security functionality of an internal component of the DSpace design. Our work is concerned with external security vulnerabilities that a DL architecture might be exposed to—such as the ability to exploit user credential information through an SQL injection attack—typically caused by the interplay between components at the boundary between DL design and web server when in operation.

Figure 1 shows the setup of the virtual machines and network map we used to run our security assessments. We chose Windows 10 for the Greenstone virtual machine, as the binary version of the software for this operating system is the one most often used.[1] Greenstone v3.09 was installed. The installation procedure for DSpace centres on running a sequence of Java based tools and so, fundamentally, is operating system agnostic. Reflecting on the prominence in the DSpace Wiki given to operation on distributions of Unix,[2] we selected Ubuntu 18 as the base virtual machine for the installation of this digital library system. DSpace v6.5 was installed.

Concerning analysis tools, we set up a third virtual machine using Kali, a distribution of GNU/Linux that is geared specially for digital forensics and penetration testing. In particular, this comes pre-configured with Nmap, Nessus, and Metasploit, which are tools that we made use of during the port scanning phase of our testing.

For Static Analysis Software Testing (SAST) of the two code bases, we selected SonarQube[3] as the environment in which to conduct the assessment, after a review of pertinent tools. SonarQube is itself an open source project, and was notable in its plugin-in support which included Apache Ant and Maven, the respective build tools used by Greenstone and DSpace.

For Dynamic Analysis Software Testing (DAST), we chose the Zed Proxy Attack (ZAP) tool from the Open Web Application Security Project (OWASP). This was based on a review of DAST tools by Makino and Klyuev [7] where, in trials, this tool detected the most number of vulnerabilities.

3 PORT SCANNING

The Nmap command used to scan the DSpace server was:

```
nmap -p- -sS -sU -sC -sV 10.0.2.16
```

For the specified IP, this scans all ports (0–65535), directing the tool to test for TCP and UDP activity using the default set of heuristic scan scripts to identify the type of server present on a given port, and if possible the version as well. Changing IP number appropriately, a scan with the same command-line arguments was run against the Greenstone server.

For the DSpace scan, 2 ports were found to be open: 8080 and 8009; for Greenstone 4 ports: 5001, 8309, 8383, and 7680. The fourth port from the Greenstone scan (7680) was found to be related to the Windows update service, and while any risk borne by this exposure should always be kept in mind if operating on a DL server on this type of operating system, as it was unconnected with the actual running of the digital library software we discounted it from our subsequent analysis.

For both installations, scanning revealed that each DL system was operating through a Tomcat web server (8080 and 8383 respectively for DSpace and Greenstone), and further the scan was able to determine the version number of the particular Tomcat being utilized (respectively 8.5.46 and 8.5.35). Cross-checking the Tomcat versions reported with Common Vulnerabilities and Exposures (CVE),[4] showed no known issues (at the time of writing) with these versions of the server.

The other port detected by the DSpace scan, 8009, was identified as the Apache JServ Protocol (v1.3). AJP13 is a Java-oriented protocol that operates in a similar manner to reverse-proxying, allowing a different web server to act as the public facing connection point. The Greenstone scan found port 8309 to be open, and while it was not explicitly reported as AJP13 (because it is not using the regular port number associated with this protocol) it only takes a little bit more work to establish this is what it is. There are several CVE reports relating to AJP exploits connected with Tomcat up to version 7, but none currently posted that affect the version 8 servers used by DSpace and Greenstone.

The final open port. 5001, occurred on the Greenstone virtual machine and was detected as responsive to Java Remote Method Invocation (Java RMI), and tallies with the use of SOAP (Simple Object Access Protocol) as a message passing mechanism in its design [5]. To test for vulnerabilities, we ran the RMI exploit module of Metasploit on our Kali machine against port 5001 of the Greenstone server. The attempted attack ended with a report from Metasploit detailing its failure to establish a session with the server through the exposed port.

