LAPPEENRANTA-LAHTI UNIVERSITY OF TECHNOLOGY LUT
School of Engineering Science
Software Engineering

Bachelor’s thesis

Elmer Häyrynen

EVALUATION OF STATE-OF-THE-ART WEB APPLICATION VULNERABILITY SCANNERS

Examiners: Associate Professor Ari Happonen
Supervisors: Associate Professor Ari Happonen

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma

Elmer Häyrynen

Nykyaikaisten verkkosovellusten tietoturvaskannereiden arviointitutkimus

Kandidaatintyö

2020

54 sivua, 0 kuvaa, 7 taulukkoa, 1 liite

Työn tarkastajat: Apulaisprofessori Ari Happonen

Hakusanat: verkkosivujen tietoturvaskannerit, verkkosivujen haavoittuvuudet, penetraatiotestaus, vertailututkimus

Verkkosivujen tietoturva on yksi nykypäivän suurimmista ongelmista useiden perinteisten palveluiden siirtyessä verkkoon uusien verkkoteknologioiden siivittämänä. Ketterien metodien yleistynyt käyttö kehitysympäristöissä ovat nopeuttaneet sovelluksen kehitysprosesseja sekä sovellustestausta, mutta sovellusten tietoturvatestaus sekä etenkin penetraatiotestaus eivät ole vielä mukautuneet näihin muutoksiin. Tämä johtuu suurelta osin perinteisen tietoturvatestauksen luonteesta, sillä testaaminen on usein manuaalista sekä aikaa vievää työtä, joka vaatii vankkaa tietoturvaosaamista. Verkkosivujen tietoturvaskannerit ovat työkaluja, jotka pyrkivät helpottamaan tietoturvatestauksen sisällyttämistä sovelluskehityksen eri vaiheisiin automatisoimalla testausprosessia sekä integroitumalla osaksi muita kehityksessä käytettäviä sovelluksia. Vaikka tietoturvaskannerit ovat lähtökohtaisesti samaan tarkoitukseen suunniteltuja, ne eroavat usein keskenään haavoittuvuuksien kattavuuden, sovellusten yhteensopivuuden sekä integraatiomahdollisuuksien suhteen. Tässä kandidaatintyössä analysoidaan ja vertaillaan keskenään luokkansa parhaita verkkosivujen tietoturvaskannereita ja työn sisältö on jaettu neljään osioon. Ensimmäisessä osiossa käydään läpi erilaisia verkkosivujen haavoittuvuuksia sekä penetraatiotestauksen ja skannereiden roolia osana verkkosivun kehitystä. Toisessa osiossa määritellään millä kriteereillä skannereita voidaan pitää luokkansa parhaina sekä etsitään ja valitaan tutkimuksen kohteeksi kuvaukseen sopivia skannereita olemassa olevasta kirjallisuudesta. Kolmannessa osiossa tutkimuksen tulokset on esitetty kootusti ja neljännessä osiossa keskitytään tutkimuksen tulosten arviointiin sekä vertailuun.

ABSTRACT

Lappeenranta-Lahti University of Technology
School of Engineering Science
Software Engineering

Elmer Häyrynen

Evaluation of state-of-the-art web application vulnerability scanners

Bachelor's Thesis

2020

54 pages, 0 figures, 7 tables, 1 appendix

Examiners: Associate Professor Ari Happonen

Keywords: web application security scanners, web application vulnerabilities, penetration testing, comparison study

