Check Yourself Before You Wreck Yourself -

A study of how to assess security vulnerabilities of web servers through configuration analysis

INGRID HYLTANDER
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Check Yourself Before You Wreck Yourself-
A study of how to assess security vulnerabilities of web servers through configuration analysis

Ingrid Hyltander
13ihy@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Martin Hell

Examiner: Thomas Johansson

September 23, 2019

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The web server is an essential component of many systems today. It has the possibility to give access to files with sensitive information and it is the backbone that enable a vast amount of applications. This makes it critical to ensure that files are only accessed and altered by intended users and that web servers are always up and running when expected. One important aspect of doing this is to ensure that the configuration of the web server does not cause security vulnerabilities. However, this is not a straightforward task as there are normally hundreds of configurations parameters and different vulnerabilities to take into account. This thesis explores security vulnerabilities related to the configuration of web servers, more specifically the web server software Apache and Nginx, and how to verify absence of security misconfiguration.

The exploration consists of three major segments. First, information sources regarding security misconfiguration of Apache and Nginx are analyzed and compared. The conclusion is that there are beneficial sources but none is covering every configuration needed to avoid security misconfiguration. They could also benefit from using scoring systems to allow users to understand which security misconfigurations are the most critical. Next, tools available today that can help users verify absence of faulty configuration are examined and compared. The conclusion is that there is no tool with ready to use content fully covering every configuration needed to avoid security misconfiguration. Besides, they are, to a varied extent, not satisfactory regarding how they present rationale about and possible consequences from needed configuration and an easy to survey output. This result lead to the exploration if it is possible to use available tools to create a beneficial solution which can verify the presence of all needed configuration and at the same time educate users about why this configuration is needed and neatly present the result of this verification. This resulted in the development of new ready to use content for one of the examined tools called Chef Inspec. The purpose of the new content was to see if it was possible to cover all types of needed configuration and how Chef Inspec performed with this new content. The conclusion is that it is possible to create content covering all needed configuration, but problems arise if a user is running multiple Apache instances on the same machine. The solution is fairly satisfying but there is room for improvement of the output of Chef Inspec to facilitate the users understanding of the rational behind the suggested configuration and the survey of the result from the verification.

Popular Science Summary

Web applications are present in a wide range of areas, not only in business related operations but also in financial, healthcare, defense, and other critical infrastructures. It is of high importance to ensure that web applications are secure and that they do not expose security vulnerabilities that malicious users can take advantage of to create damage.

The web server is an essential component of many web applications. It has the possibility to give access to files with sensitive information and it is a backbone that enable a vast amount of systems. Thus, it is critical to ensure that files are only accessed and altered by intended users and that web servers are always up and running when expected.

One important aspect of doing this is to ensure that the configuration of the web server does not cause security vulnerabilities. However, this is not a straightforward task as there are normally hundreds of configurations parameters and different vulnerabilities to take into account. Besides, research have shown that configuration is today not only performed by professional system administrators but also by pluralistic and novice administrators as a result of open-source software and the on-demand cloud computing infrastructure.

This thesis explores the relationship between configuration of web servers and security. It analyzes what configuration is required to counteract security vulnerabilities of web servers and if, or how, validation to ensure presence of this correct configuration can be performed today. The thesis shows that there are beneficial information sources regarding security misconfiguration of web servers, but none is covering every configuration needed to avoid security misconfiguration. The information sources could benefit from using scoring systems to allow users to understand which security misconfigurations are the most critical. It also demonstrate that no tool was found with ready to use content fully covering every configuration needed to avoid security misconfiguration. Besides, the examined tools are, to a varied extent, not satisfactory regarding how they present rationale behind and possible consequences from needed configuration and an easy to survey output. The thesis suggests that there is one beneficial tool with the possibility to educate users but that the ready to use content for it found was not adequate. New content was written for this tool, showing that it has the possibility to cover almost all types of needed configuration for the web servers Apache and Nginx. However, there is room for improvement of the output and some functions.

Acknowledgement

First and foremost I would like express my gratitude to my supervisor Martin Hell at the Department of Electrical and Information Technology at Lund University for giving me the chance to work on this thesis and for all the feedback and guidance throughout its execution.

I would also like to praise the CTO of debricked AB Oscar Reimer for all of his technical support, valuable insights and feedback.

Lastly, I would like to show appreciation to the front end developer of debricked AB Alexander Cobleigh and my partner Albin Garpetun for keeping my spirit and inspiration high throughout the thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter introduces the problem, specifies the objectives for the thesis, presents previous related research and describe the outline of the report.

1.1 Background

The web server is an essential component in a wide range of systems, covering everything from simple file sharing to business-critical applications. The basic architecture no matter the system is that a client makes HTTP requests to the web server to receive information, i.e. files, and the web server analyze the request and send a HTTP response containing the file, if it exists/can be generated and the user is entitled to access it. If this is not the case, an error message is sent signaling what went wrong. In this way, web servers have the possibility to give clients access to files that can contain sensitive information such as financial or private data and are the backbone that enable a vast amount of network connected systems today. Thus, it is of high importance to ensure that files are only accessed and altered by the intended users and that web servers is always up and running when expected. A critical activity to achieve this is ensuring that the configuration of the web server is correct and not creating any vulnerabilities that malicious attackers can exploit. However, this is not a straight forward task as a web server usually have hundreds of configuration parameters and there are many different types of vulnerabilities that one needs to take into account. It seems as this is indeed a problem within the industry as security misconfiguration is ranked as number six of the ten most critical web application security risks in the OWASP Top Ten, which is described further in Section 2.2. The goal of this thesis is to investigate this problem further and analyze what configuration is required to counteract vulnerabilities of web servers and if or how validation to ensure presence of this correct configuration is performed today.

1.2 Problem description and research objectives

The thesis will examine the relationship between between vulnerabilities and the configuration of web servers and investigate questions such as

- What vulnerabilities related to configuration of web servers exist and what sources of information exists to learn about this? How does these sources compare to each other?
- What tools exist today to access and evaluate vulnerabilities related to the configuration of web servers?
- How are vulnerabilities related to configuration of web servers presented to the user? Are the different vulnerabilities ranked? Is the user educated about the vulnerabilities?

1.3 Scope

The thesis will only focus on configuration related to vulnerabilities and not configuration that can lead to system misbehavior such as crashes, hangs, indeterminate failures, etc. To distinguish between configuration issues related to security vulnerabilities and configuration issues related to system misbehavior the term "security misconfiguration" will be used to describe the former and the term "behaviour misconfiguration" will be used to describe the latter. Security misconfiguration is a term that is used by the Open Web Application Security Project (OWASP), which is further described in Section 2.2, while behaviour misconfiguration is a term which is formulated for this thesis.

The thesis will only focus on the configuration of web servers, more specifically, their directories, files, the content of these and their permissions. Only tools that have the possibility to evaluate this in detail will be examined. If an information source or a tool is found with content for other software and not just web servers, e.g. that a tool have tests available for several types of databases, this will be mentioned but the content will not be examined.

1.4 Related work

The focus of paper [197] is not specifically on security misconfiguration but rather on behaviour misconfiguration, however, the results are still relevant for this thesis as it confirms the need for support when configuring systems. The authors conclude that systems are in general today, and that this trend will continue in the next decades, more complex due to their large number of configuration parameters and the correlation and dependency between these parameters, both inside the software programs and across different software components. They also state that as a result open-source software and the on-demand cloud computing infrastructure, configuration is today not only performed by professional system administrators but also by pluralistic and novice administrators who operate their own systems. Another problem is that many software projects have inexplicit or even hidden configuration constraints that are neither documented in user manuals nor informed via system reactions. This invisibility diminish the possibility for users to understand the configurations and determine the settings. The authors also cover approaches and tools to discover configuration errors, though only

mentioning their names and not examining them in depth as tools will be in this thesis.

The authors of [117] contributed with 78 generic security best practices, weighted in importance, that users can use to manually assess any web server engine. When evaluating the best practise documents they noticed that servers with the same web server vendor and same version had very different result and that the running operating systems, the configuration of the web server, and the conformity with basic security policies is of importance. They also found that the conclusion of how secure a web server is is not solely dependent on the amount of passed test, but rather how important the passed/failed tests are. Thus, this paper does not examine the differences between the sources and does not evaluate configuration tools. However, the sources of information used to derive the security best practice is used and evaluated in this thesis.

The authors of [62] contributed with the open source tool called SCAAMP which audit, fix and rate the security of the configuration of Apache, MySQL and PHP to cover the "lack of freely available tool that assists developers and administrators to automatically audit, fix and rate security configuration risks for web server environments on which web applications are tested (during development) and then deployed" [62]. When testing the tool they found that the default security configuration offered by eleven real-life web application development and deployment environments were not in conformity with recommended security configuration settings and that it existed observable variations among the eleven packages while the security differences between OS platforms and between manual and pre-packaged server environments was not highly significant. The results confirms the need for support when configuring systems but unfortunately it does not mention what sources of information was used as a base for defining configuration vulnerabilities. The tools mentioned will be further examined in this thesis.

Paper [22] covers why diagnosing security misconfigurations is difficult, prevalent configuration and its structure and approaches for detecting misconfiguration. The authors develop a tool which is functional across heterogeneous entities such as applications, systems and the cloud with a declarative rule specification to make validation checks easy to encode, comprehend, extend and maintain. They compare it with other security misconfiguration detection tools over the difference in time required to run the tools and how many lines of code it takes to declare a rule. The sources of information are not compared, and the tools mentioned will be further examined in this thesis.

In [21] the author focus on the need for automatic solutions to aid the management of large-scale deployments of disparate computing devices over evolving dynamic networks. He highlights that within security issues there is untapped potential of using automatic computing. He develops a tool that translate OVAL (see more in Section 2.4.1) to CfEnginge policies (see more in Section 5.2.2), to enable devices to assess and reconfigure themselves without the need of an administrator. The report is relevant as it covers both OVAL and CfEngine and highlight the benefits of automatic solutions to manage security misconfiguration. It also mentions other tools that will be further examined in this thesis.

The security of container images is examined in [187] and evaluation of configuration files is part of this procedure. To test security misconfiguration, they

specify so called compliance rules derived from relevant sources. When the article was published the rules where mostly targeting best practices for Linux-class operating system but the authors states that "we plan to expand our compliance rules to cover not only the OS checking rules, but also application-specific rules, such as for MySQL, ngnix, etc" [187]. The authors reach the conclusion that 99% of the public images contain 5 compliance rule violations and that the average compliance rule violation is barely related to how popular the container images is, but seems to vary when grouping the container images by distributions. The research of the paper is relevant to security misconfiguration, but does unfortunately not yet include relevant sources or tools related to web servers.

Another tool called EnCore is presented and evaluated in [198] which automatically detect behaviour misconfigurations while taking into account the interaction between the configuration settings and the executing environment and the correlations between configuration entries. The tool learns configuration rules from a given set of sample configurations and does thus not require manually specified rules. The research is relevant as, although it is not the primary focus, it is able to find security misconfigurational relevant information. In addition, the tool take advantage of machine learning and not manual labour to define rules and also take environment and correlation into account which is an approach that differs from the research previously mentioned. In addition, numerous tools are mentioned which are examined in this thesis.

1.5 Outline

The this thesis is structured as follows:

- *Chapter 1 Introduction:* Introduces the problem, specifies the objectives for the thesis, presents previous related research and describe the outline of the report
- *Chapter 2 Preliminaries:* Introduces relevant background knowledge
- *Chapter 3 Method:* Describes how the research was conducted and the rationale behind decisions made throughout the process
- *Chapter 4 Analysis of Information Sources:* Presents the result of the research regarding sources of information
- *Chapter 5 Analysis of Tools:* Presents the result of the research regarding tools
- *Chapter 6 Implementation:* Presents the result of the implementation of content for the tool Chef Inspec
- *Chapter 7 Discussion:* Discuss the results of the previous chapters
- *Chapter 8 Conclusion and Future Research:* Summarize the thesis and suggest relevant related research

Chapter 2
Preliminaries

This chapter provides a brief summary of relevant web server software, organizations, frameworks and concepts.

2.1 Web Servers

Web servers have a so called client-server structure and run over a networked environment, e.g. the Internet [117]. Clients connect to a web server and makes a request by Uniform Resource Locators (URLs), which specify what communication protocol to use, e.g. http, the server name, e.g. www.apache.org, a URL-path, e.g. /docs/current/getting-started.html and possibly a query string, e.g ?arg=value, for passing additional arguments to the server. The URL-path represents the content that the client is requesting. The server sends a response to the client containing a status code, e.g. an indication whether the request was successful or not, and possibly a response body [74]. Figure 2.1 illustrates the place and function of a web server in a general system architecture. There are numerous web server software and Apache HTTP and Nginx are the most used in terms of open source versions [115].

2.1.1 Apache HTTP

The Apache HTTP server, also called httpd, was launched in 1995. The current version is 2.4 and previous versions such as 2.2 are no longer supported [68]. Apache HTTP is configured by placing directives in plain text configuration files and the main configuration file commonly called httpd.conf. Other configuration files may be added using the Include directive. Changes to the main configuration files are only recognized when it is started or restarted. The configuration files contain one directive per line and arguments to them are separated by a whitespace. Directives in the configuration files are case-insensitive, but arguments to directives are often case sensitive. If a line begin with a hash character # it is considered a comment and is ignored. Comments may not be included on the same line as a configuration directive.

Apache HTTP is a modular server and only includes the most basic functionality in the core server. Users can add extended features through so called modules. By default, a base set of modules is included in the server at compile-time. The

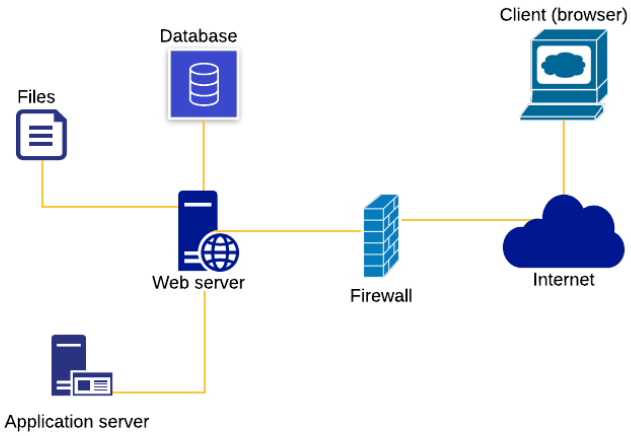

Figure 2.1: The place of a web server in a general system architecture

server can be compiled to use dynamically loaded modules and then modules can be compiled separately and added at any time by using the LoadModule directive. Otherwise, it must be recompiled to add or remove modules.

Directives placed in the main configuration files apply to the entire server but users can configure only a part of the server by scoping directives by placing them in <Directory>, <DirectoryMatch>, <Files>, <FilesMatch>, <Location>, and <LocationMatch> sections. These sections limit the application of the directives which they enclose to particular filesystem locations or URLs. They can also be nested, thus allowing fine grained configuration. The Apache HTTP also has the capability to serve many different websites simultaneously by using so called Virtual Hosting and directives can also be scoped by placing them inside <VirtualHost> sections, so that they will only apply to requests for a particular website.

Users can also use decentralized management of configuration through special files placed inside the web tree. They are usually called .htaccess, but any name can be specified in the AccessFileName directive. Directives placed in .htaccess files apply to the directory where the file is located, and all the sub-directories. The .htaccess files follow the same syntax as the main configuration files. Changes made in the .htaccess files take immediate effect as they are read on every request [73]. For more detailed information about Apache HTTP and its configuration, visit [73].

2.1.2 Nginx

Nginx, also stylized as NGINX and nginx, was released in 2004 and was originally developed to solve the difficulty with handling large numbers of concurrent connections and to "create the fastest web server around" [106]. The current version is

1.16 [105]. Nginx have one master process and several worker processes, where the main purpose of the master process is to read and evaluate configuration and maintain worker processes while worker processes do the actual processing of requests. The number of worker processes is defined in the configuration file and can be fixed or automatically adjusted to the number of available CPU cores. The way that Nginx works is determined in the configuration file. By default, the configuration file is named nginx.conf and placed in the directory `/usr/local/nginx/conf`, `/etc/nginx`, or `/usr/local/etc/nginx` [102]. Other configuration files can be added by using the include directive in the main nginx.conf file to reference the contents of the other configuration files. Changes made in the configuration file will not be applied until the command to reload configuration is sent to nginx or it is restarted [104].

Nginx consists of modules which are controlled by the configuration options called directives. The directives are specified in the configuration file and are divided into simple directives and block directives. A simple directive consists of the name and parameters separated by spaces and ends with a semicolon (;). A block directive has the same structure as a simple directive, but ends with a set of additional instructions surrounded by braces (and) instead of a semicolon. If a block directive can have other directives inside braces, it is called a context. Directives located outside of any contexts are considered to be in the main context. Lines that start with a hash character # is considered a comment and are ignored [102]. For more detailed information about Nginx and its configuration, see [102].

2.2 Open Web Application Security Project (OWASP)

OWASP is an international and non-profit organization established in 2001 with an open community "dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted" [133]. There are more than 45,000 people that voluntarily participate in the project [144]. OWASP produces different types of materials such as tools, documents, forums, and chapters which are all free to access [133] and their work is recommended/mentioned by numerous governmental institutions, e.g. the Canadian Cyber Incident Response Centre and the American National Institute of Standards and Technology (NIST), and international organizations, e.g. Institute of Electrical and Electronics Engineers (IEEE) and the European Union Regulations [142]. One of their most popular projects is the OWASP Top Ten which describe and rank the most critical security risks to web applications and is continuously updated (released so far year 2004, 2007, 2010, 2013 and 2017) [141].

2.3 The Center for Internet Security, Inc. (CIS)

The Center for Internet Security, Inc. (CIS) is a non-profit entity that through an international community provide various resources such as security best practises and tools to help private and public organizations safeguard against cyber threats [27]. There are more than 200 employees [63] but the development of their resources also involves volunteer participants outside the organization. They

state on their website that "The CIS Controls and CIS Benchmarks are the global standard and recognized best practices for securing IT systems and data against the most pervasive attacks. These proven guidelines are continuously refined and verified by a volunteer, global community of experienced IT professionals" [27]. There is no reference to governmental or international organizations citations on their website, but their best practice documents called CIS Benchmarks was downloaded more than a million times during 2018, more than 300 of the members of their SecureSuite membership are a governmental institutions [64] and as seen in Chapter 3 their work is also used in the academia.

2.4 National Institute of Standards and Technology (NIST)

NIST is an American governmental institute with the vision to be "the world's leader in creating critical measurement solutions and promoting equitable standards" [126]. NIST work in many areas, e.g. Bioscience, Energy and Cybersecurity [128]. They maintain numerous security related resources, including the widely used National Vulnerability Database (NVD) which is a database with information of software security vulnerabilities [125]. They also maintain the SCAP suite, further described below.

2.4.1 Security Content Automation Protocol (SCAP)

SCAP is a suite of specifications and is developed for expressing, exchanging, and processing security automation content. SCAP content describes what endpoint posture information to collect and how to compare this information against a desired state using implementation-neutral, standardized formats. SCAP tools use these formats to drive the collection and evaluation of posture information to determine if software vulnerabilities and misconfigurations exist on a specific managed endpoint [195]. The same SCAP content can be used by multiple tools to perform a given assessment described by the content [127]. SCAP was created as a NIST program in 2006, and the first SCAP v1 specification, SCAP 1.0, was published in 2009. The current revision, SCAP 1.3, was released in February 2018. A new version, called Security Content Automation Protocol (SCAP) Version 2, is under development and is described as "the next major revision of the Security Content Automation Protocol (SCAP)" [195]. A white paper was released in September 2018 which presents the design goals, set of issues to address from the previous versions of SCAP, a draft SCAP v2 architecture, and outlines an iterative development process for SCAP v2 [195].

The article "Understanding SCAP Through a Simple Use Case" [60] was published 2016 and it provides a quite comprehensive description of the elements of SCAP. In the article, the SCAP specification is divided into three (disjoint and complementary) categories:

1. Languages for checklists and tests specification and reporting,
2. Enumerations (dictionaries) of security information,
3. Vulnerability measurement and scoring systems.

The authors depict that the languages are aimed to give a meaning to the whole information in the context of the systems to be checked, the enumerations contain the information about platforms, configurations and vulnerabilities and the measurement and scoring systems allow to give a ranking to the results of the tests and to prioritize remediation. They then describe that the base components of SCAP are:

- **XCCDF**: Extensible Configuration Checklist Description Format is a language for writing security checklists/benchmarks and report the results of their evaluation.
- **OVAL**: Open Vulnerability and Assessment Language is a language for representing system configuration information, assessing machine state, and reporting the results.
- **CPE**: Common Platform Enumeration is a nomenclature and dictionary of hardware, operating systems, and applications.
- **CCE**: Common Configuration Enumeration is a nomenclature and dictionary of software security configurations.
- **CVE**: Common Vulnerability and Exposures is a nomenclature and dictionary of software security flaws.
- **CVSS**: Common Vulnerability Scoring System is a system for measuring the severity of software vulnerabilities.

The authors also described the basic workflow of SCAP as "a tool takes an XML file as input written in XCCDF with references to CPE, CCE, CVE and CVSS items and with eventually links to OVAL definitions. It then outputs an XML file from which an HTML report or guide can be generated " [60]. The flow is illustrated in Figure 2.2:

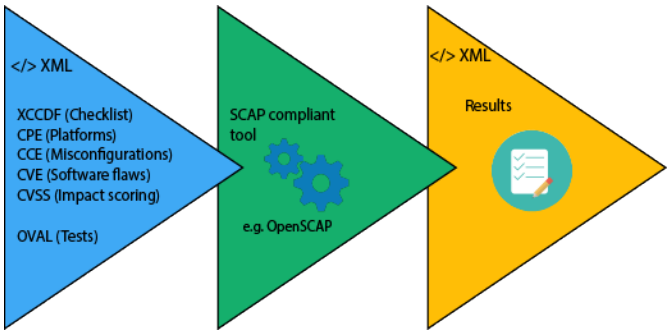

Figure 2.2: Basic SCAP workflow

OVAL, XCCDF and CCE are described more in detail below.