As part of our analysis, we also ran Nessus, using its default settings, against DSpace and Greenstone. This tool reports issues in the following order of severity: Critical, High, Medium, Low, and Info. For DSpace it produced a report consisting of 1 Medium and

[1]Based on the Greenstone factsheet, http://www.greenstone.org/factsheet
[2]https://wiki.lyrasis.org/dsdoc6x/installing-dspace
[3]https://www.sonarqube.org/

[4]https://cve.mitre.org/cve/

17 Info items. For Greenstone it produced 1 Medium and 12 Info items. Focusing in on the Medium category, both DL systems were found to be susceptible to the same issue on the Common Vulnerability Scoring System (CVSS): Plugin ID 12085, which refers to a default index page, error page, or other servlets and pages installed in an Apache Tomcat server allowing access to information related to the server. The key issue with both servers that Nessus reports, then, is in line with what our Nmap scan also revealed about server versioning information being revealed.

3.1 Discussion

From a cyber-security standpoint, it is best practice to not reveal to external connections the type and version of a server running on the machine. Both DSpace and Greenstone would benefit from tightening up on this aspect of their default installments. Concerning AJP13, neither default installation makes explicit use of this protocol, and so an improvement would be for this to be initially disabled, and instructions included on how to activate this if setting up a front-end server that uses AJP to connect to the DL. A similar line of argument applies to the use of Java RMI in Greenstone, since external in-coming SOAP requests are not supported in a default installation. There is no need for this to be on initially, and so it too should be disabled in the default configuration files supplied.

4 STATIC APPLICATION SECURITY TESTING

We utilized SonarQube to undertake Static Application Security Testing of the DSpace and Greenstone code bases. Because this is a tool designed for supporting code quality management in general, we focused our attention in the generated reports on the items related to the category of Security. Identified issues are marked as Bug, Vulnerability, or Code Smell, which then—in decreasing order of severity—are further classified as Blocker, Critical, Major, Minor or Info. Concerning the use of the term "Bug," in the context of code quality management software, the role of the label is more a case of signalling a "Potential Bug," although it is more cumbersome to continually express as the latter. The section of code identified should be reviewed, in light of the specific "bug" the tool has identified—for instance not using a *finally* clause to close a connection—and if warranted the code is revised.

The particular command-line tool for analyzing a code base ('scanning' in SonarQube's parlance) is based around a plugin architecture designed to support a wide range of programming languages such as Java, C/C++, and C#, as well as compilation build tools such as Gradle and MSBuild. To process the Greenstone source code the SonarScanner for Ant plugin was installed, and the DL's top-level *build.xml* file was updated to utilize the scanner. To process the DSpace source code, its top-level Maven file *pom.xml* was updated in a similar fashion to the Greenstone change, only there was no need to explicitly install the Scanner plugin for this build tool as it is downloaded as part of the process of initiating the scan.

Factoring in all aspects of software management, for DSpace, SonarQube identified 750 Bugs, 1834 Vulnerabilities, and some 13,000 Codes Smells. For Greenstone, SonarQube identified 187 Bugs, 378 Vulnerabilities and some 7,800 Code Smells As noted above, the Draconian sounding Bug label is not actually as severe as it first sounds. Taking those identified in the Greenstone code as an example, the 187 "bugs" fell into one of three categories, triggered by certain code patterns used by the development team. A review of these patterns concluded that the programming practices were in fact sound, as coded, but the code would benefit from changes inline with SonarQube's recommendation, as it was a clearer way for things to be expressed.

Concerning Security issues specifically, there were 9 locations spanning 3 kinds of issues marked as Critical in DSpace, and 11 locations all of the same kind marked as Critical in Greenstone. The type of Critical error identified in Greenstone coincided with one of the kinds of errors identified in DSpace, and related to how the *Transformer()* function from the *javax.xml.transform* package in the Java Platform, Standard Edition API was being used. Upon review of both DL systems, a particular coding pattern was again in evidence, only this time its implications in terms of security exposure were more acute. The identified issue centred on XML external entities where the settings in the code—or rather lack of settings!—leads to the security risk known as XML External Entities (XXE), part of the OWASP Top 10 security threats.[5] This is a form of attack where the XML parser running on the server can be tricked into using components such as DTDs and XSL Stylesheets externally on the client's/hacker's computer, thereby transferring in the process content containing potentially sensitive system data.