Security of web applications is one of the largest problems of today due to the transformation of traditional services to online variants, fuelled by constantly evolving new technologies. While the modern development and testing of web applications has become faster and more efficient through the deployment agile methodologies, software security testing and especially penetration testing is lacking as it's traditionally a manual and time-consuming process conducted by security experts. In order to include security testing into various stages of software development life cycle, modern web application security scanners have been developed in order to make integrated web application security a reality through automation and integrating with the existing development related software. Essentially web vulnerability scanners detect potential security vulnerabilities in web application by performing automated tests, and while nearly all of them are based on similar functionalities and aim to resolve the same core problem, there are vast differences amongst them related to their threat detection capabilities, environment compatibility, integration possibilities and usability. This thesis analyses and compares different state-of-the-art web vulnerability scanners and consists of four main sections: the first section covering the vulnerabilities related to modern web applications and explaining the role of penetration testing and web application scanners in software development, the second section defining the state-of-the-art web scanners, inspecting available literature and benchmark studies for scanners that match the defined qualities and forming an assessment criteria for the comparison study, the third part focusing in presenting the collected results and the last section focusing around analysing the study outcomes.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CAPTCHA	Completely Automated Public Turing Test To Tell Computers and Humans Apart
CR	Carriage Return
CRLF	Carriage Return Line Feed
CSRF	Cross-site Request Forgery
CTO	Chief Technology Officer
CWE	Common Weakness Enumeration
DAST	Dynamic Application Security Testing
DOM	Document Object Model
EU	Europe Union
GDPR	General Data Protection Regulation
HTML	Hypertext Markup Language
HTTPS	Hyper Transfer Text Protocol Secure
IBAN	International Bank Account Number
ID	Identifier
IFRAME	Inline Frame
IMG	Image
IP	Internet Protocol
JS	JavaScript
LDAP	Lightweight Directory Access Protocol
LDAP	Lightweight Directory Access Protocol
LF	Line Feed
LFI	Local file includes
N/A	Not Applicable
NAT	Network Access Translation
NoSQL	Not only SQL
OS	Operating System
OWASP	Open Web Application Security Project

PCI	Payment Card Industry
RFI	Remote file includes
SAST	Static Application Security Testing
SDE	Sensitive Data Exposure
SDLC	Software Development Life Cycle
SOAP	Single Objects Access Protocol
SQL	Structured Query Language
SSI	Server-side Include
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
WASC	Web Application Security Consortium
WAVSEP	Web Application Vulnerability Scanner Evaluation Project
XML	Extensible Markup Language
XPath	XML Language Path
XPath	XML Language path
XSS	Cross-site scripting
XXE	XML External Entity
ZAP	Zed Attack Proxy

1 INTRODUCTION

This thesis evaluates and compares the differences of current state-of-the-art web application vulnerability scanners designed for automated security testing of web applications. The aim is to ease the process of choosing a suitable web application vulnerability scanner for a software product by providing a report examining their vulnerability detection features, integration possibilities and overall development environment support for easy comparison.

1.1 Background

Due to rapid digital transformation of our society, understanding the presence of web application vulnerabilities and the fundamentals of security testing in modern web application development is the responsibility of all CTO's in both private and public companies as well as municipality units. This is emphasized when these instances go through digital transformation towards knowledge companies (Kortelainen et. al. 2016; Kortelainen et al. 2019), as transformative companies are most severely impacted by security breaches. (Ko et al. 2009, p. 12) When managers understand the possible vulnerabilities in web applications as well as the methods and tools used by the software industry, they are better equipped to demand proper security measures and practices from software providers in order to minimize the chances of potential security breaches.

In the context of web applications, penetration testing is most often used as black-box testing method. Penetration testing focuses in finding potential security vulnerabilities from the web-application by analysing the application's execution and behaviour when tests pass malicious inputs into the system. Due to the nature of penetration testing, the amount of test cases can reach hundreds or thousands, thus rendering manual penetration a non-viable option. To address this issue, automated penetration testing tools have been developed in order to improve efficiency of penetration testing. (Antunes & Vieira 2014, p. 31.)

Automated penetration testing is most often achieved in one of the two ways: by ordering a customized penetration testing solution from a team of security experts, which is essentially a collection of customized and build-safe exploits created by the experts, that are combined into a single executable package, (Stefinko et al. 2016, p. 490) or by using a commercially available penetration testing tool for performing automatic web penetration testing, usually referred to as web application vulnerability scanners, (Aarya et al. 2018 p. 124) but also referenced to as dynamic application security testing tool or DAST tool, which this thesis will focus in examining and comparing. The main idea behind a web application vulnerability scanner is that it crawls a target website and collects information on possible data-entry points, which are then attacked by the scanner application by sending malicious inputs into the system and analysing the system's behaviour under attack. (Fong et al. 2008) In the context of this text, a scanner, web application scanner, web vulnerability scanner, web application vulnerability scanner and DAST tool all refer to an application of this type. Even though penetration testing as an approach does not require access to the actual source code (Antunes & Vieira 2014, p. 31), and as such is much less platform-dependant than other security testing methodologies, the process behind selecting a suitable web application vulnerability scanner for a system can often be tedious, as they often differ greatly in terms of functionality, supported architectures, vulnerability reporting methods, integration possibilities and cost.