2.4.1.1 Open Vulnerability and Assessment Language (OVAL)

OVAL is a language based on XML and is defined by XML Schemas. It involves three different categories:

- **Collecting Information from Systems** through the OVAL System Characteristics schema which defines a standard XML format to represent system configuration information, e.g. operating system parameters, installed software application settings, and other security relevant configuration values. The schema provides system characteristics which OVAL Definitions can be compared to in order to analyze a system for the presence of a particular machine state [50].
- **Standardized Tests** through the OVAL Definition schema which is the language framework for writing OVAL Definitions in XML. OVAL Definitions encode the details of a specific machine state (when is a system vulnerable, in compliance, etc.) which enable testing of a system to be automated [50]. An OVAL Definition consists of multiple tests referring to objects (i.e. a file name, a registry key) and states (i.e. file's md5 hash, registry key's value). A test will pass when a resource denoted by given object satisfies requirements in a corresponding state [137].
- **Results of the Tests** through the OVAL Results schema which defines a standard XML format for reporting the results of the evaluation of a system. The results data contains the current state of a system's configuration as compared against a set of OVAL Definitions [50].

An open source tool based on SCAP which is further described in Section 5.1.1 summarize OVAL as a "declarative language for making logical assertions about the state of endpoint system" [137]. Figure 2.3 shows an example of an OVAL Definition for Apache HTTP.

2.4.1.2 Extensible Configuration Checklist Description Format (XCCDF)

XCCDF is a specification language for writing security checklists which together form a benchmark for a given system. Each checklist consists of a set of rules logically grouped. The XCCDF syntax is based on XML. A XCCDF rule is a high-level definition which will be translated to a check on the related system. The rules are not specified directly inside the XML file, instead they point to OVAL Definition files [60].

2.4.1.3 Common Configuration Enumeration (CCE)

The CCE is identifiers for security configuration issues and exposures. It gives each particular security-related configuration issue a unique identifier and can correlate configuration data across multiple information sources [113]. The CCE has not been updated at all since 2013, only outdated versions of Apache is supported and Nginx is not supported at all [132].

```
<definition xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5" class="patch" id="
oval:org.mitre.oval:def:20246" version="9">
  <metadata>
    <title>DSA-2202-1 apache2 - failure to drop root privileges</title>
    <affected family="unix">
      <platform>Debian GNU/Linux 6.0</platform>
      <platform>Debian GNU/kFreeBSD 6.0</platform>
      <product>apache2</product>
    </affected>
    <reference ref_id="DSA-2202-1" ref_url="http://www.debian.org/security/dsa-2202-1" source="VENDOR" />
    <reference ref_id="CVE-2011-1176" ref_url="https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-1176"
    source="CVE" />
    <description>MPM_ITK is an alternative Multi-Processing Module for Apache HTTPD that is included in Debian's
    apache2 package.</description>
    <oval_repository>
      <dates>
        <submitted date="2013-12-06T10:22:47">
          <contributor organization="ALTX-SOFT">Sergey Artykhov</contributor>
        </submitted>
        <status_change date="2013-12-06T14:48:58.855-05:00">DRAFT</status_change>
        <status_change date="2013-12-23T04:01:41.117-05:00">INTERIM</status_change>
        <status_change date="2014-01-13T04:01:21.089-05:00">ACCEPTED</status_change>
        <modified comment="EDITED oval:org.mitre.oval:def:20246 - Debian Patch Update" date="
        2014-06-06T17:24:00.692-04:00">
          <contributor organization="ALTX-SOFT">Sergey Artykhov</contributor>
        </modified>
        <status_change date="2014-06-06T17:26:40.986-04:00">INTERIM</status_change>
        <status_change date="2014-06-23T04:07:05.791-04:00">ACCEPTED</status_change>
      </dates>
      <status>ACCEPTED</status>
      <min_schema_version>5.10</min_schema_version>
    </oval_repository>
  </metadata>
  <criteria>
    <extend_definition comment="Debian 6.0 is installed" definition_ref="oval:org.mitre.oval:def:12959" />
    <criteria comment="GNU/Linux or GNU/kFreeBSD kernel" operator="OR">
      <extend_definition comment="Debian GNU/Linux is installed" definition_ref="oval:org.mitre.oval:def:24894" />
      <extend_definition comment="Debian GNU/kFreeBSD is installed" definition_ref="oval:org.mitre.oval:def:24698"
       />
    </criteria>
    <criterion comment="apache2 DPKG is earlier than 0:2.2.16-6+squeeze1" test_ref="oval:org.mitre.oval:tst:89228"
     />
  </criteria>
</definition>
```

Figure 2.3: OVAL Definition for Apache HTTP

Chapter 3
Method

This chapter describes how the research was conducted and the rationale behind decisions made throughout the process.

3.1 Literature Study

When the thesis was initiated, the first objective was to understand the state of security misconfiguration of web servers and see to what extent the subject was covered within the academia and if there was any problems related to security misconfiguration. Thus, a great amount of time was spent on examining previous research and articles related to the area. The decision was made to focus on security misconfiguration of the Apache HTTP server and the Nginx server as these are the two most popular open source web servers [114] [194] [116], thus accessible by everyone and widely used. Quite a lot of research was found within the area, however, most of it covered new tools or suggestions of how to handle configuration and not detailed examination of the pros and cons of existing tools and approaches. In addition, a lot of research focus on behaviour misconfiguration and not security misconfiguration. Modest information or comparison between sources of information of security misconfiguration was found.

3.2 Information Sources

To be able to examine the tools and approaches available, their strength and weaknesses and understanding if they provided any valuable validation or information regarding security misconfiguration, it was decided to continue the thesis with focus on sources of information of security misconfiguration. This was done to gain more knowledge about problems and counter measurements related to secure configuration of web servers, but also to explore if there was any good recourse to use as a baseline to evaluate against when examining the tools and approaches found. The sources of information found through the primary literature study was first examined. However, the examination of sources was an iterative process as some of the sources and tools examined referred to not previously found sources of information and as new papers and studies was read as the thesis advanced. The sources of information found was:

- CIS Benchmarks (mentioned in [113] [129] [117] [60] [22] [145] [131])
- OWASP Testing Guide (mentioned in [193] [23] [25] [22] [85])
- Guide to General Server Security (mentioned in [193])
- Common Criteria (mentioned in [180] [129])
- Orange Book (mentioned in [129]
- HIPAA (mentioned in [22])
- PCI DSSS standards (mentioned in [22] [134])
- OSSG guidelines [22]
- DISA STIG (mentioned in [22]) [134] [129]
- FEDRAMP (mentioned in [22])
- Federal Information Security Management Act (FISMA) (mentioned in [22])
- United States Government Configuration Baseline (USGCB) (mentioned in [22] [134])
- The National Information Assurance Partnership (NIAP) (mentioned in [22])
- SANS (mentioned in [51])
- National Checklist Program Repository (mentioned in [24])

The sources of information with many references to them found and/or with in-depth security misconfiguration of web server related content, i.e. with actual suggestion of values and not general recommendations such as "verify that the proper encryption strength is implemented for the encryption methodology in use" [52], was chosen to be examined further. These were the CIS and OWASP documents and the DISA STIGs. The CIS Benchmarks (see more in Section 4.1) for Apache and Nginx were favourable baselines for comparing other sources of information and the ready to use content of the tools against as they:

- Are easy accessible as they are provided as PDFs unlike the DISA STIGs which need a special tool for proper viewing or the OWASP documents which have quite dispersed configuration recommendations, requiring to manually go through pages to find them.
- Cover both Apache and Nginx, unlike the DISA STIGs which only cover Apache and the OWASP documents that mostly cover general cases or giving example recommendations only for Apache.
- Have large number of relevant recommendations. The CIS Benchmark have more than 80 recommendations all relevant for directly auditing the configuration of Apache and more than 50 for Nginx while the DISA STIGs have many duplicates and recommendations involving interviewing personnel and the OWASP recommendations are all not relevant for web servers or not specifying how to perform the audit in detail.

- Had the highest amount of references towards it out of the papers and documents examined in the literature study.

When the sources of information chosen to be examined further was examined, the main questions was

- How is the information structured?
- Does it educate the user about the reasons behind recommendations/concerns mentioned?
- Are software specific remediation given or are they general?
- Is there any ranking of recommendations/concerns?
- When was it last updated?
- Who are the author(s)?
- Does it cover information that the CIS Benchmarks for Apache and Nginx does not cover?

The result of this analysis in found in Chapter 4.

3.3 Tools

A variety of tools and approaches was found through studying literature and searching online for tools related to the subject, although many primarily targeted to behaviour misconfiguration. However, these tools were not immediately rejected as they might be adapted for security misconfiguration purposes. Tools found that were not part of the previously mentioned scope were rejected, e.g. online scanners or tools only specified towards Java-objects. It was also decided to reject tools that did not support common open source operating systems such as Ubuntu, Debian, Red Hat and SUSE as this imply that the tool is not suitable for the general public. The tools and approaches found that were not rejected was:

- CIS-KAT (mentioned in [22] [145])
- Nessus (mentioned in [62] [180] [21] [113])
- OpenSCAP (mentioned in [22] [60] [187])
- CFEngine (mentioned in [197] [21]) [196]
- Chef (mentioned in [21] [197])
- Chef Inspec (mentioned in [22])
- Puppet (mentioned in [22] [21] [197])
- Ansible (mentioned in [22])
- GovReady (mentioned in [22])
- LCFG (mentioned in [197] [196])

- NewFig (mentioned in [197])
- Confaid (mentioned in [197])
- SCAAMP (mentioned in [62])
- Ovalyzer (mentioned in [21])
- Ferret (mentioned in [180])
- Cops (mentioned in [180])
- Spex (mentioned in [198])
- EnCore (mentioned in [198])
- CODE (mentioned in [198])
- ConfigRE (mentioned in [196])
- Glean (mentioned in [196])
- Validation (mentioned in [196])
- Chronus (mentioned in [196])
- Joval

CIS-KAT and Nessus are both proprietary solutions but they do have scaled down free versions. Unfortunately, the free versions did not cover web servers (CIS-KAT) or support configuration checks (Nessus). Contact was made to the companies behind these tools to see if it was possible to obtain the tools with the required functions. Unfortunately this was unsuccessful and these tools are not evaluated. A lot of the tools or approaches was only to be found in papers and with no possibility to access or evaluate them. This was the case for NewFig, Ovalyzer, Ferret, COPS, Spex, ECnCore, CODE, ConfigRE, Glean, Validation, Chronus, and Confaid. LCFH only supports Red Hat Enterprise Linux and Scientific Linux and will therefore not be examined. SCAAMP only evaluates Apache, has not been updated since 2013 and is stated to be in beta mode. GovReady is a bash wrapper around OpenSCAP and is stated to not currently being actively maintained. As OpenSCAP is still maintained and like GovReady require SCAP content it was chosen to be examined instead. OpenSCAP was also chosen to be examined instead of Joval as OpenSCAP is open source and Joval is not, and they use the same kind of SCAP content for evaluation. Chef Inspec is the component from the suite of Chef software with focus on security and compliance and therefore this was the chosen software of the Chef software suite to be examined. Thus, the tools further examined in the thesis are OpenSCAP, CFEnginge, Chef Inspec, Puppet and Ansible. These tools were all examined using Ubuntu 18.04 as operating system.

When the tools were examined, the main questions were:

- What operating systems are supported?
- Under what licence is it released?
- Does it need to be installed onto the device it examines?

- Is it possible to evaluate both Apache and Nginx? Is there ready to use content to evaluate them? Are other types of software supported?
- Does it have the possibility to remediate misconfiguration found?
- Does it work as intended or are there problems?
- Does it offer evaluation and/or configuration that correspond to secure best practice? How does it compare to the CIS Benchmarks?
- Is there any type of scoring or ranking of the configuration parameters?
- Is the user educated about why a certain configuration is suggested?

Different tools offered different extent of ready to use content, as seen in Chapter 5. If there was a multitude of ready to use content, it was decided to examine the one with most downloads and the one which resulted in the most extensive secure configuration by default, i.e. not requiring the user to have knowledge about security misconfiguration and specify his secure configuration by himself. This was decided as it could give interesting insights in if popular ready to use content is secure and if there are ready to use content that correspond to available best practice. Ready to use content that did not only focus on Apache or Nginx but also other software such as run-time environment was disregarded. The result of the analysis is found in Chapter 5.

3.4 Exploring a Possible Solution

As seen in Chapter 5, the examination of the tools and their content showed that the majority hardly had any ready to use content corresponding to available best practice. There were cases with quite extensive content to improve the security, such as at Ansible Galaxy, although nothing was found that had a complete coverage of all needed configuration that had been found in the different information sources and ready to use content. Another aspect to take into account, as stated in [22], is that verification and deployment of configuration are two distinct operations and by combining them, assertions and rationale behind configuration is not cleanly specified and mixed with the deployment code. This is the case for both Puppet and Ansible, as seen in Section 5.4 and Section 5.5. Each configuration decision made by these tools is not explicitly stated and instead a high-level description such as "Configure Apache" and the result of this action, i.e. if everything was corresponding to the endorsed configuration or if a remediation was needed, is presented. This does not create a solution which makes it easy for a user of the tool to understand and learn more about why or why not a certain configuration should be made and if this is the right option for his use case. Besides, a user might be tempted to just run the deployment code without understanding what is really affected and possibly create a suboptimal configuration.

In conclusion, no solution was found that satisfied full coverage of necessary configuration to fully protect web servers and that enabled users to understand why that configuration should be made and possible consequences that he should be aware of. This generated the question whether it is possible or not to create such a solution with the available examined tools. Chef Inspec was the only examined tool

specifically developed for verification of configuration and thus avoiding previously mentioned problems with mixing verification and deployment. It has a design that allows ranking, rational and remediation to be clearly specified, it is still being maintained, it has an community that share ready to use content and its design makes it quite straightforward for user to create their own content. Thus, it seemed most beneficial to choose Chef Inspec as the tool for exploring if verification of fully coverage of needed configuration and education about this configuration was possible.

To verify if Chef Inspec truly was a feasible solution for verifying of absence of security misconfiguration it was decided to create new content for Chef Inspec. The recommendations of the CIS Benchmarks and the missing ones found in other sources of information and ready to use content of the tools was divided into categories depending on what type of configuration it was. Then, at least one verification of a recommendation belonging to each category was tried to be implemented. In a few cases, such as the Nginx category "Verify Existence of Directive Inside a Context and Verify Its Value", two tests were implemented to verify that both quite straight forward and more complex recommendations audits were possible. The implementations were tested thoroughly, verifying that correct configuration and missing or erroneous configuration was detected accordingly according to each written control, including situations where the same type of configuration was enabled in different contexts or in different configuration files. The result of the implementation and constraints found is found in Chapter 6.

Chapter 4
Analysis of Information Sources

This chapter describes the results of the analysis of information sources related to security misconfiguration.

4.1 CIS Benchmarks

The CIS Benchmarks are best practices to enable secure configuration. CIS states on their website that "CIS Benchmarks are the only consensus-based, best-practice security configuration guides both developed and accepted by government, business, industry, and academia " [28]. They are continuously developed and refined through volunteer efforts of subject matter experts, technology vendors, public and private community members and the CIS Benchmark Development team. There are benchmarks available for Operating Systems, Server Software, Cloud Providers, Mobile Devices, Network Devices, Desktop Software and Multi Function Print Devices [28]. The benchmarks are free to access and can be found at [29]. There are benchmarks relevant for this thesis:

- Apache HTTP have the CIS Apache HTTP Server 2.4 Benchmark v1.4.0 which was published 13/07/2018, aimed towards Apache Web Server versions 2.4 running on Linux.
- Nginx have the CIS NGINX Benchmark v1.0.0 which was published 28/02/2019, aimed towards NGINX version 1.14.0 running on Linux.

The CIS Apache HTTP Server 2.4 Benchmark v1.4.0 will be called CIS Apache and the CIS NGINX Benchmark v1.0.0 will be called CIS NGINX through the rest of the thesis. In general, subchapters of the benchmarks will be mentioned with their number and not name, e.g. CIS Apache 3.5.

It is interesting to notice that the number of authors of or contributors to CIS Apache are 18, where 8 are security related organizations and the rest are individuals without a title or association to any company stated, while CIS NGINX only have 3, all of them individuals without a title or association to any company stated. The are about 50 recommendations in CIS NGINX while CIS Apache has more than 80 recommendations. This might be due to that Apache HTTP is older and is more widely used.

The CIS Benchmarks are granular, with specific audits and remediation recommendations depending on what software the benchmark is developed for. There

are numerous recommendations for each benchmark, e.g. the CIS Apache have more than 70 recommendation which are divided into 12 different categories such as "Minimize Apache Modules" and "Principles, Permissions, and Ownership". The recommendations usually follow the syntax:

- Title (Scoring status)
- Configuration profile
- Description
- Rationale
- Audit
- Remediation
- Default Value
- References
- CIS Controls

The **scoring status** can be either "Scored" or "Not Scored", where the failure or compliance to comply with "Scored" recommendations will decrease or increase the final benchmark score and the failure or compliance to comply with "Not Scored" recommendations will not decrease/increase the final benchmark score. There are no clear definition in the benchmarks or homepage of CIS of how the benchmark score is calculated, but there is a checklist of all the recommendation in the end of the benchmarks with the scoring status included. In a forum discussion from 2005 it is stated that "CIS Benchmark Score details show, for each of the security settings required by the benchmark, whether your computer meets that requirement" [59]. Thus, a possible description could be that recommendation with the status "Not Scored" are not as important to have set correctly compared with the recommendations with the status "Scored".

Configuration profile is either "Level 1" or "Level 2", where the Level 1 profile is intended to be practical and prudent, provide a clear security benefit and not inhibit the utility of the technology beyond acceptable means. The Level 2 profile extends Level 1 and items that have this profile can be intended for environments/use cases where security is paramount, acts as defense in depth measure or may negatively inhibit the utility/performance of the technology. It is also possible to have subcategories related to the two profiles, e.g. CIS NGINX have the profiles "Level 1 - Webserver", "Level 1 - Proxy", "Level 2 - Webserver", "Level 2 - Proxy" and "Level 2 - Loadbalancer".

Description describe what the recommendation say and potential exceptions for the recommendation, e.g. the recommendation in CIS Apache named "3.5 Set Group Id on Apache Directories and Files (Scored)" with the description "The Apache directories and files should be set to have a group Id of root, (or a root equivalent) group. This applies to all of the Apache software directories and files installed. The only expected exception is that the Apache web document root ($APACHE_PREFIX/htdocs) is likely to need a designated group to allow web content to be updated (such as webupdate) through a change management process."

Rational describe and educate the reader about the reason behind the recommendation. Some recommendation have quite extensive rational, such as CIS Apache 6.4 while others only have a sentence, e.g. CIS Apache 3.5.

Audit describe how a user can examine if the recommended setting is configured.

Remediation describe how a user can configure the recommended setting.

Default Value describe what the default value of the setting is.

References refer to external sources of information relevant for the recommendation, e.g. documentation from the organization behind the software.

CIS Controls are reference to relevant parts of the CIS Controls. The CIS Controls are a prioritized set of actions that form a set of best practices that mitigate the most common attacks against systems and networks. The document contains controls and their sub-controls which are high-level descriptions of what to do and they require the readers to further educate themselves and find out how they can actually perform the Sub-Control. The controls do not cover information about how to configure web servers which the CIS Benchmarks do not cover.

4.2 OWASP Testing Guide

The OWASP Testing Guide is part of the OWASP Testing Project, which has the aim to "help people understand the what, why, when, where, and how of testing web applications" [152]. The Testing Guide describes a testing framework and techniques to implement the framework. According to OWASP "Many industry experts and security professionals, some of whom are responsible for software security at some of the largest companies in the world, are validating the testing framework." The current version is v4 and was most recently updated in 2016 [143]. There are more than 50 authors and 7 reviewers for the Testing Guide v4 and it is not mentioned if the authors/reviewers work at or represent a certain organization or company [151]. Although this is a testing guide and not strictly a guideline of how to avoid security misconfiguration, relevant information regarding server configuration was found, including information that the CIS Benchmarks do not cover.

The OWASP Testing Guide v4 have nine chapters with subchapters that covers how to test different areas of a web application. Usually, the subchapters follow the syntax:

- Summary
- How to test

Where **summary** describes why the test(s) are performed and what can happen if an attacker would succeed with the operation that the test examine, and **how to test** usually have descriptions of Black Box Testing and Gray Box Testing. Sometimes the subchapters have content with recommendations and guidelines in how and why certain things should be done, although this is not the general case.

There are subchapters with content related to how to test the configuration of servers, but it is seldom that an entire subchapter is specifically targeted against webservers and their configuration. E.g, in the subchapter 4.3.2 Test Application

Platform Configuration (OTG-CONFIG-002) [145], the Black Box Testing cover Sample and known files and directories, i.e. that unnecessary sample applications and files are removed, Comment review, i.e. ensuring that comments included inline in HTML code does not leak information and System Configuration, which recommends using CIS-CAT to asses systems' conformance to CIS Benchmarks. The Gray Box Testing states that "It is impossible to generically say how a server should be configured, however, some common guidelines should be taken into account: " [145], and then 14 guidelines follow, e.g:

- "Only enable server modules (ISAPI extensions in the case of IIS) that are needed for the application. This reduces the attack surface since the server is reduced in size and complexity as software modules are disabled. It also prevents vulnerabilities that might appear in the vendor software from affecting the site if they are only present in modules that have been already disabled [145]".
- "Handle server errors (40x or 50x) with custom-made pages instead of with the default web server pages. Specifically make sure that any application errors will not be returned to the end-user and that no code is leaked through these errors since it will help an attacker. It is actually very common to forget this point since developers do need this information in pre-production environments [145]".
- "Make sure that the server software runs with minimized privileges in the operating system. This prevents an error in the server software from directly compromising the whole system, although an attacker could elevate privileges once running code as the web server [145]".

Thus, the subchapter 4.3.2 Test Application Platform Configuration (OTG-CONFIG-002) covers both issues specifically related to configuration of web servers but also cover developer comments which is not strictly related how to a web server is configured.

It is also not obvious which subchapters actually contain information regarding server configuration. Most of the subchapters belonging to chapter 4.3 Configuration and Deployment Management Testing mention tests of web server configuration or how the configuration should be, except for 4.3.8 Test RIA cross domain policy (OTG-CONFIG-008), but there are also relevant content in other subchapters, but not all. Examples of relevant subchapters are 4.2.2 Fingerprint Web Server (OTG-INFO-002), 4.8.3 Testing for HTTP Verb Tampering (OTG-INPVAL-003, 4.9.1 Analysis of Error Codes (OTG-ERR-001), 4.10.1 Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection (OTG-CRYPST-001) and 4.10.4 Testing for Weak Encryption (OTG-CRYPST-004). However, this is not obvious and required manual examination of the different subchapters. Another matter is that there is seldom guidance of how to remediate the configuration and what correct values are. Thus, even if the reader discern that the information and tests covered in a subchapter is relevant to the security of the configuration of a webserver, he will most likely need to spend time to understand how to correctly configure his web server. This can become tedious since there are numerous configuration settings covered.