The solution is to either enable secure processing or else disable use of external resources when parsing XML in the code. Rather than rushing to create a Transformer, with a single line of code such as *TransformerFactory.newInstance().newTransformer()*, this needs be broken down into smaller steps so settings in the TransformerFactory can be more carefully stipulated to disable *ACCESS_EXTERNAL_DTD* and *ACCESS_EXTERNAL_STYLESHEET*.

For DSpace the other two kinds of critical vulnerabilities were also from the OSWAP Top 10: Broken Authentication and Security Misconfiguration. The specifics of the former is related to the use of an unauthenticated LDAP connection (under certain conditions) making it susceptible to injection attacks. Further analysis of the code by a member of the development team would be the most expedient way to determine if this is a true vulnerability, but a general rule to eliminate this risk is to rewrite the code to use *simple* SECURITY_AUTHENTICATION rather than *none*.

The Security Misconfiguration error is related to an oversight in a configuration file, where a servlet filter, CocoonDebugFilter, is defined but no corresponding mapping is provided. The remedy to this is straightforward: either a mapping needs to be added to the configuration file to tie a URL pattern to the filter, or else it is determined the filter is no longer in use, and the filter rule is expunged from the configuration file.

5 DYNAMIC APPLICATION SECURITY TESTING

OWASP Zed Attack Proxy (ZAP) was installed on the Kali virtual machine, and an 'attack' against each of the DL servers initiated. Table 1 summarizes the result. In comparison to SAST, it is the Greenstone system this time that shows the higher number of deficiencies. Focusing in on the three High vulnerabilities in Greenstone these were: path traversal, SQL injection, and cross-side scripting

[5]https://www.cloudflare.com/learning/security/threats/owasp-top-10/

DL System	High	Medium	Low	Info
DSpace	0	2	2	0
Greenstone	3	5	5	0

Table 1: Results by OWASP ZAP when run against DSpace and Greenstone

(XSS). Based on this information a trial and error phase of the investigation was entered into, where manually crafted attacks—through, in particular, changes to URL parameters—were attempted to see if they could change the behaviour of the server to operate outside of the intended behaviour. Our attempts to exploit path traversal and SQL injection attacks were unsuccessful; however, we were able to construct a URL that embodied JavaScript syntax for an alert popup window in such a way that it caused the server to execute that JavaScript—the classic signature of an XSS attack. While our alert merely displayed the digit '1' in the alert popup, the actual implications of being able to do such a thing is severe, and the details of how the attack was constructed was immediately passed on to the Greenstone development team, resulting in changes to the code that have removed the security flaw.

For DSpace, the two Medium level vulnerabilities were to do with LDAP. On closer inspection, these were found to be triggered by the same use of an unauthenticated connection, identified earlier on through our SAST analysis. Consequently, the code revision detailed previously should be sufficient to eliminate this risk.

6 DISCUSSION AND CONCLUSION

In summary, in this article we have put two widely used Open Source Digital Library systems, DSpace and Greenstone, under a cyber-security audit. Through port scanning we found, in both systems, certain ports were opened up to external access as a result of the installation procedure, but these ports were not in use by the DL without further changes to how the system is configured. From a cyber-security perspective, having such ports open is an unnecessary risk. Greater care is needed over the default settings these DL systems provide, particularly concerning the defaults used by the packages the DL system relies upon itself. In the case of AJP13 being active in Tomcat, for instance, we traced the source of this to the fact that it is specified to be on in the files supplied by the Apache project itself.[6] The dangers in not conducting a review of what these supporting packages are doing in terms of their impact on delivering a public facing web resource is a useful lesson to learn for all designers of DL systems.

Reflecting further on this, we note that both systems installation instructions are geared towards setting up operation of the DL over HTTP rather than HTTPS, with the complexity of setting up the latter covered by supplemental material. There are ample examples, however, of human nature at work in the IT domain that show that once something is working, the desire to make further changes intended to enhance or improve on what has been set up is greatly diminished. In the context of the work presented here, we suggest one way to curtail this phenomenon is to provide two different downloads: one for trial purposes in-line with what these projects currently offer, and one that has been hardened [1] for deployment in a production environment, requiring the entry of signed certificates as part of the process, for example. Or, as an alternative, retain a single installation process that operates over HTTP, but configure it so it can only be access from the computer where it is installed (i.e., localhost). To make it public facing, the person responsible for the installation is directed to connect it through an already hardened web server, such as Nginx. In the case of the DL using a Tomcat server, doing this through the activation of the AJP mechanism would be a natural choice.