The main advantage of having a functional web application vulnerability scanner testing your system is that it can detect the vulnerabilities in the target system in real-time. In addition to providing valuable information about the system's current state of vulnerability, having a functional web application vulnerability scanner tightly coupled with the development process can aid in improving the overall security of the system by also educating developers in building more secure systems. The downside is that vulnerability scanners tend to find security vulnerabilities that are often considered to be "low hanging fruits", meaning that the results of a scan often miss various vulnerabilities that would be detectable by a more through manual penetration testing. This might result in the scan results creating a false sense of security for software developers and application managers in cases this drawback is not properly understood.

1.2 Goals and delimitations

The goal of this thesis is to provide a table which includes the results of the examination that can be used for comparing features and functionalities of various state-of-the-art penetration testing tools, which will ease the process of selecting a suitable penetration testing tool for a web application. The thesis will consist from three main phases. The first phase will focus in examining web application vulnerabilities as well as explaining the role of penetration testing and web application scanners in the modern web application development. The second phase of the study will focus in defining and selecting state-of-the-art web application vulnerability scanners for further analysis. In addition to this, the section will also focus defining the assessment criteria so that the assessed tools can be examined from various perspectives such as:

-Threat coverage: What type of vulnerabilities does the tool scan for?

-Reporting: How are the reports formed and presented to the user?

-System support: What kind of web applications are supported?

-Setting up: How are the tests created and maintained and what kind of preparations are required?

-Usage: How are the tests conducted?

-Cost: What does the tool cost?

The last part of the study focuses in presenting and analysing the results of the study.

As stated by Stuttard & Pinto (2011, p. 781-784), the performance of an individual penetration test tool is tightly coupled with the software being tested and different tools excel in different type of applications and differ in their capability to address and identify different types of vulnerabilities. In the context of this thesis, the focus will not be in the efficiency and functionality of the web application vulnerability scanners.

The focus of this study is in examination of web application vulnerability scanners from perspectives of software documentation provided by the respective software companies or communities as well as existing literature that focuses in the field of automated penetration testing.

1.3 Structure of thesis

Section 1 contains a deeper background view to the automated penetration testing and the field of web application security as a subject, explaining the advantages, challenges and topicality of these subjects. In addition, the challenges in modern web-application penetration testing will be introduced as well as the role of automated penetration testing tools in order to overcome them.

Section 2 defines the tools that qualify for this study and the assessment criteria for conducting the research. Existing studies and information will be used in order to identify state-of-the-art tools for this study and also to define the perspectives from which to assess the selected tools and define the key functionalities and capabilities of each tool in order to carry out the study. The process and results of studying each individual tool will be presented in the Appendix 1 of this thesis.

Section 3 describes the overall results of the study by collecting the results of each individually assessed tool into a comparison matrix for easy examination. Section 4 continues with further discussion about the overall results and section 5 concludes the paper and proposes possible future work.

2 WEB APPLICATION SECURITY

Software security as a subject is all about taking security in account at all stages and aspects of the development process. This means that in addition to building the software to be secure, the personnel developing and designing the software, as well as the users using the software must be educated in order to improve security. (Mcgraw 2004, p. 80) In comparison to traditional software testing which is most often conducted by running dynamic and functional tests in order to ensure the validity and correct functionality of the application's features, security of the system cannot be validated since security cannot be addressed as set of features. (Arkin, Stender & McGraw 2005, p. 84)

2.1 Web-application security threats

The volume of attacks against web applications has increased substantially in the past years (Sonicwall 2019), and according to the most recent Symantec Internet Security Threat Report, on average approximately 953,800 web attacks are performed towards websites every day. (Symantec ltd. 2019) These attacks are most often built to exploit a design or security flaw in a web application in order to gain confidential data, and while the variety of cybersecurity attacks and attack vectors is immense, the subject of common web-application vulnerabilities is quite easy to perceive. A commonly referenced list of most severe vulnerabilities is the OWASP Top 10-project, updated and maintained by the OWASP-community. (OWASP org. 2017a) In addition to the OWASP Top 10 listing, there are other more comprehensive listings such as the Web Application Security Consortium Threat Classification (WASC 2009q), but the listings are often complementary and contain similar vulnerabilities with different emphasis.