4.2.1 Comparison between OWASP Testing Guide and CIS Benchmarks

Most of the content related to secure configurations of web servers is covered in CIS NGINX and CIS Apache, but there are cases where the OWASP Testing Guide v4 cover areas which neither of the CIS Benchmark mention, or only one of them.

4.2.1.1 Sensitive Information

In Test Application Platform Configuration (OTG-CONFIG-002) [145] under the headline "Logging" it is mentioned that sensitive information should be kept out of logs, and this is only covered by the CIS NGINX (CIS NGINX 3.1) and not CIS Apache. In the same chapter it is also mentioned that log information should never be visible to end users or even web administrators as it breaks separation of duty controls. This is not mentioned by either CIS NGINX or CIS Apache.

Testing for Sensitive information sent via unencrypted channels(OTG-CRYPST-003) [148] mention to check if password or encryption keys are hardcoded in the source code or configuration files, which neither CIS Apache or CIS NGINX mention. When installing the default distro package for Ubuntu 18.04 for Apache HTTP and Nginx there are no hardcoded passwords och encryption keys, but it could still be wise to check for this.

4.2.1.2 HTTP

Another difference in recommendation is that in Testing for HTTP Verb Tampering (OTG-INPVAL-003) [147]. OWASP recommends only to accepts GET or POST request, while CIS Apache recommends GET, HEAD, POST and OPTIONS to be accepted in CIS Apache 5.7 and CIS NGINX recommends GET, POST and HEAD to be accepted in CIS NGINX 5.1.2.

4.2.1.3 Permissions and Ownership

In Test File Permission (OTG-CONFIG-009) [146] there are differences in recommendations for permissions on different filetypes. Configuration Directory have the recommended permission 700(rwx——) and Configuration 600(rw——-) while CIS Apache 3.6 recommends that permissions on Apache directories should generally be rwxr-xr-x (755) and file permissions should be similar except not executable unless appropriate. This applies to all of the Apache software directories and files installed with the possible exception of the web document root [23] and CIS NGINX 2.3.2 recommends 750 on all etc/nginx directories and 640 on all `/etc/nginx` files. Thus, OWASP recommend more strict permissions than both CIS Apache and CIS NGINX.

4.2.1.4 SSL/TLS

OWASP also gives more extensive information regarding cryptography and TLS configuration. In Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection (OTG-CRYPST-001) [150] various recommendations are given. It is among other things stated that weak ciphers must not be used, e.g. less than

128 bits, no NULL ciphers suite and no Anonymous Diffie-Hellmann. To exclude SSL ciphers with less than 128-bit, only those belonging to the category HIGH should be allowed, i.e. those with key lengths larger or equal to 128 bits [65]. CIS Apache mention that "it is critical to ensure the configuration only allows strong ciphers greater than or equal to 128-bit to be negotiated with the client" in CIS Apache 7.5 but does only exclude ciphers belonging to the category LOW, i.e. low strength encryption cipher suites, currently those using 64 or 56 bit encryption algorithms [65]. CIS NGINX gives different recommendation for configuration of what SSL ciphers to use for different types of server("Server block configuration for client connectivity to web server", "FIPS 140-2 compliant web server", "No weak ciphers SSLLABS web server" and "Mozilla modern profile web server" are those relevant for this thesis) in CIS NGINX 4.1.5. The importance of 128-bit ciphers is never mentioned, and only recommendation for "No weak ciphers SSLLABS web server" and "Mozilla modern profile web server" disables ciphers belonging to categories other than HIGH. As this is not the general suggested configuration and that the importance of 128 bits is not mentioned, CIS NGINX, as well as CIS Apache, can not be said to put emphasis on 128-bit ciphers.

Both CIS Apache and CIS NGINX exclude NULL ciphers in the recommendations in CIS Apache 7.5 and CIS NGINX 4.1.5. Only CIS Apache efficiently exclude all Anonymous Diffie-Hellmann in CIS Apache 7.5 through the option !aNULL, i.e. excluding the anonymous DH algorithms and anonymous ECDH algorithms. In CIS NGINX 4.1.5 anonymous DH cipher suites but not anonymous Elliptic Curve DH (ECDH) cipher suites are excluded through the use of !ADH in "Server block configuration for client connectivity to web server" and "FIPS 140-2 compliant web server", the other two exclude all anonymous DH algorithms.

Not allowing Export (EXP) level cipher suites [181] is also mentioned. These are disabled by default as of OpenSSL 1.0.2, but it can also be disabled through configuration. It is covered by disabling Export strength encryption algorithms through the use of !EXP in CIS Apache 7.5/7.8 and by all recommendations in CIS NGINX 4.1.5 except in "No weak ciphers SSLLABS web server configuration" which instead does it by only accepting those with key lengths larger than 128 bits.

In this chapter it is also mentioned that servers should support Forward Secrecy [165]. This is done by supporting Ephemeral Diffie-Hellman, i.e. cipher suites belonging to the categories kDHE/kEDH, DHE/EDH, kECDHE/kEECDH, ECDHE/EECDH, or only DHE/EDH or ECDHE/EECDH to exclude anonymous cipher suites [65]. This is covered by the recommended configuration in both CIS Apache 7.5/7.8 and CIS NGINX 4.1.5, but only the recommendation in CIS NGINX 4.1.5 "Mozilla modern profile web server" exclude all key-exchange algorithm that does not provide Forward Secrecy.

Testing for Weak Encryption (OTG-CRYPST-004) [149] mention several hash and encryption algorithms that should not be used. Those that are relevant for this thesis, i.e. configurable to support or not for a web server, are MD5, RC4, DES and SHA1. MD5, RC4 and are disabled in CIS Apache 7.5/7.8 and in all of the recommendations in CIS NGINX 4.1.5, except "No weak ciphers SSLLABS web server configuration" for MD5 and RC4. However, neither MD5 or RC4 is supported in the TLS v1.2 cipher suites, and since Nginx is recommended to only

support TLS 1.2 in CIS NGINX 4.1.4, they are covered [65]. DES is disabled in CIS Apache 7.5/7.8 and CIS NGINX 4.1.5 by excluding cipher suites belonging to the category LOW [65] or specifically stating what ciphers suites are supported. CIS Apache also excludes 3DES in CIS Apache 7.8, and CIS NGINX does this indirectly by recommending to only support TLS v1.2 cipher suites as 3DES is not supported in TLS 1.2. This is also the case for SHA1. CIS Apache does not disable SHA1.

4.2.1.5 Key entropy

That keys need to be generated with proper entropy is also covered in chapter Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection (OTG-CRYPST-001) [150], which is not mentioned by neither CIS Apache or CIS NGINX. More information about this issue can be found in [186] where entropy and entropy production is discussed. It states that OpenSSL does not verify how much entropy is in the random data it pulls from `/dev/urandom` and that since it is not certain that `/dev/urandom` have a high level of entropy, as it will provide data from its PRNG regardless of the amount of entropy in the entropy pool, the random data might not be that random. It continues with that as OpenSSL only seeds its internal Pseudo Random Number Generator (PRNG) once per runtime this is an issue for long running processes, such as servers that link to the OpenSSL libraries and that all PRNG operations performed for the duration of the server only have as much entropy as was available at the invocation of the OpenSSL library. The problem of `/dev/urandom` is however debated. In [99], it is argued that using `/dev/random` rather than `/dev/urandom` is not to recommend in most use cases as they stem from the same cryptographically secure pseudorandom number generator (CSPRNG). Also, that since `/dev/random` blocks when it is estimated to have run out of entropy, `/dev/random` diminish availability which might lead to undesirable work arounds or a not well-functioning service. When running "man urandom" in the terminal it is stated that "The /dev/random interface is considered a legacy interface, and /dev/urandom is preferred and sufficient in all use cases, with the exception of applications which require randomness during early boot time;". Thus, it seems as `/dev/urandom` is a sufficient source of entropy. Still, if a user believes that `/dev/urandom` is not secure enough, Apache users have the possibility to configure sources for seeding the PRNG in OpenSSL at startup time and/or before a new SSL connection through the directive called SSLRandomSeed [72]. Nginx users do not have this possibility, and would need to use OpenSSL to reseed [186]. In conclusion, the default source for random data for OpenSSL seems to be good enough and thus not something that needs remediation, but Apache users have the possibility to configure other options if they would prefer.

4.2.1.6 Protection Against Famous Attacks

SSL/TLS compression, which enables the so called CRIME attack [162], is another problem that OWASP mention in chapter Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection (OTG-CRYPST-001) [150]. In Apache, it is

possible to configure whether the web server should accept SSL/TLS compression or not and the directive SSLCompression is recommended to be set to off in CIS Apache 7.7. SSL/TLS compression is briefly mentioned in CIS NGINX 4.1.14, and it is stated that HTTP/2.0 disables this. As mentioned previously, there might be a risk that HTTP/2.0 is not used even if it is configured. However, this is not a problem, as SSL compression is disabled for all versions of OpenSSL since Nginx 1.3.2 [103].

In chapter Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection (OTG-CRYPST-001) [150] it is mentioned that SSL Renegotiation must be properly configured to avoid Man in the Middle [49] and Denial of Service [166] attacks. The fix for these attacks, RFC 5746 [100], is available in OpenSSL as of version 0.9.81 [67] by disabling all renegotiation. It is also mentioned in CIS Apache 7.6. There is an option to allow renegotiation despite the RFC 5746, and this option is disabled in CIS Apache 7.6. In CIS NGINX, it is mentioned in CIS NGINX 4.1.14 that by enabling HTTP/2.0 renegotiation is disabled. Unfortunately, if a client does not support HTTP/2.0, a lower version of the protocol will be used [110] and the server is not protected by HTTP /2.0. However, since Nginx 0.8.23 client-initiated renegotiation has not been allowed [103], and thus Nginx servers as of Nginx 0.8.23 is protected from this attack. This is not mentioned at all in the CIS NGINX, reasonably as it is not configurable in modern versions, but one could still prefer some information about this to educate the reader.

Several other attacks are also mentioned in this chapter. The BEAST attack [164] is not a problem if not TLS 1.0 or earlier is enabled. CIS Apache recommends to disable TLS 1.0 in CIS Apache 7.4/7.9 and CIS NGINX 4.1.4 disabled all but TLS 1.2. An instance of the CRIME attack, called BREACH [153], take advantage of compression, but it is not as straightforward to act against as just disabling SSL/TLS compression. Disabling HTTP compression is the most effective counter measurement [153], but this can have performance and financial effects [163]. Different methods of counter measurements are discussed in [163]. The attack is mentioned in CIS NGINX 2.1.3, where the compression functionality gzip is disabled. The attack or counter measurements against it is not mentioned in CIS Apache. Another instance of the CRIME attack, called the TIME attack is also mentioned. The attack is not as documented or discussed as CRIME or BREACH [183] [140] and it is not easy to find counter measurements. However, several solutions is mentioned in [175], and it seems that enabling X-Frame-Options is the only counter measurements related to web server configuration. This is done in CIS Apache 5.14 and CIS NGINX 5.3.1. The FREAK attack [57] is not a problem when disabling export cipher suites, which is as previously mentioned is covered in CIS Apache 7.5/7.8 and by all recommendations in CIS NGINX 4.1.5. Heartbleed is an attack related to OpenSSL and counteracted by not using OpenSSL 1.0.1 through 1.0.1f (inclusive) [185], thus not strictly related to the configuration of a web server. The weak algorithms MD2, MD4, MD5 and SHA1 is also mentioned. MD2 and MD4 is not supported in OpenSSL [65], and MD5 is disabled in CIS Apache 7.5/7.8 and by all recommendations in CIS NGINX 4.1.5. Nginx is recommended to only support TLS 1.2 in CIS NGINX 4.1.4 and thus disabled usage of SHA1 since TLS 1.2 do not support it. CIS Apache does not

disable the usage of SHA1.

4.2.1.7 Summary

Table 4.1 is a summary of missing or different recommendations in CIS Apache and CIS NGINX in comparison with the OWASP Testing Guide v4. Recommendations that are in alignment are not mentioned.

4.3 DISA STIG

There are DISA STIGs for several types of software, including web servers databases, operating systems and runtime environments, however only for Apache and not Nginx [55]. Viewing the content is not effortless, it is said in the manual of the STIG that it can be viewed in a web browser, this however failed. Downloading and using the so called STIG Viewer tool, available at [54], was necessary to access the content properly.

No individual authors are named, the STIGs are only stated as a document developed by DISA for the DoD. Each guideline has a vulnerability ID, a rule ID, a STIG ID, a severity grade and a classification. The three severity grades. CAT I, "Any vulnerability, the exploitation of which will directly and immediately result in loss of Confidentiality, Availability, or Integrity", CAT II, "Any vulnerability, the exploitation of which has a potential to result in loss of Confidentiality, Availability, or Integrity" and CAT III, "Any vulnerability, the existence of which degrades measures to protect against loss of Confidentiality, Availability, or Integrity." The guidelines follow the syntax:

- Group Title
- Rule Title
- Discussion
- Check Text
- Fix Text
- References

Where **discussion** describe background and rationale behind the guideline, **check text** describe how to test if recommended state is implemented or not, **fix text** describe how to remediate if the recommended state is not implemented, and references is references to relevant sources of information for this particular guideline, usually a document provided by NIST.

There are in total 76 guidelines for Apache 2.4. They are divided into two categories, one for server-related guidelines and one for website-related guidelines. Both categories should be applied to an Apache Server according to the README file. Slightly more than half of the guidelines in the website-related category is a copy of of the server-related category. Some guidelines exists which are not relevant for this thesis such as recommendations regarding configuration of proxy/load balancer server or the necessity to interview personnel or review documentation.

Table 4.1: Comparison between OWASP Testing Guide and CIS Benchmarks

ID	Recommendations by OWASP	Status in CIS
OWASP-01	Sensitive information should be kept out of logs	Missing in Apache
OWASP-02	Log information should not be visible to end users/web administrators	Missing in Apache & NGINX
OWASP-03	Only accept GET or POST request	Differs from Apache & NGINX
OWASP-04	Permission 700(rwx——) on configuration directory	Differs from Apache & NGINX
OWASP-05	Permission 600(rw——-) on configuration	Differs from Apache & NGINX
OWASP-06	No ciphers with less than 128 bits	Missing in Apache & partly NGINX
OWASP-07	No Anonymous Diffie-Hellmann	Missing partly in NGINX
OWASP-08	Keys need to be generated with proper entropy	Missing in Apache & NGINX
OWASP-09	Servers should support Forward Secrecy	Differs partly from NGINX
OWASP-10	Protect Against the BREACH attack	Missing in Apache
OWASP-11	Do not use weak algorithm SHA1	Missing in Apache
OWASP-12	Check If Password or Encryption Keys Is Hardcoded In The Source Code or Configuration Files	Missing in Apache & NGINX

Several CIS Apache recommendations are covered, CIS Apache 2,3, 2.6, 3.2, 3.6, 4.1, 5.4, 5.7, 5.8, 5.13, 6.1, 6.5, 7.1, 7.2, 7.7, 7.9, 9.1, 9.2, 9.3, 9.5. There is also numerous guidelines covering areas that CIS Apache does not or that differ in recommended argument. They ones that CIS Apache is missing can be seen in Table 4.2 and the ones that differ in recommended arguments are discussed below. The STIG IDs are shortened to U1-XXXX (server-related category STIG ID) and U2-YYYY (websit-related category STIG ID), e.g. AS24-U1-000010 to U1-0010 and AS24-U2-000780 to U2-0780.

- U1-0070 recommends the LogFormat directive to have "%a %A %h %H %l %m %s %t %u %U \"%{Referer}i\"" , i.e. to log Client IP address of the request, Local IP-address, Remote hostname, the request protocol, Remote logname, The request method, Status, Time the request was received, Remote user, The URL path requested and the contents of Referer header line(s) in the request sent to the server. CIS Apache 6.3 recommends "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User- agent}i\"", i.e. to log Remote hostname, Remote logname, Remote user, Time the request was received, the request line that includes the HTTP method used, the requested resource path, and the HTTP protocol that the client used, the Final status, Size of response in bytes and the contents of Referer and User-agent header line(s) in the request sent to the server. Thus, DISA STIG suggest to log some more information than CIS Apache. As this information might be useful for analysis, it can be argued to be more beneficial to use the recommendations by DISA STIG. server [190] [112].
- U1-0230 recommends to review what modules are loaded, except for the modules named core_module, http_module, so_module and mpm_prefork_module which are described to be needed for core functionality. In CIS Apache chapter 2 several modules are mentioned to be enabled or disabled, ones above are however not mentioned. As they are part of core functionality, its reasonable that they are not mentioned, but it is nice for an inexperienced user to know that these are not modules he will need to worry about.
- U1-0310 recommends to review the directives named Script, ScriptAlias or ScriptAliasMatch, and ScriptInterpreterSource and remove cgi scripts that is not needed for application operation, while CIS Apache 5.5 and 5.6 mention two particular scripts, printenv and test-cgi, to be removed. CIS Apache mention that Apache is loaded with default scripts, but it would be beneficial with an explicit recommendation to review all default scripts.
- U1-0330 recommends to remove the Web Distributed Authoring (WebDAV) modules dav_module and dav_fs_module, which CIS Apache 2.3 also does. U1-0330 mention the dav_lock_module, which CIS Apache does not. However, in the documentation for the dav_lock_module it says that the module requires the service of mod_dav, thus if mod_dav is disabled, this module should not be a problem [189].
- U1-0620 recommends to modify Warning and error messages to minimize the identity of the Apache web server, patches, loaded modules, and directory

paths through the directive named ErrorDocument, e.g. with parameter "ErrorDocument 500 'Sorry, our script crashed. Oh dear' ". CIS Apache mention disguising error messages in 6.7 relating to the configuration of the OWASP ModSecurity Core Rules Set (CRS) for the ModSecurity web application firewall (WAF). It is stated in CIS Apache 6.6 that ModSecurity requires a significant commitment of staff resources for initial tuning of the rules and handling alerts and an ongoing commitment for monitoring logs and ongoing tuning and that without this commitment to tuning and monitoring, installing ModSecurity may not be effective and may provide a false sense of security. As it is not certain that users have the resources possible to maintain ModSecurity, the recommendations in U1-0620 is easier to implement and thus probably more efficient.

- U1-0820 cover recommendations for securing the ID(pid) file which CIS Apache 3.9 also cover. They give similar recommendations except for that CIS Apache also mention to ensure that the pid file directory is not a directory within the Apache DocumentRoot since if it is placed in a writable directory, other accounts could create a denial of service attack and prevent the server from starting by creating a pid file with the same name. This is a recommendation that DISA STIG is missing.
- U1-900 recommends to only allow TLSv1.2 while CIS Apache allows both TLSv1.1 and TLSv1.2. CIS NGINX 4.1.4 states that because of the increased security associated with higher versions of TLS, TLS 1.1 should be disabled and that modern browsers will begin to flag TLS 1.1 as deprecated in early 2019. Thus, it is reasonable to only allow TLSv1.2.

Not all of the missing recommendations are absolutely necessary to cover to consider a configuration to be secure. In STIG U1-0670 the RequireAll directives is used to restrict access of IP adresses and hosts. It is beneficial to know that this possibility exists, however, it is not certain that an organization or user has a specified list of nonsecure IP adresses or hosts. Thus, this recommendation is perhaps not necessary for the general use case and thus not a requirement to cover to ensure a safe configuration. In STIG U2-0380 it is recommended to use SSLVerifyCLient and SSLVerifyDepth. These are however only beneficial when you know all of your users, e.g. as is often the case on a corporate Intranet. As this is not the general use case it is not necessary a requirement to cover to ensure safe configuration.

Table 4.2: Summary of information covered by the DISA STIG for Apache 2.4 and not by CIS Apache

STIG ID	Subject
U1-0010	session_module and usertrack_module should be enabled
U1-0180	Correct permissions on log files
U1-0190	Correct ownership of log files
U1-0240	Ensure that the web server is not being used as a user management application
U1-0300	Ensure that the web server only allow hosted application file types to be served
U1-0460	Invalidate session identifiers upon hosted application user logout or other session termination
U1-0470	Ensure that cookies exchanged have security settings that disallow cookie access outside the server and hosted application
U1-0510	Ensure that generated session IDs are long enough so they cannot be guessed through brute force
U1-0520	Ensure that generated session IDs use as much of the character set as possible to reduce the risk of brute force
U1-0670	Restrict inbound connections from nonsecure zones
U1-0820	Ensure correct ownership and permissions on utilities used to start/stop the server
U1-0870	Ensure that cookies exchanged have cookie properties set to prohibit client-side scripts from reading the cookie data
U1-0900	Ensure correct ownership and permission on htpasswd files
U2-0380	Ensure RFC 5280-compliant certification path validation
U2-0620	Ensure that a default hosted application web page is displayed when a requested web page cannot be found

Chapter 5
Analysis of Tools

This chapter describes the results of the analysis of the tools chosen for examination. At the end of the chapter there is a summary of the results.