Our SAST and DAST reviews resulted in identified areas where code and/or configuration files need changing. The Transform changes have already been made in the Greenstone code base, and details of our findings have been sent to the DSpace team for their consideration. It is our prediction that repeating the same set of analyses detailed here against other Open Source Digital Library systems will also be beneficial.

Finally, we acknowledge that our investigations to date have been preliminary in nature. In future work we plan to investigate the security implications resulting from the use of file upload, which is a feature commonly active in a DL system that is being used, for instance, as an Institutional Repository.

[6]In fact, subsequent to the analysis work we undertook, a major security flaw, dubbed Ghostcat, concerning AJP was reported, affecting all versions of Tomcat back to version 6. Disable AJP immediately, and then upgrade if needed was the advice from the Tomcat developers.

REFERENCES

[1] Julia Allen. 2001. CERT System and Network Security Practices. In *NCISSE 2001: 5th National Colloquium for Information Systems Security Education*. Fairfax, VA.

[2] H. Anwar and M. A. Shibli. 2012. Attribute based access control in DSpace. In *7th International Conference on Computing and Convergence Technology (ICCCT)*. 571–576.

[3] D. Bainbridge and I. H. Witten. 2020. A Renewed Look at Greenstone: Lessons from the Second Decade. *Transactions on Internet Research (TIR)* 16, 1 (January 2020). Special issue - 'Digital Heritage and Related Tools'.

[4] Goutam Biswas and Dibyendu Paul. 2010. An evaluative study on the open source digital library softwares for institutional repository: Special reference to Dspace and Greenstone digital library. *International Journal of Library and Information Science* 2, 1 (2010), 001–010.

[5] George Buchanan, David Bainbridge, Katherine J. Don, and Ian H. Witten. 2005. A New Framework for Building Digital Library Collections *(JCDL '05)*. Association for Computing Machinery, New York, NY, USA, 23–31. https://doi.org/10.1145/1065385.1065392

[6] Issie Lapowsky. 2019. In Latest Facebook Data Exposure, History Repeats Itself. *Wired* (March 2019).

[7] Yuma Makino and Vitaly Klyuev. 2015. Evaluation of web vulnerability scanners. In *8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. 399–402. https://doi.org/10.1109/IDAACS.2015.7340766

[8] Steve Morgan (Ed.). 2019. *2019 Official Annual Cybercrime Report.*

[9] Matt Powell. 2019. 11 Eye Opening Cyber-Security Statistics for 2019. *CPO Magazine* (June 2019).

[10] Sukhwinder Randhawa. 2012. Open source software for creation of digital library: A comparative study of Greenstone Digital Library Software and DSpace. *Indian Journal of Library and Information Science* 6, 3 (2012), 45–52.

[11] MacKenzie Smith, Richard Rodgers, Julie Walker, and Robert Tansley. 2004. DSpace: A Year in the Life of an Open Source Digital Repository System. In *European Conference on Digital Libraries*, Rachel Heery and Liz Lyon (Eds.). Springer, Berlin, Heidelberg, 38–44.

[12] Robert Tansley, MacKenzie Smith, and Julie Harford Walker. 2005. The DSpace open source digital asset management system: challenges and opportunities. In *European Conference on Digital Libraries.* Springer, 242–253.

[13] Shahkar Tramboo, Humma, S M Shafi, and Sumeer Gul. 2012. A Study on the Open Source Digital Library Software's: Special Reference to DSpace, EPrints and Greenstone. *International Journal of Computer Applications* 59, 16 (12 2012), 1–9. https://doi.org/10.5120/9629-4272

[14] Ian H. Witten, Stefan J. Boddie, David Bainbridge, and Rodger J. McNab. 2000. Greenstone: A Comprehensive Open-Source Digital Library Software System *(DL '00)*. Association for Computing Machinery, New York, NY, USA, 113–121.