5.1 OpenSCAP

5.1.1 About OpenSCAP

OpenSCAP is an open source project that provides solutions which help users to assess, measure, and enforce security baselines [136]. It is licensed under GNU Lesser General Public License v2.1 [76]. It is available for Fedora, RHEL 6, RHEL7, CentOS 6 and CentOS 7, Debian, Ubuntu and Windows [135]. OpenSCAP represents a command line tool and a library. The command line tool is called oscap and performs configuration and vulnerability scans of local systems. It evaluates the system based on the XCCDF or OVAL content given as input. The library allows for the swift creation of new SCAP tools [135]. The library is used by other tools such as the SCAP Workbench, which provides a graphical tool to perform the operations that oscap do through the command line, on local or remote systems [139]. OpenSCAP provides ready to use SCAP content, named SCAP Security Guide. The SCAP Security guide implements security recommendations by authorities such as Payment Card Industry Data Security Standard(PCI DSS) [**?**], STIG, and United States Government Configuration Baseline(USGCB) [138]. The SCAP Security Guide provide SCAP content primarily for operating systems but also for other software e.g. web browsers, see Figure 5.1:

I.e, to use OpenSCAP for e.g. Apache HTTP, users need to provide XCCDF or OVAL content themselves. This is also the case for proprietary solutions found such as Joval which is a tool that allows both local and remote automated testing based on SCAP. They provide SCAP content for common operating systems and also allow users to provide their own content [107–109]. Thus, users need to provide XCCDF or OVAL content themselves to test e.g. Apache HTTP. Possibilities to do this is found in Section 5.1.2. When examining the tool, the command line tool oscap and content from the official OVAL Definitions repository (see Section 5.1.2) was used. No errors was found when running the tool. The result was saved as an XML file, available at [1], and viewed in a web browser. There was no ranking of the result and no motivation behind the tests. However, there were links to the

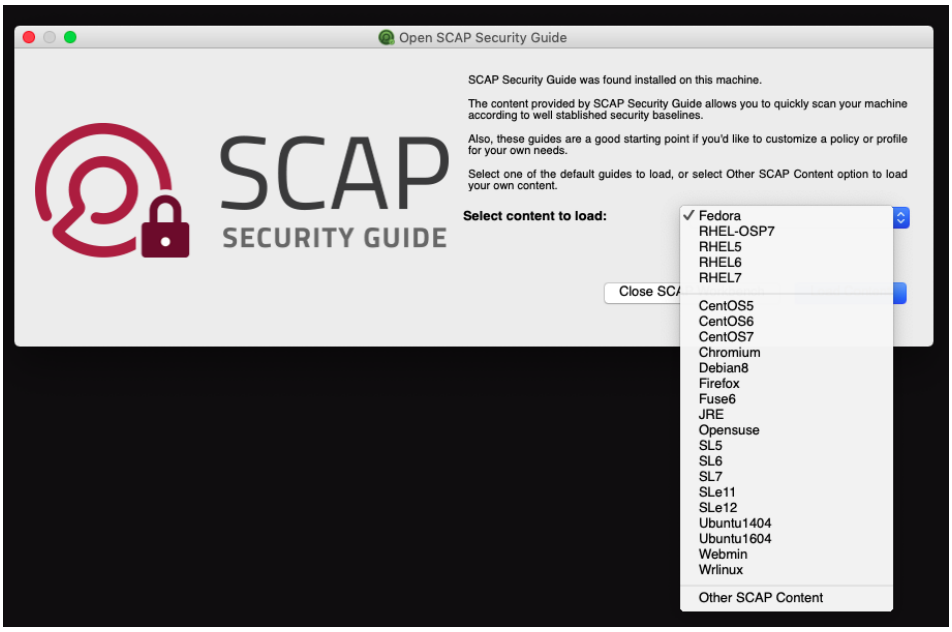

Figure 5.1: Front page of SCAP Workbench showing the SCAP Security Guide provided by OpenSCAP

"Reference ID", i.e. the CVE or similar identifications of published vulnerabilities, and thus users are guided how to find out more information of the problem behind the test. More information about the content of the tests is found in Section 5.1.2.

5.1.2 Obtaining OVAL and XCCDF content

Users can write their own XCCDF or OVAL content. This is however not a straightforward task at present. As stated in the NIST paper "Transitioning to the Security Content Automation Protocol (SCAP) Version 2", one of the critical gaps with SCAP v1 is "**Difficult Content Creation and Limited Content Availability:** Development of content in SCAP v1 usually requires 1) familiarity with complex SCAP v1 languages; and 2) knowledge of the low-level system artifacts needed to direct collection and evaluate endpoint posture. Finding SCAP content developers that have the necessary skills and knowledge to do both can be challenging. This has increased the time and expense necessary to create content, causing limited content availability" [195]. This problem is also mentioned in [22], where standardized specification formats like XCCDF is described as "hard and cumbersome to comprehend and encode in" [22], and they show that XCCDF/OVAL requires many more lines of code compared to the novel language they present in their paper, one example with 45 lines compared to 10.

There are however options for users who are not interested in or don't have the time to spend to comprehensively understand OVAL and XCCDF. The continuing part of this section will exhibit the result of the research of finding open source

XCCDF/OVAL content.

There is an official repository for OVAL Definitions. It was previously maintained by MITRE, but their website is in "Archive" status and was lastly updated 2016 [119]. The repository is today hosted by CIS and is updated several times per month [26]. There is a vast amount of platforms and products covered, including Apache HTTP, Nginx. There are more than 60 OVAL Defintions in the product category apache, more than 50 in the product category apache2, 1 in the product category apache httpd and 15 in the product category nginx. All of the OVAL Definitions of different product categories (apache, apache2, apache httpd, nginx, mysql and postgresql) was downloaded and examined. All of the OVAL Definitions covered if a certain software, such as an operating system, the product itself, a patch or a package is installed and what version it have. Thus, they do not test more granular settings such as what encryption ciphers is supported or disabled to use by SSL/TLS protocols or if the HTTP TRACE request method is disabled. It might be that the correct setting, e.g. disabling insecure encryption ciphers, is included in a certain version or patch, but if a user have unfortunately changed that setting after downloading the version or patch, this will remain unnoticed.

The American Defense Information Systems Agency (DISA) provide Security Technical Implementation Guides (STIGs) which are configuration standards for devices/systems used by the Department of Defense [58]. There are STIGs available for Apache HTTP and also other software such as Postgresql and MongoDB Enterprise but there is nothing to be found for Nginx or other databases such as MySQL. As described in [75] a STIG is packaged in a zip file that contains numerous files. SCAP relevant content is the STIG manual.xml described as "the STIG XML file that contains the manual check procedures" [75] and the STIG benchmark-xccdf.xml described as "the STIG XML file that contains the automated check procedures, and not the manual procedures. This file is only included for technologies that contain OVAL checks " [75]. Not any of the STIG zips examined (Apache 2.2, Apache Server 2.4, PostgreSQL 9.x, MongoDB Enterprise Advanced 3.x) contained a xccdf.xml file which was possible to use for automated test. Worth noticing is that the previously mentioned STIG manual.xml in the STIG zips is written in XCCDF in the examples examined. But this is still only manually usable content.

NIST provides a repository called National Checklist Program Repository with "metadata and links to checklists of various formats including checklists that conform to the Security Content Automation Protocol (SCAP)" [131]. However, no SCAP content to use for automated tests was found for the applications examined (such as Apache 2.4/2.0/1.3, Apache Tomcat, Debian, Kubernetes 1.8.0/1.7.0/1.6.0/1.13.0, Nginx and Postgresql 9.x/9.6/9.5). There are only references to previously mentioned CIS benchmarks or STIG zips available.

Organizations such as Red Hat, The SUSE Linux Enterprise and the Debian Project hosts repositories of OVAL content, however these only cover Linux distributions [98] [184] [154].

The security and management technology company SecPod provides a SCAP repository but it only covers operating systems [178] [177].

The security company Security Database hosts an OVAL repository. There are OVAL content available for a range of operating systems and VM/Microsoft

windows/MacOS server, however, the most recent updates are from 2015 [179]. It also reference to MITRE, which as previously mentioned has not been managing the official OVAL repository since at least 2016.

The IT Security Database is a mirror of several other websites which provide SCAP content. These are websites of the companies/organizations behind operating systems or websites with content relating to operating systems. There is also a reference to Internet Explorer and Apache HTTP. Unfortunately, the link provided to Apache HTTP website is broken and there is nothing to be found regarding SCAP, OVAL or XCCDF at www.apache.org or httpd.apache.org. It also references to MITRE. Of the OVAL defintions explored at the site, the latest one was updated at 2015, and since it references to MITRE, no longer the official OVAL repository as mentioned previously, and not recent versions of the software of the websites (e.g, Windows 7 as the most recent version of Windows which has not had mainstream support since 2015 [188]), the IT Security Database seem to not be maintained.

The final repository examined is OVALdb, a repository owned by the company ALTEX-SOFT. On their website they state that "OVALdb consists of an open-source part containing information of the upper-level descriptions (OVAL language, no checks), and a private part (OVAL language with checks), including vulnerabilities, patches, inventory and compliances that can be exported into XML-files for automated scan settings" [3]. In the database, there seem to be more that 331094 items of OVAL Definitions but there is no possibility to download any content to use for automated testing, as a non-paying user you can only see description and high level content.

As mentioned, OpenSCAP and the SCAP Security Guide provide ready to use content, but it unfortunately mostly covers operating systems and not common applications as mentioned in Section 5.1.1. Worth noticing is that the SCAP Security Guide does cover more granular settings than the OVAL Definitions from the official OVAL repository. There are e.g. tests for Ubuntu 16.04 that examine if only SSH protocol version 2 connections are permitted and verifying permissions and ownership of `/etc/passwd`. Thus, there are use cases where more granular tests are written in XCCDF/OVAL.

5.2 CFEngine

5.2.1 About CFEngine

CFEngine is a framework for configuration management and automation. There are two editions, the CFEngine Enterprise and CFEngine Community. CFEngine Community was first released in 1993 and is licensed under the GNU General Public License, version 3 [4] [5].

Through a domain specific language of CFEngine users can define desired states, so called policies, of the components of their IT infrastructure. An agent, described as lightweight on the CFEngine website, runs locally on the components by default every five minutes and to ensure that the components converge to the specified desired states and then report the outcome of each run. Before each run, the agent will try to connect to a policy server which contain all policies to see

there is an update to the policies. The policy server is hosted and maintained by the user of CFEngine. The policy server needs CentOS/RHEL, Debian or Ubuntu and the components running the agent, called Clients or Hosts, can have various operating systems, e.g. Debian, Solaris or Windows [20].

5.2.2 About CFEngine policies

As mentioned, the desired states are specified by policies. The syntax and architecture of the policy files and the domain specific language will not be covered in detail as this is out of scope for this thesis, but the concept of promises and classes will be mentioned. Promises are described at CFEngines website as: "One concept in CFEngine should stand out from the rest as being the most important: promises. Everything else is just an abstraction that allows us to declare promises and model the various actors in the system" [19]. A promise can be e.g. that a certain port should be open on a web server, or that a directory has a certain set of permissions or owner. The promise made by a promiser, which can be anything from a file, a package, access control or a database [19] [18].

An example of a promise is shown on CFEngines website [19], which is showed in Figure 5.3:

```
# Promise type
files:
    "/home/mark/tmp/test_plain" -> "system blue team",
        create  => "true",
        perms   => owner("@(usernames)"),
        comment => "Hello World";
```

Figure 5.2: Example of a CFEngine promise

The example is described as "In this example, the promise is about a file named test_plain in the directory /home/mark/tmp, and the promise is made to some entity named system blue team. The create attribute instructs CFEngine to create the file if it doesn't exist. It has a list of owners that is defined by a variable named "usernames" (see the documentation about Bodies for more details on this last expression). The comment attribute in this example can be added to any promise. It has no actual function other than to provide more information to the user in error tracing and auditing. This is a promise that will affect the state of a file on the filesystem. In CFEngine you can do this without having to execute the touch, chmod, and chown commands. CFEngine is declarative: you declare a contract (or a promise) that you want CFEngine to keep and you leave the details up to the tool" [19].

Promises can be regulated through so called classes. Through classes a promise can be made to e.g. only apply to Linux systems, or only be applied on Sundays, or only when a variable has a certain value [7].

Thus, CFEngine allows users to specify granular and context specific configuration of a great variety of objects, which is beneficial if a user would like to follow granular secure configuration recommendations. In addition, since CFEngine is

running locally, the configuration is continuously verified.

5.2.2.1 Obtaining CFEngine policies

On the CFEngine website, users are encouraged and educated in how to write their own policies [17] [9]. However, CFEngine also provide ready to use content hosted in a repository named Design Center which contains so called sketches, which are policy templates and can be configured and deployed without the user needing to have in depth knowledge of the domain specific language of CFEngine. Sketches are managed and deployed through the command line for CFEngine Community users and through a graphical user interface for CFEngine Enterprise users. To deploy a sketch a user can download it from the repository, configure it by providing relevant parameters and then deploy the runfile. A sketch can have multiple configurations depending on the parameters that are set [11]. The Design Center repository have sketches for a range of software, see Figure 5.3 [6]:

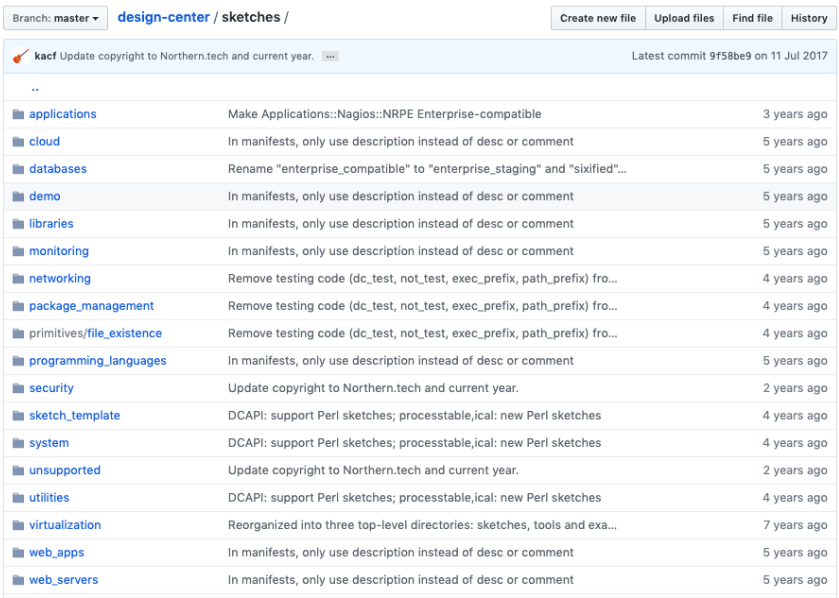

Figure 5.3: Sketches available in Design Center repository

There are only sketches available for Apache and not Nginx. The directory was lastly updated 2014 [12]. The configuration in not extensive. It specifies ports.conf to have permission 644 and to listen to the specified port and the file at `/etc/apache2`(or httpd depedning on the operating system)/sites-available/default to have permission 644 and to have the configuration presented in Figure 5.4 [14]:

The configuration in Figure 5.4 does not align well with the recommendations of CIS Apache. According to the CIS Apache 5.3 FollowSymLinks and SymLinksIfOwnerMatch is not recommended and should be disabled if possible. In CIS Apache 5.1 and 5.2 Options is recommended to be set to value None for the

33 lines (27 sloc) | 807 Bytes Raw Blame History

```
<VirtualHost *:$(cfdc_webserver:apache_conf.port)>
    ServerName $(cfdc_webserver:apache_conf.hostname)
        ServerAdmin webmaster@localhost

        DocumentRoot $(cfdc_webserver:apache_conf.docroot)
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Figure 5.4: Content available at Design Center for Apache

OS Root Directory and the Web Root Directory. Also, the value of LogLevel is recommended to be info or lower for the core module and notice or lower for all other modules in CIS Apache 6.1.

If the user have specified that SSL should be used, `/etc/apache2/sites-available/default-ssl` is set to have permission 644 and to have the configuration found in [13]. This configuration also have non-recommended setting with FollowSymLinks and SymLinksIfOwnerMatch and Log-Level not set to notice core:info. It also creates a self-signed certificate, which is not recommended by CIS. Instead, CIS recommends to use a valid certificate signed by a commonly trusted certificate authority [23].

Thus, the ready to use configuration provided lacks a considerable amount of the recommendations from CIS Apache. Worth to notice is also that the sketches for web servers and databases has not been updated since 2014 and thus some content might not be relevant today/needing an update. E.g. `/etc/apache2/sites-available/default` is not used in newer versions of Apache [192].

5.2.2.2 Using CFEngine

The sketch for Apache was first tried with version 3.12.2 of the CFEngine Community Edition. It was apparently not compatible with the Apache sketch and it merely gave tracebacks. Is was suspected that since the sketch had not been updated for long it might be a compatibility problem and the CFEngine Community Edition version 3.6.0 was used instead, as this is stated to be lowest version needed to use Design Center Sketches [10]. This was an improvement, however, it was still

not possible to run the policy. Different approaches was tried, such as trying to run the test.cf that is mentioned in the README file mentioned in the repository [16] to get an example. This file was not available in the repository, however, a file named test.pl was. Apparently, test.pl is run by using Perl's Test::Harness [15]. This time running the test.pl file worked, but the tests themselves errored out on not finding the corresponding resources. Again, the installation seemed incompatible with the test itself. Then, running the "cf-sketch.pl" tool was tried, which is a tool that is supposed to help managing sketches [8]. This tool, however, did not even find the Apache sketch when utilizing the "search" command. It found some of the other ones contained in said git repository, but the majority of them were not found by the tool. As version in the span between 3.6.0 and 3.12.2 was not tried, there is a possibility that other versions may be compatible with the Apache sketch. If one really wanted to, they could check the date of the git commit which contained the changeset for the last update to that sketch, and then correlate that with CFEngine releases. Given that the person developing the sketch ran the (at the time) latest version, it could work. One could also look more in depth in how to write sketches for CFEngine, and then simply patch the issues as they follow. This however felt out of scope for the thesis. In summary, CFEngine Community Edition together with the Apache sketch was not possible to run out of the box.

5.3 Chef Inspec

5.3.1 About Chef Inspec

Chef Inspec is an open-source testing framework [46] provided by Chef, a company which provide DevOps solutions [47]. Chef describes Chef Inspec as "InSpec is code. Built on the Ruby programming language, InSpec tests are meant to be human-readable" [46]. Chef InSpec allows users to define security and compliance rules through its domain-specific language. These rules are run as automated tests against e.g. servers, containers and cloud APIs, the actual state of the object being examined is compared to the desired state expressed in the Chef Inspec code [31]. Chef InSpec detects eventual violations and displays the result, but the user is responsible for eventual remediation [39]. Chef Inspec can be run locally or on a remote system through SSH [46]. The current version is 3.9.0 [32]. Operating systems supported are Red Hat Enterprise Linux, mac OS, SUSE Linux Enterprise Server, Ubuntu and Windows [33].

The tests in Chef Inspec are called controls, see Figure 5.5 for an example.

The example is described in [40] as:

- **'sshd-8**' is the name of the control
- **impact, title, and desc** define metadata that fully describes the importance of the control, its purpose, with a succinct and complete description
- **desc** when given only one argument it sets the default description. As of Chef InSpec 2.3.4, when given 2 arguments (see: 'rationale') it will use the first argument as a header when rendering in Automate

```
control 'sshd-8' do
  impact 0.6
  title 'Server: Configure the service port'
  desc 'Always specify which port the SSH server should listen.'
  desc 'rationale', 'This ensures that there are no unexpected
settings' # Requires InSpec >=2.3.4
  tag 'ssh','sshd','openssh-server'
  tag cce: 'CCE-27072-8'
  ref 'NSA-RH6-STIG - Section 3.5.2.1', url: 'https://www.nsa.gov
/ia/_files/os/redhat/rhel5-guide-i731.pdf'

  describe sshd_config do
    its('Port') { should cmp 22 }
  end
end
```

Figure 5.5: Example of Chef Inspec control taken from [40]

- **impact** is a string, or numeric that measures the importance of the compliance results. Valid strings for impact are none, low, medium, high, and critical. The values are based off CVSS 3.0. A numeric value must be between 0.0 and 1.0. The value ranges are:
  - 0.0 to <0.01 these are controls with no impact, they only provide information
  - 0.01 to <0.4 these are controls with low impact
  - 0.4 to <0.7 these are controls with medium impact
  - 0.7 to <0.9 these are controls with high impact
  - 0.9 to 1.0 these are critical controls
- **tag** is optional meta-information with key or key-value pairs
- **ref** is a reference to an external document
- **describe** is a block that contains at least one test. A control block must contain at least one describe block, but may contain as many as required
- **sshd_config** is an Chef InSpec resource. For the full list of Chef InSpec resources, see Chef InSpec resource documentation
- **its('Port')** is the matcher; should eq '22' is the test. A describe block must contain at least one matcher, but may contain as many as required.

There is no clear definition on the website what a resource is, but in a repository maintained by MITRE called inspec_training_courses it is stated that "If you're familiar with Chef, you know that a resource configures one part of the system. Chef InSpec resources are similar. For example, the Chef InSpec file resource tests for file attributes, including a file's owner, mode, and permissions" [84]. Thus, a

resource can be described as the object which a control validates to see whether or not its features and values are in the desired state. There are over 150 resources available [41] but users can also write their own if they need to [44]. For more detailed information about Chef Inspec controls and their syntax, see [34].

Controls can be grouped into so called profiles if they are intended for a certain object, e.g. an Apache HTTP server. Users can write their own controls/profiles or use profiles created by the Chef Inspec community [46]. The ready to use profiles available are found at the Chef Supermarket [45] and there users are provided with links to the repositories where the profiles can be found and downloaded. There are profiles available for Apache and Nginx. There are also profiles for e.g. MySQL, PostgreSQL, Linux, Windows, PHP, Java, Meltdown and Spectre, SSH, SSL/TLS etc. There seem to be more ready to use profiles available outside of Chef Supermarket, but only for customers who have access to the tool Chef Automate. In the datasheet for Chef Inspec it is stated that "Finally, Chef Automate supplies over 90 built-in InSpec profiles for standards like the Center for Internet Security (CIS) benchmarks, Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIGs) and more, to get you started quickly with applying continuous compliance to your infrastructure" [38]. When Chef Inspec was examined for this thesis, Chef Automate was not open source, but in April it was announced that there are plans to make 100 % of the company's software open source under the Apache 2.0 license [42]. Thus, there might be more ready to use profiles available in the near future.

When running Chef Inspec, the user is presented the title, the test and the result of the test, but not the impact or description. Thus, if tests are failed, the user need to look up the file with the control to see if it is an important test and/or the eventual motivation behind the test. In the end, a summary of the amount of completed, failed or skipped controls and tests that constitute the controls, see Figure 5.6 for an example.

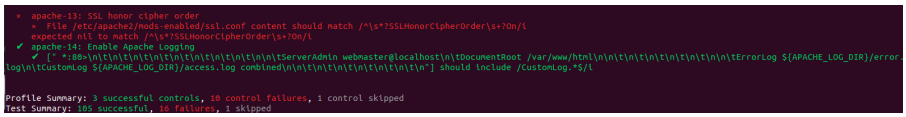

Figure 5.6: Example of result from Chef Inspec

5.3.2 Apache profiles

There are two policies with main focus on Apache, called "Apache DISA STIG" [30] and "DevSec Apache Baseline" [36]. There is also one called "myApacheTest" which is described as "Just a test to pull in a compliance profile to test in test kitchen" [43] that only contain one example control which test if Apache is running or not [86].

5.3.2.1 Apache DISA STIG

The repository for the Apache DISA STIG profile, called inspec-stig-apache, have not been updated since 2016 [87] apart for one commit that updated one line in a

control [82]. In the description of the profile it is stated that the profile is intended for “apache web server version 2.2” and that it only supports Red Hat operating systems [83]. As it only supports Red Hat operating systems, this profile will not be examined.

5.3.2.2 The DevSec Apache Baseline

The DevSec Apache Baseline is developed by the DevSec project, which is a project that provide content for automatically securing e.g. servers and operating systems [56]. The repository for the profile, called apache-baseline, is more frequently updated compared to the inspec-stig-apache, with 10+ commits per year between 2014, the year that the repository was founded, until 2018 [77]. In the description of the profile there is no specification of that it is intended towards a certain Apache version. It is however stated that it supports unix operating systems [78]. There are in total 14 controls, all of them have impact 1.0 specified, i.e. critical, except for the control named apache-03 which do not have an impact specified at all. Some of the controls have a motivation in their description, such as the control named apache-03 with the description “The Apache service in its own non-privileged account. If the web server process runs with administrative privileges, an attack who obtains control over the apache process may control the entire system”. However, there is also several controls without motivation, such as the control named apache-13 with the description “When choosing a cipher during an SSLv3 or TLSv1 handshake, normally the client’s preference is used. If this directive is enabled, the server’s preference will be used instead”, which only describes that the servers preference should be used but not why [80].

As there are only 14 controls, there are many settings recommended in CIS Apache that are not validated. It covers the first recommendation of CIS Apache 2.3, 3.1 and 7.5, the second and third of 6.3 and all of 5.3, 5.8, 8.1. The differences between configuration in DevSec Apache Baseline and CIS Apache are:

- CIS Apache endorse the permissions and ownership on Apache directories to be rwxr-xr-x root root in and should be similar for files except not executable unless appropriate in CIS Apache 3.4, 3.5 and 3.6, while the DevSec Apache Baseline endorse rwxr-x–x root root for the Apache directory and rw-r—— root root for the main configuration file in control apache-04 and apache-05. Thus, the DevSec Apache Baseline is even stricter and arguably more secure in their endorsement of permissions.

- CIS Apache endorse Apache to be run as a non-root user with the User and Group apache in CIS Apache 3.1 while the DevSec Apache Baseline endorse User and Group to be www-data in control apache-06. Since both apache and www-data are non-root and have similar privileges, there should be no difference in level of security.

- The DevSec Apache Baseline endorse that neither the cgi_module [69], the cgid_module [70] or the include_module [71] are enabled in control apache-08. However, there is no rationale in apache-08 more than “Apache HTTP should not load legacy modules”. The modules are not deprecated and or

have security warnings/suggestion of replacement in the Apache 2.4 documentation [69–71]. These modules are not mentioned by CIS Apache, it is only stated in CIS Apache 5.5 that CGI programs have a long history of security bugs. As mentioned in the OWASP Testing Guide v4 [145], only needed modules should be enabled. This, and that CGI programs have a history of security bugs, it could be beneficial endorse that these are disabled and highlight their potentially unnecessary existence, and let the user make an educated decision on whether to keep them or not.

- CIS Apache endorse to only allow the HTTP methods GET, POST, and OPTIONS in CIS Apache 5.7, while the DevSec Apache Baseline endorse to only accept GET and POST in control apache-10. To only allow GET and POST is also recommended by OWASP in Testing for HTTP Verb Tampering (OTG-INPVAL-003), and as removing unneeded functionality reduce the attack surface, DevSec Apache Baseline can arguably be seen as more secure in this area.

When running the profile, the first thing that is printed several times in the terminal is "[DEPRECATED] The 'apache' resource is deprecated and will be removed in InSpec 4.0". The reason behind this can be found in a pull request to the Chef Inspec repository, where it is stated that "The apache resource only serves as a way to surface sane per-OS defaults for a standard Apache install. Internally, this is only used by the apache_conf resource and doesn't actually perform any inspection" [79]. Thus, the "DevSec Apache Baseline" will need to be updated when Chef Inspec 4.0 is released.

Unfortunately the remaining parts of the execution of the profile does not entirely work as intended and there are some issues:

- Controls apache-05 and apache-10 test permission and content on a file called hardening.conf, assumed to be located in the conf-enabled directory. This is not a standard file enclosed with an Apache installment, and there is no description of this file in the DevSec Apache Baseline [81]. When googling for "apache "hardening.conf"" there are only 569 hits and no coherent information of what this file should contain. Thus, these controls are not clear, cause controls to fail and fail to validate the permissions on the main configuration file and the enabled HTTP methods of a general Apache server

- Control apache-06 validate if the User and Group if set to www-data and look in the main configuration file. However, in the default installation of Apache 2.4 these are set in the file called envvars and the main configuration file reference to the settings in envvars. Thus, the test fails even though the User and Group is set to www-data. This can also be verified through the terminal, on Ubuntu through the command "ps axo user,group,comm | egrep '(apache|httpd)' ".

- Controls apache-08, apache-13 and apache-14 fail with the non-informative error message "can't modify frozen String". As there is no error log, this error was examined by modifying the file apache_spec.rb where the controls

are located and commenting away parts of the controls. The problem was located in the expression “command('ls ' « some_path)”, where some_path is “File.join(apache.conf_dir, '/mods-enabled/')” in control apache-08, corresponding to etc/apache2/mods-enabled in Ubuntu 18.04, and “sites_enabled_path = File.join(apache.conf_dir, '/sites-enabled/')” in the controls apache-13 and apache-14, corresponding to `/etc/apache2/sites-enabled` in Ubuntu 18.04. If the expression is changed to command('ls /etc/apache2/ mods-enabled') and command('ls /etc/apache2/sites-enabled') respectively, the controls work as intended.

‘’

5.3.3 Nginx policies

There is one profile available for Nginx, named DevSec Nginx Baseline which is also developed by the DevSec project [37]. The repository, named nginx-baseline, is more or less maintained at the same level as the apache-baseline, with 7+ updates per year between 2014, the year that the repository was founded, until 2018 [88]. In the description of the profile there is no specification of that it is intended towards a certain Nginx version. It is however stated that it supports unix operating systems [85]. There is in total 16 controls, although at first glance there seem to be 17. However, after control named nginx-10 comes control named nginx-12, thus there is no control named nginx-11. All of the tests have impact 1.0 specified, i.e. critical. As with the DevSec Apache Baseline, some of the controls have a motivation in their description, such as the control named nginx-12 with the description “Buffer overflow attacks are made possible by writing data to a buffer and exceeding that buffer boundary and overwriting memory fragments of a process. To prevent this in nginx we can set buffer size limitations for all clients”. However, there is also several controls without any motivation, such as the control named nginx-08 with the description “Do not allow the browser to render the page inside an frame or iframe” [89] which does not explain why one should not allow browsers to render pages inside an frame or iframe. Of course a user could perform research to obtain this information, but it would be neat to have this presented directly.

As with the Apache Baseline, since there are only 16 controls, there are many settings recommended in CIS NGINX that are not validated. It covers the first recommendation of CIS NGINX 2.2.1, the first recommendation and the Mozilla modern profile web server recommendation for ciphers in 4.1.5, and all of 2.5.1, 4.1.4, 4.1.6, 4.1.8, 4.1.13, 5.3.1, 5.3.2, 5.3.3. The differences between configuration in DevSec Nginx Baseline and CIS NGINX are:

- The DevSec Nginx Baseline only validates if the main configuration file nginx.conf is not readable, writable or executable by others and that group and owner is root in control nginx-02, i.e. ******— root root, while CIS NGINX endorse the permissions and ownership to be rwxr-x— root root on directories and rw-r—— root root on files in CIS NGINX 2.3.1 and 2.3.2. Thus, DevSec Nginx baseline have less specific and secure permissions endorsed than CIS NGINX.

- The DevSec Nginx Baseline validates if the files `/etc/nginx/conf.d/default.conf` and `/etc/nginx/sites-enabled/default` exists in control nginx-03 with the motivation "Remove the default nginx config files" [89], which CIS NGINX does not mention. The file `/etc/nginx/conf.d/default.conf` does not exist when installing from default distro package, sites-enabled/default exist and it configures e.g. what ports to listen to. One could argue that by removing default configuration, a user will need to be more aware of what configuration is actually required and thus more likely avoid unnecessary functionality and reduce the attack surface. However, if the user is aware of the configuration in the files, it is not necessary that files should never exist. As the control does not necessarily imply more secure settings or only required functionality, The DevSec Nginx Baseline can arguably be said not to endorse anything more secure than CIS NGINX in this matter.
- In control nginx-04 The DevSec Nginx baseline validates that there is only one master process per environment. This is not mentioned in CIS NGINX. The control does not have any clear motivation described, and no explanations was found online or in the academia. As there is no motivation found behind this control, it does not likely provide more secure configuration.
- The DevSec Nginx Baseline validates the value of client_body_buffer_size, client_max_body_size, client_header_buffer_size and large_client_header_buffers in control nginx-06, while CIS NGINX only endorse values for client_max_body_size and large_client_header_buffers in CIS NGINX 5.2.2 and 5.2.3. They recommend the same value for large_client_header_buffers, but differ in client_max_body_size where the DevSec Nginx Baseline recommends 1k while CIS NGINX recommends 100K. The DevSec Nginx Baseline does endorse more secure settings in this matter as it covers more settings and recommends even smaller values in one case.
- The DevSec Nginx Baseline endorse that the maximum amount of simultaneous connections per IP adresss should be 5 in control nginx-07, while CIS NGINX endorse 10 simultanious connections in CIS NGINX 5.2.4. CIS NGINX does mention that the value should be set to meet the organizational policies, but of the two recommended value, the DevSec Nginx Baseline is more strict. However, as both actually control the maximum amount of simulations connections and their values are not widely different, these recommendations are quite similar.
- The DevSec Nginx Baseline endorse that the Content Security Policy (CSP) is configured in control nginx-15, which CIS NGINX also does in CIS NGINX 5.3.4. However, different values are suggested. CIS NGINX 5.3.4. endorse to allow loading resources from the same origin (same scheme, host and port) for everything (e.g. JavaScript, Images, CSS, AJAX requests etc.) [101], while control nginx-15 only configure to allow JavaScript and plugins from the same origin [101]. In other words, CIS NGINX specifies the same kind of CSP configuration but for more types of content than the DevSec Nginx Baseline. Thus, CIS NGINX endorse more secure configuration in this

matter as the Devsec Nginx Baseline omit regulation for content other than JavaScript and plugins.

- The DevSec Nginx Baseline endorse to have HTTPOnly secure set in the Set-Cookie HTTP response header in control nginx-16, which ensure that cookies related to HTTP responses with the Set-Cookie header can't be accessed by javascript and only transmitted by HTTPS [53]. CIS NGINX do not mention anything about this setting at all.

- The DevSec Nginx Baseline endorse to limit the time a persistent connection may remain open to 5 in control nginx-17 through keepalive_timeout directive, while CIS NGINX endorse a parameter of 10 or less but not 0 and show 10 as the example in CIS NGINX 2.4.3. The DevSec Nginx Baseline also endorse the value of the "Keep-Alive: timeout=time" response header field, that indicate the minimum amount of time an idle connection has to be kept opened (in seconds) [121], to be 5, which CIS NGINX do not mention. The "Keep-Alive: timeout=time" response header field does not have impact on security [191]. Thus, the parameter for keepalive_timeout directive are more or less in alignment between the DevSec Nginx Baseline nad CIS NGINX.

When running the profile, some concerns arise:

- All of controls nginx-05 to nginx-13, nginx-15, nginx-16 and nginx-17 are dependent on the command "nginx -T" and fail with the error "expected: "some_value" got: nil", where some_value is the value that the control endorse to be configured. These is due to that the command "nginx -T" requires sudo to be executed. Thus, if the user does not have privileges to run sudo or if he does not realize that this is requiered, the majority of the tests will fail.

- Controls nginx-14 ("Disable insecure HTTP-methods") and nginx-16 ("Set cookie with HttpOnly and Secure flag") skipped as this is set in the controls. This can be changed by modifying their execution condition from false to true in the nginx_spec.rb file where the controls are defined. However, these controls validates non-trivial configuration, especially control nginx-14 which makes sure that e.g. a malicious actor can't upload or delete a file on the web server through risk related HTTP methods such as PUT or DELETE. Thus, it would be preferable if these tests were not skipped by default.

- Control nginx-12 verifies if there is configuration for usage of a file with DH parameters for DHE ciphers. As the name of the file is dependent of what the user specified when running the OpenSSL command "dhparam" [66], this test will fail if the user has not specified the location and filename to be "/etc/nginx/dh2048.pem" when generating the parameters. However, it is beneficial that the test highlight that the parameters for key exchange should be DH Ephemeral (DHE) parameters with at least 2048 bits [2], which is also suggested in CIS NGINX 4.1.6.

5.4 Puppet

Puppet is a company that provides tools to help companies manage their infrastructure. There is both an enterprise and open source version of their main solution, called Puppet Enterprise and Open Source Puppet respectively [158]. The current version of the Open Source Puppet is 6.4 [161] and it is licensed under the Apache License 2.0 [90]. They both follow a so called agent-master architecture, where there is a master node with the configuration information for the fleet of agent nodes. The agent run on the node that need to be configured properly and validates whether it is corresponding to the configuration information of the master node. The agents have the possibility to make changes if the node is not in its desired state [157].

5.4.1 About the Puppet Language, Manifest, Catalogues and Modules

The desired state of a node is described through the Puppet language. The Puppet language files are called manifests, which the master compiles to a so called catalogue which is sent to a node and used to validate the state of the node [155]. To define a desired state, the Puppet language use so called resources. At minimum, a resource has a resource type, a title and a set of attributes with values. Commonly used built-in resource types are files, users, packages and services. For more information and examples of the built-in resource types see [160]. Resources can be wrapped into so called classes. Classes are code blocks that can be called elsewhere and allows reuse of Puppet code [118]. The author of [61] wrote an example of how a class and resources can function shown in Figure 5.7.

```
class openssh {

  package { 'openssh-server':
    ensure => installed,
  }

  file { '/etc/ssh/sshd_config':
    ensure => file,
    owner  => 'root',
    mode   => '0600',
  }

  service { 'ssh':
    ensure => running,
  }

}
```

Figure 5.7: Example of a Puppet class

Puppet code can be organized into modules. Each module manages a specific task, e.g. installing and configuring a piece of software. The author of [118] write that modules allow users to split their code into multiple manifests and that it is considered best practice to use modules to organize almost all of a users Puppet manifests. There is a open source web page for sharing and downloading modules called the Puppet Forge, with thousands of modules written by Puppet developers and the open source community for a wide variety of use cases [156]. For more information about modules and the module structure, see [156].

5.4.2 About Puppet Forge

Puppet Forge is as previously mentioned where a user can find ready to use modules. Modules can be marked as "Supported", i.e. being rigorously tested with Puppet Enterprise and supported by Puppet, Inc, "Partner", i.e. rigorously tested with Puppet Enterprise and supported by a partner organization, and "Approved", i.e. meeting Puppet standards for being well-written, reliable, and actively maintained [159]. There was no modules found with security in focus for neither Apache or Nginx, thus only the most popular module for each web server was examined.

5.4.2.1 puppetlabs/apache by Puppet

This is the most popular module related to Apache with more than 7000000 downloads and is marked as "Supported". It is still maintained and has more than 200+ commits per years since its founding 2011 [95]. It covers some CIS Apache recommendations such as 3.1, 4.3, 5.10, 6.3, 8.1, 9.2, 9.3, 9.4, 9.5, 9.6 and 10.2. However, it violates the majority of the recommendations. Some examples are:

- CIS Apache 2.3 as the mod_dav module is not disabled
- CIS Apache 5.1 as the root directory contains the directive Options with argument FollowSymLinks and not the argument None
- CIS Apache 5.8 as directive TraceEnable has the argument On
- CIS Apache 6.1 as directive LogLevel has the argument warn and not the argument notice or lower
- CIS Apache 8.2 as directive ServerSignature has the argument On
- CIS Apache 9.1 as directive Timeout has the argument 60 and not a value of 10 or less
- CIS Apache 10.3 as the directive LimitRequestField has the argument 8190 and not a value of 1024 or less

5.4.2.2 puppet/nginx by Vox Pupuli

This is the most popular module related to Nginx with more than 45000000 downloads. It is marked as "Approved". It is still maintained and has more than 200+ commits yearly since its founding in 2011, except for in 2012 when there was 80+

commits [96]. Without the user specifying particular configuration, using the module is equivalent of installing Nginx from source. This result in a very stripped down configuration with not much configured at all. It covers some CIS NGINX recommendations such as 2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.2.1 and first part of 4.1.5, but violates the remaining recommendations.

5.4.3 Using Puppet

Puppet and the ready to use content examined works without any errors, but requires some time to be set up as it has a agent-master architecture. The output one receives when running a module or manifest is not very informative. It only reports if a action has been taken, e.g. that a file has been created or its content has been changed, but not what specific content of that file has been changed or why. See Figure 5.8 which is the output from running the puppetlabs/apache by Puppet for an example. Rationale behind configuration decisions could be added as a comment in the code of a model or manifest, but that does not enable a very clean or consistent practice to educate users. There are no comments with rationale found in any of the examined content.

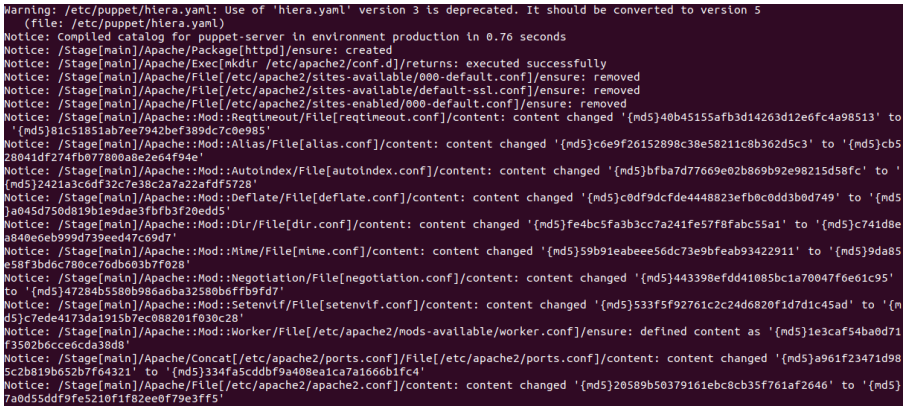

Figure 5.8: Example of output when running Puppet

5.5 Ansible

Ansible is an IT automation engine used for configuration management, provisioning, and application deployment. It is an open source community project sponsored by Red Hat. There is also an enterprise edition called Red Hat Ansible Tower that use the Ansible with a UI and RESTful API [173]. Ansible connect to nodes, over SSH by default, and push out small programs called "Ansible modules", i.e. the specified desired state of the system, executes the modules and removes them when finished. Thus, Ansible is an agentless solution.

Modules can be executed either from the command line or through so called Playbooks. The command line is intended only for quick things that is not relevant

to save for later [171] while Playbooks allow more complicated orchestration of the modules [169]. Playbooks are expressed in YAML and are intended to be a model of a configuration/process, not a script or programming language. The execution of a module in a Playbook is called a task. The Ansible documentation [170] show an example of a Playbook, see Figure 5.9. In the example, two so called plays are executed, i.e. the targeting of a group of hosts to specified tasks, one for web servers and one for database servers. The modules used in this example are yum, templates and service and they take in the key=value arguments name=httpd, state=latest, src=/srv/httpd.j2 etc. For more detailed information about the modules and playbooks, see e.g. [174] [170]. Playbooks can be run in check mode which will not make any changes on the remote systems but report what changes would have made [168].

```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that postgresql is started
    service:
      name: postgresql
      state: started
```

Figure 5.9: Example of Ansible Playbook

5.5.1 Obtaining modules and Playbooks

Ansible comes with numerous modules by default [167], but users can write their own modules. Modules can be written in any programming language [174]. There is also a web page for finding and sharing Ansible content called “Ansible Galaxy”.

This content can be loaded into Playbooks by using so called “roles”, for more information of how to do this see [172].

5.5.2 Analysis of Ready to Use Content

5.5.2.1 apache by user geerlingguy

This Playbook is downloaded more than 2 million times and is the most popular playbook related to Apache. It is still maintained and have more than 15+ commits per year since the founding in 2014 [91]. It covers some CIS Apache recommendations such as 5.8, 9.2, 9.3 and 9.4. It is almost in complete accordance with 6.3 except that virtual hosts logs the additional information the canonical ServerName of the server serving the request and the process ID of the child that serviced the request. However, it violates the majority of the recommendations. Some examples are:

- CIS Apache 2.4 is violated as Status Module is enabled
- CIS Apache 2.5 is violated as Autoindex Module is enabled
- CIS Apache 5.2 is violated as FollowSymLinks is configured for the document root.
- CIS Apache 6.1 is violated as LogLevel is configured to warn and not notice or lower.
- CIS Apache 8.1 is violated as ServerTokens is configured to OS and not Prod.
- CIS Apache 8.2 is violated as ServerSignature is configured to On and not Off.
- CIS Apache 9.1 is violated as Timeout is set to 300 and not a value equal or less than 10.

5.5.2.2 harden_apache by user juju4

This Playbook was the one with the most extensive secure configuration found for Apache, however only downloaded 41 times. None of the Playbooks related to Apache with security in focus found had more than 100 downloads, thus this does not seem to be a priority by the users of Ansible Galaxy. The Playbook is still maintained with 30+ commits per year since its founding in 2017 [93]. The configuration settings available in the Playbook sometimes have a rationale in form of a comment or a link to web page with relevant information, but in general this is not the case and the configuration settings lack rationale. The Playbook cover CIS Apache 3.1, 4.1, 4.3, 4.4, 5.7, 5.8, 5.10, 7.5, 7.7, 7.10, 8.1, 8.2, 9.2, 9.3, 9.4. It is also almost in complete accordance with 6.3 and differ in the same way as the Playbook ansible-role-apache. It also violates some recommendations, differs in recommended arguments or cover recommendations that CIS Apache does not. The ones that are missing can be seen in Table 5.2, and the ones that violated/differ in recommended arguments are discussed below. The missing recommendations are given IDs in form of juju4-apache-XX.

Table 5.1: Summary of information covered by harden_apache by user juju4 and not by CIS Apache

ID	Subject
juju4-apache-01	Deny access to potentially sensitive files
juju4-apache-02	Block vulnerability scanners and robots
juju4-apache-03	Use Rate Limiting to mitigate attacks such as DDoS
juju4-apache-04	Disable MIME sniffing to avoid Cross Site Scripting attacks
juju4-apache-05	Signal support for the upgrade mechanisms of upgrade-insecure-requests
juju4-apache-06	Signal to stop loading pages from loading when reflected cross-site scripting (XSS) is detected
juju4-apache-07	Control what resources are allowed to load for a given page to avoid Cross Site Scripting attacks
juju4-apache-08	Opt in to report on fulfillment of Certificate Transparency requirements
juju4-apache-09	Enforce cookie limitations

- harden_apache suggests the OS root directory and the web root directory to have the directive Options with argument SymLinksIfOwnerMatch, i.e. ownership must match in order for a link to be used, while CIS Apache 5.1 and 5.2 recommends the argument None. The motivation of this is that no options should be enabled for the root OS level or web root directory and that options may be enabled as needed for specific web sites or portions of the web site. This seems as a better alternative, enforcing to only use tailored options configuration in use cases where it is needed and thus limiting the attach vector.

- Both harden_apache and CIS Apache suggest argument sameorigin for the X-Frame-Options, i.e. that a web page may only be displayed in a frame on the same origin as the page itself [124]. However, harden_apache use Header set, while CIS Apache 5.14 use Header append. Thus, the response header is set and will replace any previous header with this name instead of being appended to any existing header of the same name. As Header set gives a more strict control of the header content, this could be an even better option than append.

- The LogLevel directive have the argument warn in harden_apache while CIS 6.1 suggests the argument info or lower for the core module and notice or lower for other modules, i.e. the argument notice core:info. This is motivated by that only the argument of info imply that the error logs include the not found errors, which can be used for forensics investigation and host intrusion detection purposes. As error logs are valuable for analyzing poten-

tial or occurred problems, it seems reasonable to follow the recommended configuration of CIS Apache.

- The directive SSLCipherSuite has the recommended argument EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH in harden_apache. This differs from the one suggested in CIS Apache 7.8, ALL:!EXP:!NULL:!LOW:!SSLv2:!MD5:!RC4:!aNULL:!3DES:!IDEA. harden_apache does however exclude all of the ciphers excluded in the CIS Apache recommendation, except for anonymous DH algorithms (excluded by !aNULL) which are not excluded through the use if EECDH (cipher suites using ephemeral ECDH key agreement, including anonymous cipher suites) [65].

- harden_apache exclude all TLS version except for TLSv1.2, while CIS Apache allows both TLSv1.1 and TLSv1.2 in CIS Apache 7.4 and 7.9. As discussed in Chapter 4.3, it is reasonable to only allow TLSv1.2.

- harden_apache suggest the configuration "Header set Strict-Transport-Security "max-age=16070400; includeSubDomains"" while CIS Apache 7.11 suggest "Header always set Strict-Transport-Security "max-age=600"". This setting imply that only HTTPS communication should be used rather than HTTP and that the browser should remember that a site is only to be accessed using HTTPS for the number of seconds specified by max-age. If includeSubDomains is specified, the rule applies to all of the site's subdomains as well. Using includeSubDomains is adressed in CIS Apache 7.11 and it is stated that using this option require carefully consideration of all various host names, web applications and third-party services used to include any DNS CNAME values that may be impacted. Thus, it seems wise not to use this option freely.

- harden_apache suggests a value of 100 seconds for the TimeOut directive, while CIS Apache 9.1 recommends a value of 10 seconds or less to decrease the timeout for old connections to migate DoS attacks. As CIS Apache 9.1 states, there is no 100% solution for preventing DoS attacks, but it seems reasonable to at least take action to migate the risk where it is possible. Thus, follow the recommendation given by CIS Apache 9.1 is preferable.

Some content of harden_apache are not mentioned above they do not directly effect the vulnerability status of a web server. These are:

- "Header set X-Robots-Tag none", which only if pages will be indexed or not by search engines [97]

- Configuration related to OSCP stapling as this is related to the security of the client and not the web server [111].

- "Header set Referrer-Policy origin" which implies to only send the origin of the document as the referrer which might protect sensitive client information but not directly affect the security of the server [122]

5.5.2.3 nginx by user geerlingguy

This Playbook is downloaded more than 2,5 million times and is the most popular playbook related to Nginx. It is still maintained with more than 10+ commits per year since its founding in 2014 [92]. It is It covers some CIS Apache recommendations such as 5.8, 9.2, 9.3 and 9.4. It is almost in complete accordance with 6.3 except that virtual hosts logs the additional information the canonical ServerName of the server serving the request and the process ID of the child that serviced the request. However, it violates the majority of the recommendations. Some examples are:

- CIS NGINX 2.1.3 is violated as the gzip directive is configured to be on, while gzip compression is recommended to be disabled in CIS NGINX 2.1.3 as an defense-in-depth strategy to mitigate attacks such as BREACH. Disabling gzip compression comes with a price as this would reduce performance and bandwidth and thus this might not be a valuable option for every user. Still, it is beneficial for users to be aware of the potential problem with compression. A solution could be to by default have the parameter off, with a comment referring to material that discuss potential solutions and their trade-offs such as [182]. Then users can make their own educated decision.
- CIS NGINX 2.4.3 is violated as the keepalive_timeout directive have the parameter 65, while CIS NGINX 2.4.3 recommends to have a value larger than 0 but equal or lower than 10 to mitigate denial of service attacks.
- CIS NGINX 2.5.1 is violated as the server_tokens directive have the parameter on while CIS NGINX 2.5.1 recommends the parameter off to avoid displaying the Nginx version number and operating system version
- CIS NGINX 3.1 is violated as the log_format directive have the default configuration, while CIS NGINX 3.1 suggests to use detailed logging. It is also mentioned that the log format should be adapted to suit an organizations need. Of course one can not expect a Playbook to have a log format suitable for all, but even so it would be preferable if the logging level was more detailed.
- CIS NGINX 3.3 is violated as the logging level have the parameter warn and not info.
- CIS NGINX 5.2.2 is violated as the client_max_body_size directive have the parameter 64m. CIS NGINX 5.2.2 recommends to limit the size of the request body to prevent buffer overflow attacks and that the value should be set low enough to protect an application but high enough not to interfere with functionality and block legitimate request bodies. The parameter in CIS NGINX 5.2.2 is 100K, the default is 1m. 64m is a large value and should reasonably be lower to prevent from buffer overflow attacks.

5.5.2.4 harden_nginx by user juju4

This Playbook was the one with the most extensive secure configuration found for Nginx, however only downloaded 8 times. As with Apache, none of the Playbooks

Table 5.2: Summary of information covered by harden_nginx by user juju4 and not by CIS NGINX

ID	Subject
juju4-nginx-01	Set buffer size limitations to migate buffer overflow attacks
juju4-nginx-02	Instruct user agents to upgrade a priori insecure resource requests to secure transport before fetching them if they support the upgrade-insecure-requests CSP directive
juju4-nginx-03	Disable etag directive to avoid that remote attackers may be able to discern the inode number from returned values
juju4-nginx-04	Ensure usage of secure ECDHE cipher curves
juju4-nginx-05	Deny access to potentially sensitive files
juju4-nginx-06	Block vulnerability scanners and robots

related to Nginx with security in focus found had more than 100 downloads, thus this does not seem to be a priority by the users of Ansible Galaxy. It is still maintained and have 8+ commits since its founding in 2017 [94]. The configuration settings available in the Playbook sometimes have a rationale in form of a comment or a link to web page with relevant information, but in general this is not the case and the configuration settings lack rationale. The Playbook cover CIS Nginx 2.5.1, 4.1.6, 5.3.2, 5.2.3, 5.3.1, 5.3.3 It also violates some recommendations, differs in recommended arguments or cover recommendations that CIS Nginx does not. The ones that are missing can be seen in Table 5.2, and the ones that are violated/differ in recommended arguments are discussed below. The missing recommendations are given IDs in form of juju4-nginx-XX.

- As with the Playbook nginx by user geerlingguy in Chapter 5.5.2.3 CIS NGINX 2.1.3 is violated as the gzip directive is configured to be on. The same discussion found in this chapter applies here.
- CIS NGINX 2.4.3 is violated as the keepalive_timeout directive is set to 65, while CIS NGINX 2.4.3 recommends to have a value larger than 0 but equal or lower than 10 to mitigate denial of service attacks.
- harden_nginx configures the HTTP Strict-Transport-Security response header, which is explained more in detail at [123], as “add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";”. CIS NGINX 4.1.8 recommends to use a value for max-age of 15768000 seconds (six months) or longer, and CIS NGINX 4.1.12 recommends using includeSubDomains and preload as well. However, CIS NGINX 4.1.12 emphasize that this configuration requires careful consideration and informs the user of the consequences of using the parameters includeSubDomains, previously discussed in 5.5.2.2, and preload, which adds the domains and subdomains to

the HSTS preload list permanently (if all of the requirements is satisfied) which is a slow and painful process to undo [48]. harden_nginx does not mention this at all. As with configuring gzip, a solution could be to by default not configure includeSubDomains and preload, and have a comment referring to material that discuss the configurations and consequences such as [48] and CIS NGINX 4.1.12. Then users can make their own educated decision.

- harden_nginx configures the ssl_prefer_server_ciphers directive with parameter on and the ssl_ciphers directive with parameter EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH. CIS NGINX 4.1.5 recommends the same parameter for the ssl_prefer_server_ciphers directive and as mentioned in Chapter 4.2 there are four different suggestions for the ssl_prefer_server_ciphers directive. The configuration of harden_nginx is in accordance with all of them except for the one for "Mozilla modern profile web server" which have very specific cipher suits and modes. Thus, there are several secure options for the ssl_ciphers directive and one is available in harden_nginx. CIS NGINX 4.1.5

- harden_nginx configures add_header Content-Security-Policy "default-src 'none'; script-src 'self'; connect-src 'self'; img-src 'self'; style-src 'self'; upgrade-insecure-requests;", i.e. the default policy for loading content is that it is prevented from being loading from any source, valid sources for loading JavaScript, XMLHttpRequest (AJAX), WebSocket, EventSource, images and stylesheets are from the same origin(same scheme, host and port) [101] and that user agents shall treat all of a site's insecure URLs (those served over HTTP) as though they have been replaced with secure URLs(an option intended for web sites with large numbers of insecure legacy URLs that need to be rewritten) [120]. CIS NGINX 5.3.4 recommends "add_header Content-Security-Policy "default-src 'self'";", i.e. the default policy for loading content is that it is allowed to be loaded from the same origin(same scheme, host and port). As harden_nginx by default does not allow loading resources unless specified (there are 15 types of resources that can be controlled through the Content-Security-Policy HTTP response [101]), this is a more strict and controlled configuration, which should reasonably minimize the attack vector of the web server and thus be a more beneficial configuration.

Some content of harden_nginx are not mentioned above they do not directly effect the vulnerability status of a web server. These are:

- Directive ssl_session_cache with parameters shared:SSL:10m and directive ssl_session_timeout with parameter 10m. No secure rationale behind this was described in the configuration or found in other sources of information. The reason why this configuration is set seems to be to increase performance by reusing SSL session parameters through a cache shared between all worker processes to avoid SSL handshakes for parallel and subsequent connections. If session parameters are to be reused, reasonable they should be valid for the

duration of a user session and not too long of a period to minimize impact of potentially compromised master secrets. The default of ssl_session_timeout is five minutes and ssl_session_cache none [130], i.e. that Nginx tells a client that sessions may be reused, but does not actually store session parameters in the cache. Thus, this configuration does not seem to really decrease the attack vector of a web server.

- The configuration "add_header Referrer-Policy "strict-origin-when-cross-origin"" which implies to send a full URL for same-origin requests and only send the origin when the protocol security level stays the same (HTTPS to HTTPS), and send no header to a less secure destination (HTTPS to HTTP). This might protect sensitive client information but not directly affect the security of the server [122].
- Configuration related to OSCP stapling as this is related to the security of the client not the the web server [111].
- Excluding logs for missing favicon.ico and robots.txt as this does not directly affect the vulnerability of a web server. One could argue that to facilitate as useful analysis of logs as possible in case of attacks or other suspicious activities, the log data should only include relevant information and this might not include logs for misisng favicon.ico and robots.txt. Still, the exclusion of the logs should not have a significant impact on the vulnerability of a web server.
- Counteracting referrer spam. This is not a desirable phenomenon, see more in e.g. [176], but it does not affect the security of a web server.

5.5.3 Using Ansible

Ansible and the examined ready to use content works without any errors and is fairly straight forward to set up as it has a agentless architecture. As with Puppet, the output from running a module or Playbook is not very informative. The output of what action has been made is a bit more descriptive than Puppet, see Figure 5.10 for an example, but still specific configuration decisions and rationale behind these are not presented to the user. As with Puppet, the rationale behind configuration decisions could be added as a comment in the code of a module or Playbook, but that does not enable a very clean or consistent practice to educate users. There are no comments with rationale found in any of the examined content.

5.6 Summary of Tools

OpenSCAP

- It supports the operating systems Fedora, RHEL 6, RHEL7, CentOS 6 and CentOS 7, Debian, Ubuntu and Windows.
- It is released under GNU Lesser General Public License v2.1.
- It does not need to be installed onto the device it examines.

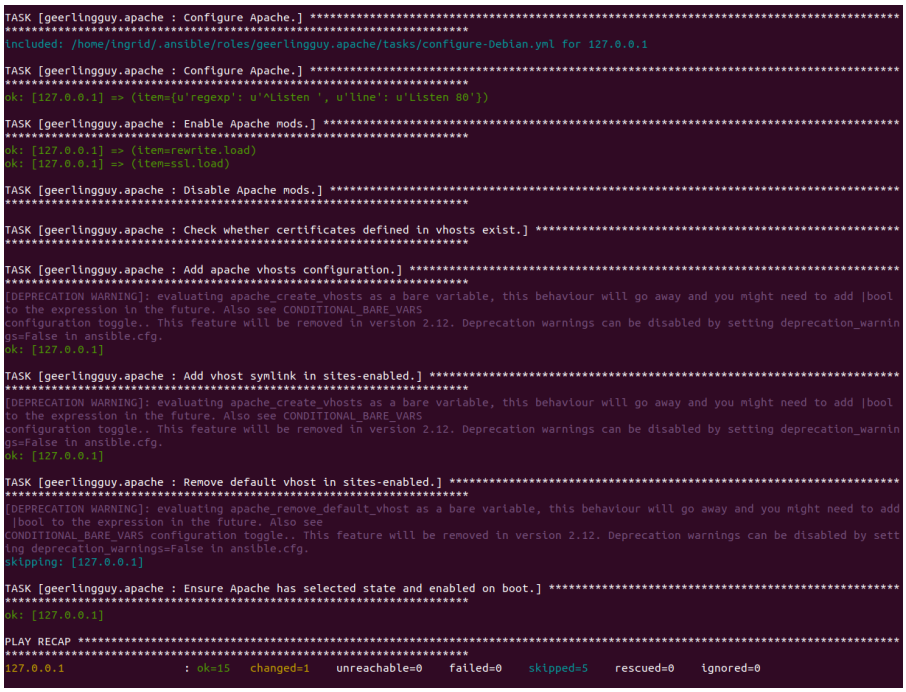

Figure 5.10: Example of output when running Ansible

- It supports Apache and Nginx, as well as other types of software.
- It does not have the possibility to remediate misconfiguration found.
- It works without problems.
- It offer evaluation and/or configuration that correspond to secure best practice, but it depends on the content that the user provides. No content representative of the CIS Benchmark recommendations was found.
- There are no type of scoring or ranking of the configuration parameters.
- The user is not precisely educated about configuration suggestions, but there is a possibility to reference to related sources of information.

CFEngine

- It supports the operating systems CentOS/RHEL, Debian or Ubuntu for Policy Servers and AIX, CentOS/RHEL, Debian, HP-UX, Solaris, Ubuntu, Windows for Clients.
- It is released under GNU General Public License, version 3.
- It needs to be installed onto the device it examines.
- It supports both Apache and Nginx, as well as other types of software, but there is no ready to use content for Nginx.

- It has the possibility to remediate misconfiguration found.
- It does not works without problems.
- The ready to use content available does not offer evaluation and/or configuration that correspond to secure best practice for web servers.
- There is no type of scoring or ranking of the configuration parameters, it could however be added as a comment in the promises.
- There is not really any beneficial alternative to educate the user about configuration suggestions, the best alternative is specifying them as comments in configuration files.

Chef Inspec

- It supports the operating systems Red Hat Enterprise Linux, mac OS, SUSE Linux Enterprise Server, Ubuntu and Windows.
- It is released under Apache 2.0 license.
- It does not need to be installed onto the device it examines.
- It supports and there is ready to use content for both Apache and Nginx, as well as other types of software.
- It does not have the possibility to remediate misconfiguration found.
- The tool works without problem, but the ready to use content does not.
- The ready to use content available offers evaluation and/or configuration that correspond to secure best practice for web servers, although not very extensive. It has configuration related to security not mentioned in the CIS Benchmarks.
- There is a type of scoring or ranking of the configuration parameters, but it is not presented directly in the terminal.
- The is a possibility to educate the user about configuration suggestions, although this could be improved in the ready to use content.

Puppet

- It supports the operating systems Debian, Fedora, macOS, Windows, Red Hat Enterprise Linux, SUSE Linux Enterprise Server and Ubuntu.
- It is released under Apache License 2.0.
- It needs to be installed onto the device it examines.
- It supports and there is ready to use content for both Apache and Nginx, as well as other types of software.
- It has the possibility to remediate misconfiguration found.
- The tool and the ready to use content works without problems.

- The ready to use content available does not offer evaluation and/or configuration that correspond to secure best practice for web servers.
- There is no type of scoring or ranking of the configuration parameters.
- There is not really any beneficial alternative to educate the user about configuration suggestions, the best alternative is specifying them as comments in configuration files.

Ansible

- It can be run from any machine with Python 2 or 3 installed, but Windows is not supported for the node to be controlled.
- It is released under GNU General Public License v3.0.
- It does not need to be installed onto the device it examines.
- It supports and there is ready to use content for both Apache and Nginx, as well as other types of software.
- It has the possibility to remediate misconfiguration found.
- The tool and the ready to use content works without problems.
- The ready to use content available offers evaluation and/or configuration that correspond to secure best practice for web servers, although not thoroughly. It has configuration related to security not mentioned in the CIS Benchmarks.
- There is no scoring or ranking of the configuration parameters.
- There is not really any beneficial alternative to educate the user about configuration suggestions, the best alternative is specifying them as comments in configuration files.

Chapter 6
Implementation

This chapter describes the implementation of new content for verifying absence of security misconfiguration and the results from the analysis of the implementation.

6.1 Categorization and Chosen Recommendations

The categorization of the configurations related to security misconfiguration is found in table Table 6.1 and Table 6.2. The verification tests implemented in each category is in bold. Recommendations from CIS are in numbers, e.g. 2.2. Recommendations from OWASP not covered in CIS are on the form of OWASP-xx, e.g. OWASP-10. Recommendations from DISA STIG not covered in CIS are on the form of U1-xxxx or U2-xxxx, e.g. U1-0010. Recommendations from content for Chef Inspec not covered in CIS are on the form of apache-xx or nginx-xx, e.g. apache-08. Recommendations from content for Ansible not covered in CIS are on the form of juju4-apache-xx or juju4-nginx-xx, e.g. juju4-apache-01. Some of the controls of the implementation is currently only usable for Debian or Ubuntu systems as some commands used in the tests are operating system specific, e.g. using the package manager tool, but they can be adapted to suit other operating systems.

Table 6.1: Categorization of the Apache recommendations

Category	**Recommendations**
Verify That Module Is Enabled	2.2 6.6 **7.1**, U1-0010, U1-0510, U1-0520, U2-0380
Verify That Module Is Disabled	2.1, 2.3, 2.4, 2.5, **2.6**, 2.7, 2.8, OWASP-10, apache-08
Verify That Apache Web Server Run as a Non-Root User	**3.1**
Verify That the Apache User Account Has an Invalid Shell	**3.2**
Verify That the Apache User Account Is Locked	**3.3**

Continued on next page

Table 6.1 – *Continued from previous page*

Category	**Recommendations**
Verify Correct Ownership	**3.4**, 3.5, 7.3, U1-0190, U1-0820, U1-0900
Verify Correct Permissions	**3.6**, 3.11, 3.12, 7.3, OWASP-02, U1-0180, U1-0820, U1-0900
Verify Non-Existance of Directive or Verify Correct Configuration	3.7, **3.8**, 3.10, 7.6, 7.7, 8.4
Verify That the Apache Process ID (PID) File Is Secured	**3.9**
Find Directive, Verify Existance or Non-Existance of Nested Directive and Its Value	4.1, 4.2, **4.3**, 5.1, 5.2, 5.3, 5.7, 5.10, 5.11, U1-0670, juju4-apache-01, juju4-apache-02, juju4-apache-03
Verify Non-Existance of Directive or Text	**4.3**, OWASP-12, U1-0240
Verify Value of Directive	4.4, 5.13, 5.14, 6.1, 6.2, 6.3, 7.4, 7.5, 7.8, 7.9, 7.10, 8.1, **8.2**, 9.1, 9.2, 9.3, 9.4, 9.5, 10.1, 10.2, 10.3, 10.4, OWASP-01, OWASP-06, OWASP-08, OWASP-10, OWASP-11, U1-0300, U1-0460, U1-0470, U1-0510, U2-0380, juju4-apache-04, juju4-apache-05, juju4-apache-06, juju4-apache-07, juju4-apache-08, juju4-apache-09
Verify That Default Content Is Removed	**5.4**
Verify That Default CGI Content Is Removed	**5.5**, 5.6
Verify Directive Exists on Server Level and Verify Its Value	5.8, 5.9, **5.12**, 7.11
Verify Correct Setting of Log Storage and Rotation	**6.4**
Verify That Applicable Patches Are Applied	**6.5**
Verify That a Valid Trusted Certificate Is Installed	**7.2**
Verify That a Default Hosted Application Web Page Is Displayed When a Requested Web Page Cannot Be Found	**U2-0620**

Table 6.2: Categorization of the Nginx recommendations

Category	Recommendations
Verify That Nginx Is Installed	**1.1.1**
Verify That the Latest Software Package Is Installed	**1.2.2**
Verify Non-Existence of Module	**2.1.1, 2.1.2, 2.1.3, 2.1.4**
Verify That Nginx Use a Non-Privileged and Dedicated Service Account	**2.2.1**
Verify That the Nginx Service Account Is Locked	**2.2.2**
Verify That the Nginx Service Account Has an Invalid Shell	**2.2.3**
Verify Correct Ownership	**2.3.1**
Verify Correct Permission	**2.3.2**, 4.1.3, OWASP-02
Verify That the Nginx Process ID (PID) File Is Secured	**2.3.3**
Verify Non-Existence of Directive or Existence of Correct Configuration	**2.3.4**
Verify Non-Existence of Directive or Text	**OWASP-12**
Verify Existence of Directive and Verify Its Value	**3.3**
Verify Existence of Directive Inside a Context and Verify Its Value	2.4.1, 2.4.2, **2.5.1**, **2.5.3**, 3.1, 3.2, 3.5, 3.6, 4.1.1, 4.1.4, 4.1.5, 4.1.7, 4.1.9, 4.1.13, 4.1.14, 5.1.1, 5.1.2, 5.2.1, 5.2.2, 5.2.3, 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.5, OWASP-06, OWASP-07, OWASP-09, nginx-06, nginx-16, juju4-nginx-01, juju4-nginx-02, juju4-nginx-03, juju4-nginx-04, juju4-nginx-05, juju4-nginx-05, juju4-nginx-06
Verify Existence of Directive Inside a Context and Verify Correct Configuration of the Value	4.1.2, **4.1.6**
Verify Existence of Directives and Nested Directives Inside a Context and Verify Their Values	5.2.4, **5.2.5**
Verify Existence of Directive Inside a Context and That Its Value Is Lower/Higher than Threshold	**2.4.3**, 2.4.4, 4.1.8

Continued on next page

Table 6.2 – *Continued from previous page*

Category	Recommendations
Verify That Default Content Is Disabled	**2.5.2**
Verify That Log Files Are Rotated	**3.4**
Verify Correct Configuration Through HTTP Request	**4.1.12**

The following recommendations was not included:

- CIS Apache 1.1, 1.2, 1.3 as they do not cover proper configuration of Apache but subjects such as pre-installation checklists and how to install Apache.
- CIS NGINX 1.1.2 where the audit is the same as in 1.1.1 but no way of verifying the the main purpose, ensuring that Nginx is installed from source, was found. This was not found for regular terminal usage or in Chef Inspec.
- CIS NGINX 1.2.1 is about configuring the package manager repositories and thus not strictly related to the configuration of the Nginx server.
- CIS NGINX 2.5.4, 3.7 4.1.10, 4.1.11 is not related to web servers but only proxy and loadbalancer.

6.2 Result of Implementation

Almost every category was possible to implement. The only subject that caused trouble was related to variables set in the so called envvars file in Apache. As example, the user of Apache is usually not set directly in the main configuration file apache.conf but rather, its parameter is ${APACHE_RUN_USER}. This is a reference to the Apache envvars file, located at `/etc/apache2/envvars` for Debian and Ubuntu systens, containing the variable APACHE_RUN_USER=www-data (this was e.g. the problem for control apache-06 in Chapter 5.3.2.2). Some of variables in the envvars file, such as the user, can be found through other commands. However, no command was found for finding the value of the variables APACHE_PID_FILE and APACHE_LOCK_DIR in envvars. In a regular terminal one can just use the command "source /etc/apache2/envvars" and then execute "echo $APACHE_PID_FILE" to obtain the value. This was however not possible to use inside Chef Inspec. This is probably because the resource named command, which is used to execute commands (see [35] for more information), does not change the state of a system but only gathers the result of the command.

Thus, the seeked variables inside the envvars file was obtained by simple extracting the content of the file. Unfortunately, their value can be on the form of /var/run/apache2$SUFFIX/apache2.pid or /var/lock/apache2$SUFFIX, where the value of $SUFFIX is decided by a bash command inside the envvarsfile. The value of $SUFFIX is usually a empty string, as it only has a value if there are multiple Apache instances running on the same machine. There was no way found to execute the bash command properly to obtain the value of $SUFFIX as there

was no way found to obtain the variables used inside the bash command. Due to this, for the tests where values inside the envvars file was needed where $SUFFIX was present, the value of $SUFFIX was assumed to be a empty string. Thus, these tests are only usable for situations where not multiple instances of Apache is on the same machine.

Another situation not entirely satisfactory is the output in the terminal when running the implementations in Chef Inspec. Some controls, e.g. the control covering CIS Apache 4.3, lead to that the terminal was filled with all the operating configuration for the web server, no matter if the control was successful or not. This creates a situation where the result of the validation is not very easy to survey since the terminal is filled with lots of information where all information is not relevant. The result is still valid and the output is not so overwhelming that it is unusable, but it would still be preferable if this was not the case.

Another subject worth mentioning is the possibility of successful validation if either one configuration or another configuration is present, described under "Advanced concepts" in [40]. This was attempted to use for verifying CIS NGINX 2.3.4, where either a directory should not be configured or it should be validated that there is presence of other correct configuration if the directive is present. Unfortunately, the method described in "Advanced concepts" only allows two tests, while the audit in CIS NGINX 2.3.4 includes four tests in total (one for validating that a directive is not present, and the other three to validate other correct configuration). Thus, if the directive is present this control will marked as failed as the tests ensuring that the directive is not present will fail, no matter if the other correct configuration is present. This can be worked around by using if conditions, e.g. only running the test that the directive is not present if the condition is met that it is not present and only running tests that validates if the other correct configuration present if the condition is met that the directive is present, but this is not a very elegant solution.

Chapter 7
Discussion

This chapter discuss the results from previous chapters.

In this thesis, the importance of the different security misconfigurations found has not been taken into account, i.e. there is no ranking. It could be the case that some sources of information or ready to use contents have only put forth security misconfigurations they believe is the most critical, making a trade off of comprehensiveness for usability. However, no source of information on ready to use content specified why the configuration put forth was chosen instead of other or how its importance was determined. Neither less, it would be favourable if the different security misconfigurations found was ranked, to make it more easy to review but also perhaps creating a foundation for more usable results to incorporate into the tools. As mentioned in Chapter 6, the output in the terminal was a bit overwhelming due to that for some control all the operating configuration was printed. Even if this was not the case, there could still arise a situation where result of the validation is not very easy to survey. If one would implement controls and tests for validating the absence of all security misconfiguration found in this thesis, e.g. the more than 120 in total for Apache, this is quite a lot of controls and tests which will be presented in the terminal of the user. This could create a solution which users do not find convenient and thus not inclined to use, leading to a situation where security misconfiguration will continue to go unnoticed. Validation for only a subset, preferably only the most critical security misconfigrations, could be made possible if a ranking of the security misconfiguration was made.

Another subject worth mentioning is that Chef Inspec do provide the possibility to orderly specify the impact and rationale behind each control, which allow user to understand the importance and background of the configuration covered in this control. However, if a user have interest in either of these two, he has to manually visit the file where the controls are written as this information is not presented with the result in the terminal when running Chef Inspec. This is not a very burdensome activity to perform, but it would still be neat to have all relevant information in one place. It could be beneficial to adapt Chef Inspec and take inspiration from OpenSCAP and make it possible to generate the result of the validation as an HTML file or to create a user interface to make the result more easy to survey. This solution could also include the impact and description belonging to each control.

Furthermore, when configuration which only related to protecting the client but not affecting the vulnerability of the web server was found, it was not presented

in any of the tables or the categorization made for the implementation. As the vulnerabilities of web servers was the focus of the thesis, this is not irrational, but most organizations would probably value the security of client using their servers highly as well. Thus, if the results of this thesis were to be used in a real case, it is reasonable to believe that users would like to validate absence of this kind of security misconfiguration as well.

Finally, it might be the case that users do not value impact, rationale and education of why certain configuration should or should not be present to avoid security misconfiguration and that they would much more prefer ready to use content for deployment tools such as Puppet or Ansible. If this is the case, results of this thesis is still valuable, as it highlights that there is a lot of configuration related to security misconfiguration missing in the existing ready to use content. In addition, it summarizes, hopefully the large majority, of the relevant configuration to implement, where it was found and problems that can arise when implementing some of them.

Chapter 8

Conclusion and Future Research

This chapter summarizes the thesis and presents suggestions for relevant related research.

8.1 Summary of Results and Conclusions

The thesis has provided several insights. Firstly, no source of information was found that completely covered all identified possible security misconfigurations of Apache and Nginx. This is at glance quite a frustrating discovery. If a person has recognized the need for secure configuration and seek to gain knowledge of how to implement this, he would reasonably like to have this information gathered in one place. It is a time consuming task to compare different sources of information. However, it could be that a too comprehensive source of information might be so exhaustive that the reader does not have the possibility to benefit all knowledge and take action. As an example, the total amount of possible security misconfigurations of Apache found are more than 120 when summarized. Learning about and configuring these is not unimaginable but it is not a straight forward task either.

Another insight is that there is not that many satisfying tools with substantial ready to use content for validating absence of security misconfiguration. Perhaps the proprietary solutions such as CIS-KAT and Nessus cover this area, but taking the strong movement of open source solutions today into account, this is a bit surprising. Even when there was ready to use content with quite extensive configuration against security misconfiguration, such as in Ansible Galaxy, these were not popular with the users. This could be a sign of that many of the users of Apache or Nginx does not validate that their configuration does not create vulnerabilities. Another explanation might be that they do it manually, not a very efficient solution, or that they have their own scripts to do validation. None of these method take advantage of or contribute to the knowledge of the community and might lead to both inefficient and not fully covering solutions.

It is also interesting to notice that rationale behind and references to more information about the configuration endorsed in the ready to use content is in general very unsatisfying. The reason behind and the possible options of the configuration is not always straight forward. Take for example the discussion about gzip compression in Chapter 5.5.2.3 and enabling preload in Chapter 5.5.2.4.

The problems that make these configuration related to vulnerability issues is not obvious and a user could clearly benefit information of why or why not these configurations should be made and what consequences the taken action could have. Of course this is something that a user could research about on his own, but given how many configuration options there are, it would decrease effort and time needed if there was more description available.

At last, the thesis provided the result that Chef Inspec is a feasible solution to use to validate absence of every security misconfiguration found, though for Apache only for cases when Apache is run as a single instance. It was also shown that improvements can be made regarding the presentation of the result from the validation to increase the ease of survey and to make use of the feautres of impact and rationale that Chef Inspec provide.

8.2 Thoughts on future research

The thesis showed that there is a vast amount of configuration related to security misconfiguration of web servers. It could be beneficial to have a clear ranking of the importance of the different configurations. This could benefit users in several ways. It might be that the amount of configuration to take into account is too extensive for the average user, and that they would only have interest in the most critical. It could also be the case that a user would like to implement all, but that this process would require too much time to be implemented at once and thus requiring partitioning and prioritization of the configuration to allow for implementation in different phases. Thus, research with the target of enabling a clear ranking and classification of the different security configurations could benefit the community.

Another relevant subject is taking the users perspective and experience into account. How does the validation of web server configuration take place today? Do they perform it at all, do they use any supporting software and how do they find information about security misconfiguration? How would the most favourable supporting software function? Would they like to have validation of only the most critical configuration? How much information and rationale behind each option do they need? Are they in accordance with that validation and deployment should be separated or do they just want to use ready to use content for deployment tools such as Puppet and Ansible to utilize a solution as efficient as possible? This research is important to understand why or why not validation is made and how to improve existing solutions or perhaps develop novel ones.

It could also be useful to broaden the scope to include other types of software such as firewalls. What configuration is really needed to be present in a web server if there is presence of a firewall? What possibilities exists to obtain information about their configuration and put them in relation to the configuration of the web server? Is the placement of certain protection for web servers more efficient in a firewall? This could be beneficial research as in many real world systems the web server is seldom alone but have surrounding software which can affect their vulnerability.

References

[1] Result from openscap from evaluating apache. https://github.com/inghylt/Check-yourself-thesis/blob/master/OpenSCAP-result.xml. Accessed: 2019-05-31.

[2] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 5–17, New York, NY, USA, 2015. ACM.

[3] ALTEX-SOFT. About OVALdb. https://ovaldb.altx-soft.ru/AboutOVALdb.aspx. Accessed: 2019-03-29.

[4] Northern.tech AS. Automate and manage your IT infrastructure. https://cfengine.com/product/. Accessed: 2019-04-09.

[5] Northern.tech AS. CFEngine Community. https://cfengine.com/product/community/. Accessed: 2019-04-09.

[6] Northern.tech AS. cfengine slash design-center. https://github.com/cfengine/design-center/tree/9f58be95d35edef9323c91091c2916e67cd26c64/sketches. Accessed: 2019-04-09.

[7] Northern.tech AS. Classes and Decisions. https://docs.cfengine.com/docs/3.13/reference-language-concepts-classes.html. Accessed: 2019-04-09.

[8] Northern.tech AS. Command Line Sketches. https://docs.cfengine.com/docs/3.10/guide-design-center-configure-sketches-community.html. Accessed: 2019-04-10.

[9] Northern.tech AS. Create, Modify, and Delete Files. https://docs.cfengine.com/docs/3.7/examples-tutorials-files-tutorial.html. Accessed: 2019-04-09.

[10] Northern.tech AS. design-center. https://github.com/cfengine/design-center/tree/604216137c324621c2ad437f5ee9b5b4f756c050. Accessed: 2019-04-10.

[11] Northern.tech AS. Design Center Overview. https://docs.cfengine.com/docs/3.10/guide-design-center.html. Accessed: 2019-04-09.

[12] Northern.tech AS. Design Center Sketches Web Servers. https://github.com/cfengine/design-center/tree/6705cfa22bda53e8fc8630cf5e985217df48e642/sketches/web_servers. Accessed: 2019-04-09.

[13] Northern.tech AS. design-center slash sketches slash web underscore servers slash apache slash templates slash debian slash default-ssl.tmpl. https://github.com/cfengine/design-center/blob/a7b930f1e9839f0571ccc38b34aa858fd7afdeb3/sketches/web_servers/apache/templates/debian/default-ssl.tmpl. Accessed: 2019-04-09.

[14] Northern.tech AS. design-center slash sketches slash web underscore servers slash apache slash templates slash debian slash default dot tmpl. https://github.com/cfengine/design-center/blob/a7b930f1e9839f0571ccc38b34aa858fd7afdeb3/sketches/web_servers/apache/templates/debian/default.tmpl. Accessed: 2019-04-09.

[15] Northern.tech AS. design-center/howto/etch_a_sketch.md. https://github.com/cfengine/design-center/blob/604216137c324621c2ad437f5ee9b5b4f756c050/howto/etch_a_sketch.md. Accessed: 2019-04-10.

[16] Northern.tech AS. design-center/sketches/web_servers/apache. https://github.com/cfengine/design-center/blob/a7b930f1e9839f0571ccc38b34aa858fd7afdeb3/sketches/web_servers/apache/. Accessed: 2019-04-10.

[17] Northern.tech AS. Examples and Tutorials. https://docs.cfengine.com/docs/3.7/examples.html#tutorial-for-running-examples. Accessed: 2019-04-09.

[18] Northern.tech AS. Promise Types and Attributes. https://docs.cfengine.com/docs/3.7/reference-promise-types.html. Accessed: 2019-04-09.

[19] Northern.tech AS. Promises. https://docs.cfengine.com/docs/3.7/guide-language-concepts-promises.html. Accessed: 2019-04-09.

[20] Northern.tech AS. Supported Platforms and Versions. https://docs.cfengine.com/docs/3.13/guide-latest-release-supported-platforms.html. Accessed: 2019-05-31.

[21] Martin Barrere. *Vulnerability Management for Safe Configurations in Autonomic Networks and Systems*. Theses, Université de Lorraine, June 2014.

[22] Salman Baset, Sahil Suneja, Nilton Bila, Ozan Tuncer, and Canturk Isci. Usable Declarative Configuration Specification and Validation for Applications, Systems, and Cloud. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track*, Middleware '17, pages 29–35, New York, NY, USA, 2017. ACM.

[23] Inc. (CIS) Center for Internet Security. CIS Apache HTTP Server 2.4 Benchmark v1.4.0. `https://www.cisecurity.org/cis-benchmarks/`. Accessed: 2019-04-09.

[24] Inc. (CIS) Center for Internet Security. Cis Controls v7.1. `https://www.cisecurity.org/controls/`. Accessed: 2019-03-19.

[25] Inc. (CIS) Center for Internet Security. CIS NGINX Benchmark v1.0.0 - 02-28-2019. `https://www.cisecurity.org/cis-benchmarks/`. Accessed: 2019-03-01.

[26] Inc. (CIS) Center for Internet Security. OVALRepo/commits/master. `https://github.com/CISecurity/OVALRepo/commits/master`. Accessed: 2019-04-02.

[27] Center for Internet Security, Inc. (CIS). About Us. `https://www.cisecurity.org/about-us/`. Accessed: 2019-03-19.

[28] Center for Internet Security, Inc. (CIS). CIS Benchmarks™ FAQ. `https://www.cisecurity.org/cis-benchmarks/cis-benchmarks-faq/`. Accessed: 2019-03-18.

[29] Center for Internet Security, Inc. (CIS). Download Our Free Benchmark PDFs. `https://learn.cisecurity.org/benchmarks`. Accessed: 2019-03-19.

[30] Chef. Apache DISA STIG. `https://supermarket.chef.io/tools/apache-disa-stig`. Accessed: 2019-04-12.

[31] Chef. CHEF INSPEC. `https://www.chef.io/products/chef-inspec/`. Accessed: 2019-04-11.

[32] Chef. Chef InSpec 3.9.0. `https://downloads.chef.io/inspec`. Accessed: 2019-04-12.

[33] Chef. Chef InSpec 3.9.0. `https://downloads.chef.io/inspec/stable/3.9.0`. Accessed: 2019-04-11.

[34] Chef. Chef Inspec Glossary. `https://www.inspec.io/docs/reference/glossary/`. Accessed: 2019-06-17.

[35] Chef. command. `https://www.inspec.io/docs/reference/resources/command/`. Accessed: 2019-07-15.

[36] Chef. Devsec Apache Baseline. `https://supermarket.chef.io/tools/apache-baseline`. Accessed: 2019-04-12.

[37] Chef. Devsec Nginx Baseline. `https://supermarket.chef.io/tools/nginx-baseline`. Accessed: 2019-04-15.

[38] Chef. InSpec. https://www.chef.io/wp-content/uploads/2018/05/chef-inspec-datasheet-2016.pdf. Accessed: 2019-04-15.

[39] Chef. Inspec documentation. https://www.inspec.io/docs/. Accessed: 2019-04-15.

[40] Chef. InSpec DSL. https://www.inspec.io/docs/reference/dsl_inspec/. Accessed: 2019-04-12.

[41] Chef. Inspec Resources Reference. https://www.inspec.io/docs/reference/resources/. Accessed: 2019-04-12.

[42] Chef. Introducing the New Chef: 100% Open, Always. https://blog.chef.io/2019/04/02/chef-software-announces-the-enterprise-automation-stack/. Accessed: 2019-04-11.

[43] Chef. myApacheTest. https://supermarket.chef.io/tools/myapachetest. Accessed: 2019-04-12.

[44] Chef. Resource DSL. https://www.inspec.io/docs/reference/dsl_resource/. Accessed: 2019-04-12.

[45] Chef. Tools and Plugins. https://supermarket.chef.io/tools/. Accessed: 2019-04-11.

[46] Chef. Try Inspec. https://learn.chef.io/modules/try-inspec#/. Accessed: 2019-04-11.

[47] Chef. Why Chef and Continuous Automation. https://www.chef.io/why-chef/. Accessed: 2019-04-11.

[48] Chromium. HTTP Strict Transport Security (HSTS) preload list. https://hstspreload.org/. Accessed: 2019-07-13.

[49] The MITRE Corporation. CVE-2009-3555. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3555. Accessed: 2019-05-19.

[50] The MITRE Corporation. Oval Language Overview. https://oval.mitre.org/language/about/overview.html. Accessed: 2019-03-21.

[51] PCI Security Standard Council. MAINTAINING PAYMENT SECURITY. https://www.pcisecuritystandards.org/pci_security/maintaining_payment_security. Accessed: 2019-04-02.

[52] PCI Security Standards Council. PCI DSS v3.2.1 - May 2018. https://www.pcisecuritystandards.org/document_library. Accessed: 2019-04-02.

[53] Dareboost. Secure your Cookies (Secure and HttpOnly flags). https://blog.dareboost.com/en/2016/12/secure-cookies-secure-httponly-flags/. Accessed: 2019-05-24.

[54] Defense Information Systems Agency (DISA). SRG / STIG Tools. https://public.cyber.mil/stigs/srg-stig-tools/. Accessed: 2019-07-01.

[55] Defense Information Systems Agency (DISA). STIGs Document Library. https://public.cyber.mil/stigs/downloads/. Accessed: 2019-07-01.

[56] DevSec. Devsec Project. https://dev-sec.io/project/. Accessed: 2019-04-12.

[57] DigiCert. FREAK Attack: What You Need to Know. https://www.digicert.com/blog/freak-attack-need-know/. Accessed: 2019-05-23.

[58] Defense Information Systems Agency (DISA). Security Technical Implementation Guides(STIGs). https://iase.disa.mil/stigs/Pages/index.aspx. Accessed: 2019-03-28.

[59] DSLReports. CIS Benchmark Score Details. http://www.dslreports.com/forum/r13885993-CIS-Benchmark-Score-Details. Accessed: 2019-04-16.

[60] Alexandre D'Hondt and Hussein Bahmad. Understanding SCAP Through a Simple Use Case. *Hakin9 IT Security Magazine*, 2016.

[61] Puppet Elizabeth Plumb. Starting with Puppet: Basics from a Puppet Labs Employee. https://puppet.com/blog/starting-puppet-basics-from-a-puppet-labs-employee. Accessed: 2019-06-12.

[62] B. Eshete, A. Villafiorita, and K. Weldemariam. Early Detection of Security Misconfiguration Vulnerabilities in Web Applications. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 169–174, Aug 2011.

[63] Center for Internet Security (CIS). CIS Named Top Workplace in 2019. https://www.cisecurity.org/blog/cis-named-top-workplace-in-2019/. Accessed: 2019-06-19.

[64] Center for Internet Security (CIS). Year in Review 2018. https://www.cisecurity.org/white-papers/2018-year-in-review/. Accessed: 2019-06-19.

[65] OpenSSL Software Foundation. ciphers. https://www.openssl.org/docs/man1.0.2/man1/ciphers.html. Accessed: 2019-05-15.

[66] OpenSSL Software Foundation. dhparam. https://www.openssl.org/docs/man1.0.2/man1/dhparam.html. Accessed: 2019-05-24.

[67] OpenSSL Software Foundation. Openssl Security Advisory [11-Nov-2009]. https://www.openssl.org/news/secadv/20091111.txt. Accessed: 2019-05-20.

[68] The Apache Software Foundation. APACHE HTTP SERVER PROJECT. https://httpd.apache.org/. Accessed: 2019-06-18.

[69] The Apache Software Foundation. Apache Module mod_cgi. http://httpd.apache.org/docs/current/mod/mod_cgi.html. Accessed: 2019-04-25.

[70] The Apache Software Foundation. Apache Module mod_cgid. https://httpd.apache.org/docs/2.4/mod/mod_cgid.html. Accessed: 2019-04-25.

[71] The Apache Software Foundation. Apache Module mod_include. https://httpd.apache.org/docs/2.4/mod/mod_include.html. Accessed: 2019-04-25.

[72] The Apache Software Foundation. Apache Module mod_ssl. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html#page-header. Accessed: 2019-05-20.

[73] The Apache Software Foundation. Configuration Files. https://httpd.apache.org/docs/trunk/configuring.html. Accessed: 2019-06-18.

[74] The Apache Software Foundation. Getting Started. https://httpd.apache.org/docs/trunk/getting-started.html. Accessed: 2019-06-18.

[75] Defense Information Systems Agency Field Security Operations (DISA FSO). STIG Transformation to XCCDF. https://www.fbiic.gov/public/2011/sep/U_STIG%20Transition%20to%20XCCDF%20FAQ%2020100126.pdf, January 2010.

[76] Inc. GitHub.

[77] Inc. GitHub. apache-baseline commits. https://github.com/dev-sec/apache-baseline/commits/109bb86d13b78992122b6461d1b5120db4357fc2. Accessed: 2019-04-12.

[78] Inc. GitHub. apache-baseline inspect.yml. https://github.com/dev-sec/apache-baseline/blob/937570b21ef9dc4327da2e78882868e18f0850ed/inspec.yml. Accessed: 2019-04-12.

[79] Inc. GitHub. apache resource: document and deprecate 2494. https://github.com/inspec/inspec/pull/2494. Accessed: 2019-04-12.

[80] Inc. GitHub. apache_spec.rb. https://github.com/dev-sec/apache-baseline/blob/e6b111b7fea8e346018fc6af815d776c9892b895/controls/apache_spec.rb. Accessed: 2019-04-15.

[81] Inc. GitHub. Devsec Apache Baseline - InSpec Profile. https://github.com/dev-sec/apache-baseline/tree/e6b111b7fea8e346018fc6af815d776c9892b895. Accessed: 2019-04-25.

[82] Inc. GitHub. inspec-stig-apache Fixed failing test. https://github.com/inspec-stigs/inspec-stig-apache/commit/53fb8eb5f1856d9b06ecced47791bad61d00ffb2. Accessed: 2019-04-12.

[83] Inc. GitHub. inspec-stig-apache inspec.yml. https://github.com/inspec-stigs/inspec-stig-apache/blob/c7a807cfd3ec6a64de1f7b49bd3df3cd3e75b88f/inspec.yml. Accessed: 2019-04-12.

[84] Inc. GitHub. inspec_training_courses. https://github.com/mitre/inspec_training_courses/blob/09deabeb95f3ca6c55952b11ba70ef02cd718be2/InSpec%20102%20Dev/InSpec102.md. Accessed: 2019-04-12.

[85] Inc. GitHub. inspec.yml. https://github.com/dev-sec/nginx-baseline/blob/659171fe2654d5ba3c2ce3d8f4d58f972ad9e0d7/inspec.yml. Accessed: 2019-04-15.

[86] Inc. GitHub. myApacheTest example.rb. https://github.com/dennispetrillo/myApacheTest/blob/8949e6c3097192adf1372f821f502c2deeeb5801/controls/example.rb. Accessed: 2019-04-12.

[87] Inc. GitHub. myApacheTest example.rb. https://github.com/inspec-stigs/inspec-stig-apache/tree/c7a807cfd3ec6a64de1f7b49bd3df3cd3e75b88f. Accessed: 2019-04-12.

[88] Inc. GitHub. nginx-baseline. https://github.com/dev-sec/nginx-baseline/commits/a23b568250377cc0d2ccad2c1f8a826edd92bb3d?after=a23b568250377cc0d2ccad2c1f8a826edd92bb3d+34. Accessed: 2019-04-15.

[89] Inc. GitHub. nginx_spec.rb. https://github.com/dev-sec/nginx-baseline/blob/fc4323712a9b8c25d0445906360600bbcce7e5e8/controls/nginx_spec.rb. Accessed: 2019-04-15.

[90] Inc. GitHub. puppetlicense. https://github.com/puppetlabs/puppet/blob/258edce755588387f5916898405601ea4e86c8f5/LICENSE. Accessed: 2019-06-11.

[91] GitHub, Inc. geerlingguy/ansible-role-apache. https://github.com/geerlingguy/ansible-role-apache/commits/e71d768f215a9ac08436ad666111016b7431f30e?before=e71d768f215a9ac08436ad666111016b7431f30e+35. Accessed: 2019-07-14.

[92] GitHub, Inc. geerlingguy/ansible-role-nginx. https://github.com/geerlingguy/ansible-role-nginx/commits/79650c5886e0825b317e81be029d3fecb7a1c0f0?before=79650c5886e0825b317e81be029d3fecb7a1c0f0+70. Accessed: 2019-07-14.

[93] GitHub, Inc. juju4/ansible-harden-apache. https://github.com/juju4/ansible-harden-apache/commits/ab684db7fe797f47139b178f183ee705959900b8?after=ab684db7fe797f47139b178f183ee705959900b8+104. Accessed: 2019-07-14.

[94] GitHub, Inc. juju4/ansible-harden-nginx. https://github.com/juju4/ansible-harden-nginx/commits/d1ccfd391a02813158931ececa41265277409721?before=d1ccfd391a02813158931ececa41265277409721+35. Accessed: 2019-07-14.

[95] GitHub, Inc. puppetlabs/puppetlabs-apache. https://github.com/puppetlabs/puppetlabs-apache/commits/5ff4ea101c3f3e7aa52aff4e4adc23e91f466222. Accessed: 2019-07-10.

[96] GitHub, Inc. voxpupuli/puppet-nginx. https://github.com/voxpupuli/puppet-nginx/commits/f3aaa06556ca9f2f461be764c55b28262c8e74b1. Accessed: 2019-07-10.

[97] Google Developers. Robots meta tag and X-Robots-Tag HTTP header specifications. https://developers.google.com/search/reference/robots_meta_tag. Accessed: 2019-07-11.

[98] Red Hat. OVAL definitions for Red Hat Enterprise Linux 3 and above. https://www.redhat.com/security/data/oval/. Accessed: 2019-03-28.

[99] Thomas Hühn. Myths about /dev/urandom. https://www.2uo.de/myths-about-urandom/. Accessed: 2019-05-20.

[100] Internet Engineering Task Force (IETF). Transport Layer Security (TLS) Renegotiation Indication Extension. https://tools.ietf.org/html/rfc5746. Accessed: 2019-05-20.

[101] Foundeo Inc. Content Security Policy Reference. https://content-security-policy.com/#source_list. Accessed: 2019-05-24.

[102] NGINX Inc. Beginner's Guide. https://nginx.org/en/docs/beginners_guide.html. Accessed: 2019-06-18.

[103] NGINX Inc. CHANGES. http://nginx.org/en/CHANGES. Accessed: 2019-05-21.

[104] NGINX Inc. Creating NGINX Plus and NGINX Configuration Files. https://docs.nginx.com/nginx/admin-guide/basic-functionality/managing-configuration-files/#directives. Accessed: 2019-06-18.

[105] NGINX Inc. nginx: download. http://nginx.org/en/download.html. Accessed: 2019-06-18.

[106] NGINX Inc. What is NGINX? https://www.nginx.com/resources/glossary/nginx/. Accessed: 2019-06-18.

[107] Joval. Industry Leading Platform Standards Support. https://jovalcm.com/capabilities/platform-standards-support/. Accessed: 2019-03-29.

[108] Joval. Modular Extensible. https://jovalcm.com/capabilities/architecture-extensibility/. Accessed: 2019-03-29.

[109] Joval. Scan Anything from Anywhere. https://jovalcm.com/. Accessed: 2019-03-29.

[110] KeyCDN. How to Setup Nginx HTTP/2. https://www.keycdn.com/support/nginx-http2. Accessed: 2019-05-20.

[111] KeyCDN. OCSP Stapling. https://www.keycdn.com/support/ocsp-stapling. Accessed: 2019-07-13.

[112] KeyCDN. Understanding the Apache Access Log. https://www.keycdn.com/support/apache-access-log. Accessed: 2019-07-02.

[113] Ching-Huang Lin, Chih-Hao Chen, and Chi-Sung Laih. A study and implementation of vulnerability assessment and misconfiguration detection. pages 1252–1257, 12 2008.

[114] Netcraft Ltd. February 2019 Web Server Survey. https://news.netcraft.com/archives/2019/02/28/february-2019-web-server-survey.html. Accessed: 2019-02-24.

[115] Netcraft Ltd. June 2019 Web Server Survey. https://news.netcraft.com/archives/2019/. Accessed: 2019-06-18.

[116] SimilarTech Ltd. Top Web Server Technologies. https://www.similartech.com/categories/web-server. Accessed: 2019-02-24.

[117] N. Mendes, A. A. Neto, J. Durães, M. Vieira, and H. Madeira. Assessing and Comparing Security of Web Servers. In *2008 14th IEEE Pacific Rim International Symposium on Dependable Computing*, pages 313–322, Dec 2008.

[118] LLC Mitchell Anicas, DigitalOcean. Getting Started With Puppet Code: Manifests and Modules. https://www.digitalocean.com/community/tutorials/getting-started-with-puppet-code-manifests-and-modules. Accessed: 2019-06-12.

[119] MITRE. Open Vulnerability and Assessment Language. https://oval.mitre.org/. Accessed: 2019-03-29.

[120] Mozilla. CSP: upgrade-insecure-requests. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/upgrade-insecure-requests. Accessed: 2019-07-13.

[121] Mozilla. Keep-Alive. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Keep-Alive. Accessed: 2019-05-23.

[122] Mozilla. Referer header: privacy and security concerns. https://developer.mozilla.org/en-US/docs/Web/Security/Referer_header:_privacy_and_security_concerns. Accessed: 2019-07-11.

[123] Mozilla. Strict-Transport-Security. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security. Accessed: 2019-07-13.

[124] Mozilla. X-Frame-Options. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options. Accessed: 2019-07-11.

[125] National Institute of Standards and Technology (NIST). National Vulnerability Database. https://nvd.nist.gov/. Accessed: 2019-06-19.

[126] National Institute of Standards and Technology (NIST). NIST Mission, Vision, Core Competencies, and Core Values. https://www.nist.gov/about-nist/our-organization/mission-vision-values. Accessed: 2019-03-18.

[127] National Institute of Standards and Technology (NIST). Security Content Automation Protocol- FAQs. https://csrc.nist.gov/Projects/Security-Content-Automation-Protocol/faqs. Accessed: 2019-03-20.

[128] National Institute of Standars and Technology (NIST). National Institute of Standars and Technology (NIST). https://www.nist.gov/. Accessed: 2019-06-19.

[129] A. A. Neto, M. Vieira, and H. Madeira. An appraisal to assess the security of database configurations. In *2009 Second International Conference on Dependability*, pages 73–80, June 2009.

[130] NGINX Inc. Module ngx_http_ssl_module. http://nginx.org/en/docs/http/ngx_http_ssl_module.html. Accessed: 2019-07-12.

[131] NIST. National Checklist Program Repository. https://nvd.nist.gov/ncp/repository. Accessed: 2019-03-28.

[132] National Institute of Standars and Technology (NIST). Common Configuration Enumeration (CCE) Details. https://nvd.nist.gov/config/cce/index. Accessed: 2019-03-19.

[133] Open Web Application Security Project (OWASP). About The Open Web Application Security Project. https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project#Core_Purpose. Accessed: 2019-03-18.

[134] OpenSCAP. Choosing a Policy. https://www.open-scap.org/security-policies/choosing-policy/. Accessed: 2019-03-27.

[135] OpenSCAP. OpenSCAP BASE. https://www.open-scap.org/tools/openscap-base/. Accessed: 2019-03-27.

[136] OpenSCAP. OpenSCAP homepage. url{https://www.open-scap.org/. Accessed: 2019-03-27.

[137] OpenSCAP. SCAP Components. https://www.open-scap.org/features/scap-components/. Accessed: 2019-03-21.

[138] OpenSCAP. SCAP Security Guide. https://www.open-scap.org/security-policies/scap-security-guide/. Accessed: 2019-03-27.

[139] OpenSCAP. SCAP workbench. https://www.open-scap.org/tools/scap-workbench/. Accessed: 2019-03-27.

[140] Rolf Oppliger. *SSL and Tls: Theory and Practice, Second Edition*. Artech House, Inc., Norwood, MA, USA, 2nd edition, 2016.

[141] Open Web Application Security Project (OWASP). Category:OWASP Top Ten Project. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project. Accessed: 2019-05-15.

[142] Open Web Application Security Project (OWASP). Industry:Citations. https://www.owasp.org/index.php/Industry:Citations#National_.26_International_Legislation.2C_Standards.2C_Guidelines.2C_Committees_and_Industry_Codes_of_Practice. Accessed: 2019-06-19.

[143] Open Web Application Security Project (OWASP). OWASP Testing Guide v4 Table of Contents. https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents. Accessed: 2019-04-23.

[144] Open Web Application Security Project (OWASP). OWASP™ Foundation. https://www.owasp.org/index.php/Main_Page. Accessed: 2019-06-19.

[145] Open Web Application Security Project (OWASP). Test Application Platform Configuration (OTG-CONFIG-002). https://www.owasp.org/index.php/Test_Application_Platform_Configuration_(OTG-CONFIG-002). Accessed: 2019-04-23.

[146] Open Web Application Security Project (OWASP). Test File Permission (otg-config-009). https://www.owasp.org/index.php/Test_File_Permission_(OTG-CONFIG-009). Accessed: 2019-05-22.

[147] Open Web Application Security Project (OWASP). Testing for HTTP Verb Tampering (OTG-INPVAL-003). https://www.owasp.org/index.php/Testing_for_HTTP_Verb_Tampering_(OTG-INPVAL-003). Accessed: 2019-03-25.

[148] Open Web Application Security Project (OWASP). Testing for Sensitive information sent via unencrypted channels (otg-crypst-003). https://www.owasp.org/index.php/Testing_for_Sensitive_information_sent_via_unencrypted_channels_(OTG-CRYPST-003). Accessed: 2019-05-22.

[149] Open Web Application Security Project (OWASP). Testing for Weak Encryption (otg-crypst-004). https://www.owasp.org/index.php/Testing_for_Weak_Encryption_(OTG-CRYPST-004). Accessed: 2019-05-22.

[150] Open Web Application Security Project (OWASP). Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection (otg-crypst-001). https://www.owasp.org/index.php/Testing_for_Weak_SSL/TLS_Ciphers,_Insufficient_Transport_Layer_Protection_(OTG-CRYPST-001). Accessed: 2019-05-22.

[151] Open Web Application Security Project (OWASP). Testing Guide Frontispiece. https://www.owasp.org/index.php/Testing_Guide_Frontispiece. Accessed: 2019-04-23.

[152] Open Web Application Security Project (OWASP). Testing Guide Introduction. https://www.owasp.org/index.php/Testing_Guide_Introduction#The_OWASP_Testing_Project. Accessed: 2019-04-23.

[153] Angelo Prado, Neal Harris, and Yoel Gluck. BREACH. http://breachattack.com/. Accessed: 2019-05-21.

[154] Debian Project. Index of /security/oval. https://www.debian.org/security/oval/. Accessed: 2019-03-28.

[155] Puppet. Introduction to the Puppet language. https://puppet.com/docs/puppet/6.4/lang_summary.html#concept-975. Accessed: 2019-06-12.

[156] Puppet. Module fundamentals. https://puppet.com/docs/puppet/6.4/modules_fundamentals.html#concept-1234. Accessed: 2019-06-12.

[157] Puppet. Puppet architecture. https://puppet.com/docs/puppet/6.4/architecture.html. Accessed: 2019-06-12.

[158] Puppet. Puppet Enterprise and Open Source Puppet. https://puppet.com/products/why-puppet/puppet-enterprise-and-open-source-puppet. Accessed: 2019-06-11.

[159] Puppet. Puppet Forge. https://forge.puppet.com/. Accessed: 2019-06-12.

[160] Puppet. Resource Type Reference. https://puppet.com/docs/puppet/6.4/type.html#built-in-types-and-custom-types. Accessed: 2019-06-12.

[161] Puppet. Welcome to Puppet 6 documentation. https://puppet.com/docs/puppet/6.4/puppet_index.html. Accessed: 2019-06-11.

[162] Inc. Qualys. CRIME: Information Leakage Attack against SSL/TLS. https://blog.qualys.com/ssllabs/2012/09/14/crime-information-leakage-attack-against-ssltls. Accessed: 2019-05-21.

[163] Inc. Qualys. Defending against the BREACH Attack. https://blog.qualys.com/ssllabs/2013/08/07/defending-against-the-breach-attack. Accessed: 2019-05-23.

[164] Inc. Qualys. Is BEAST Still a Threat? https://blog.qualys.com/ssllabs/2013/09/10/is-beast-still-a-threat. Accessed: 2019-05-23.

[165] Inc Qualys. Ssl labs: Deploying forward secrecy. https://blog.qualys.com/ssllabs/2013/06/25/ssl-labs-deploying-forward-secrecy. Accessed: 2019-05-22.

[166] Inc Qualys. TLS Renegotiation and Denial of Service Attacks. https://blog.qualys.com/ssllabs/2011/10/31/tls-renegotiation-and-denial-of-service-attacks. Accessed: 2019-05-19.

[167] Inc. Red Hat. All modules. https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html. Accessed: 2019-06-14.

[168] Inc. Red Hat. Check Mode ("Dry Run"). https://docs.ansible.com/ansible/latest/user_guide/playbooks_checkmode.html. Accessed: 2019-06-14.

[169] Inc. Red Hat. How Ansible Works. https://www.ansible.com/overview/how-ansible-works. Accessed: 2019-06-14.

[170] Inc. Red Hat. Intro to Playbooks. https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#about-playbooks. Accessed: 2019-06-14.

[171] Inc. Red Hat. Introduction To Ad-Hoc Commands. https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html. Accessed: 2019-06-14.

[172] Inc. Red Hat. Roles. https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html. Accessed: 2019-06-14.

[173] Inc. Red Hat. USE CASES. https://www.ansible.com/use-cases. Accessed: 2019-06-14.

[174] Inc. Red Hat. Working With Modules. https://docs.ansible.com/ansible/latest/user_guide/modules.html. Accessed: 2019-06-14.

[175] Pratik Sarkar. ATTACKS ON SSL A COMPREHENSIVE STUDY OF BEAST , CRIME , TIME , BREACH , LUCKY 13 RC 4 BIASES. 2013.

[176] SearcEngineJournal. What is Referrer Spam and How Do You Get Rid of It? https://www.searchenginejournal.com/referrer-spam-get-rid/135855/#close. Accessed: 2019-07-13.

[177] SecPod. ABOUT US. https://www.secpod.com/about-us.html. Accessed: 2019-03-28.

[178] SecPod. SCAP Repo. https://www.scaprepo.com/control.jsp?command=home. Accessed: 2019-03-28.

[179] Security-Database. Oval Repository - org.mitre.oval. https://www.security-database.com/oval.php. Accessed: 2019-03-29.

[180] A Sharma, J. R. Martin, N. Anand, M. Cukier, and W. H. Sanders. Ferret: A Host Vulnerability Checking Tool. In *10th IEEE Pacific Rim International Symposium on Dependable Computing, 2004. Proceedings.*, pages 389–394, March 2004.

[181] Chris Sincerbox. Security Sessions: Exploring Weak Ciphers An Explanation and an Example. https://electricenergyonline.com/energy/magazine/779/article/Security-Sessions-Exploring-Weak-Ciphers.htm. Accessed: 2019-05-24.

[182] Sjoerd Langkemper. The current state of the BREACH attack. https://www.sjoerdlangkemper.nl/2016/11/07/current-state-of-breach-attack/. Accessed: 2019-07-12.

[183] Nick Sullivan. CRIME, TIME, BREACH and HEIST: A brief history of compression oracle attacks on HTTPS. https://www.helpnetsecurity.com/2016/08/11/compression-oracle-attacks-https/. Accessed: 2019-05-24.

[184] SUSE. OVAL Information. http://ftp.suse.com/pub/projects/security/oval/. Accessed: 2019-03-28.

[185] Inc. Synopsys. The Heartbleed Bug. http://heartbleed.com/. Accessed: 2019-05-24.

[186] Whitewood Encryption Systems. UNDERSTANDING AND MANAGING ENTROPY. https://www.blackhat.com/docs/us-15/materials/us-15-Potter-Understanding-And-Managing-Entropy-Usage-wp.pdf. Accessed: 2019-05-20.

[187] Byungchul Tak1, Hyekyung Kim1, Sahil Suneja, Canturk Isci, and Prabhakar Kudva. Security analysis of container images using cloud analytics framework. In Hai Jin, Qingyang Wang, and Liang-Jie Zhang, editors, *Web Services – ICWS 2018, 25th International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings*, volume 10966 of *Lecture Notes in Computer Science*. Springer, June 2018.

[188] British Telecommunications. How long until Microsoft support for Windows 7 ends? https://home.bt.com/tech-gadgets/computing/windows-7/windows-7-support-end-11364081315419. Accessed: 2019-03-29.

[189] The Apache Software Foundation. Apache Module mod_dav_lock. https://httpd.apache.org/docs/2.4/mod/mod_dav_lock.html. Accessed: 2019-07-04.

[190] The Apache Software Foundation. Apache Module mod_log_config. http://httpd.apache.org/docs/current/mod/mod_log_config.html/. Accessed: 2019-07-02.

[191] Martin Thomson, Salvatore Loreto, and Greg Wilkins. Hypertext Transfer Protocol (HTTP) Keep-Alive Header. https://tools.ietf.org/id/draft-thomson-hybi-http-timeout-01.html#rfc.section.6. Accessed: 2019-05-23.

[192] Ask Ubuntu. Cannot find /etc/apache2/sites-available/default when configuring apache. https://askubuntu.com/questions/386382/cannot-find-etc-apache2-sites-available-default-when-configuring-apache. Accessed: 2019-04-09.

[193] Andrew van der Stock, Brian Glas, Neil Smithline, and Torsten Gigler. OWASP Top 10 - 2017 The Ten Most Critical Web Application Security Risks. https://www.owasp.org/index.php/Top_10-2017_Top_10. Accessed: 2019-03-19.

[194] W3Techs. Usage of web servers. https://w3techs.com/technologies/overview/web_server/all. Accessed: 2019-02-24.

[195] David Waltermire and Jessica Fitzgerald-McKay. Transitioning to the Security Content Automation Protocol- SCAP (SCAP) version 2. https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.09102018.pdf, September 2018. Accessed: 2019-04-02.

[196] Rui Wang, XiaoFeng Wang, Kehuan Zhang, and Zhuowei Li. Towards automatic reverse engineering of software security configurations. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 245–256, New York, NY, USA, 2008. ACM.

[197] Tianyin Xu and Yuanyuan Zhou. Systems Approaches to Tackling Configuration Errors: A Survey. *ACM Comput. Surv.*, 47(4):70:1–70:41, July 2015.

[198] Jiaqi Zhang, Lakshminarayanan Renganarayana, Xiaolan Zhang, Niyu Ge, Vasanth Bala, Tianyin Xu, and Yuanyuan Zhou. Encore: Exploiting system environment and correlation information for misconfiguration detection. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, pages 687–700, New York, NY, USA, 2014. ACM.

Printed by Tryckeriet i E-huset, Lund 2019

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2019-730
http://www.eit.lth.se