2.1.1 Injection

Injection is the most common security flaw in a web-application, according to OWASP 2017 Top-10 report. Injection occurs, when an attacking entity sends hostile request data to the web application as a part of command or a query. The hostile data or script included as a part of the query can then cause unwanted behaviour when it is executed by the backend, such as executing unintended commands or accessing confidential data without proper authorization. Injection flaws are most often found in the system's data-access-layer, including SQL, LDAP, XPath or NoSQL queries and other commands. (OWASP org.

2017b)

2.1.2 Broken authentication

Broken authentication refers to issues in the design of the website related to authenticating the user's session and identity. This is one of the vulnerabilities that cannot be automatically identified through automated web vulnerability scanners, as traditionally when performing penetration testing the tester does not have access to the code of the target system and as such cannot identify these faults in system design. Issues that could result in broken authentication vulnerabilities could for example be related to use of insufficient cryptography in storing the user credentials in the web application backend database. Some of the security issues connected to this area can be identified by automated tools, such as exposed session id's or the transmission of the user credentials over an insecure connection, but these are insufficient in order to identify all broken authentication related threats from the system. (OWASP org. 2017c)

2.1.3 Sensitive Data Exposure (SDE)

Sensitive data exposure occurs in cases where web applications and API's do not protect sensitive data sufficiently, which enables attackers to steal or modify included sensitive data for conducting frauds or identity thefts. This is especially severe if the data handled by the application contains data that falls under any form of privacy law such as the EU's General Data Protection Regulation or financial data protection laws such as the PCI Data Security Standard. SDE is often caused by lack of strong protection and encryption in any of the states when handling the sensitive data: storing, transferring or presenting it in an application and as such, the possibility for this vulnerability is present in every application that manages sensitive data. (OWASP org. 2017d)

2.1.4 External entities (XXE)

External entities are mainly a problem of legacy XML processors. The attack happens when an XML entity which contains a reference to an external entity is processed by a processor. This manipulation of XML contents often allows an attacker to inspect files on the application server and to interact with any external entity or system that the web application has access to. XXE vulnerability is present in applications that accept XML directly or use XML uploads, especially from untrusted sources, any XML processor or SOAP web services have their document type definitions enabled. (OWASP org. 2017e)

2.1.5 Broken Access Control

Broken access control refers to possible logical security problems included in web applications. A classic example of this threat would be that a system has introduced a profile system, in which users and their rights to access and modify system contents are defined by their set role. Broken access control would mean that a user is able to access or use the system features which they're not intended to be able to do, resulting in unwanted behaviour and a security flaw in terms of access control. To determine this kind of behaviour, the tester would be required to have an extensive knowledge about the target system in order to determine if the access control is broken. (OWASP org. 2017f)

2.1.6 Security Misconfiguration

Security misconfiguration happens when an application's configuration causes a security threat to the web application. This misconfiguration can happen at any level of the application, such as platform, server, database and framework. The most common cases of security misconfiguration are out-of-date software dependencies, insecure default accounts, security settings in development frameworks and error handling that could result in severe consequences such as the attackers being able to compromise the entire application. (OWASP org. 2017g)

2.1.7 Cross Site Scripting (XSS)

XSS flaws occur in cases where a web application includes untrusted data in a new web page without properly validating or escaping the current page, or in cases where the application updates an existing page with data that is coming from the user through an API which interprets this data into HTML or JavaScript. This allows attackers to execute malicious JavaScript code in the victim's browser, enabling them to modify the page content or hijack sessions. This vulnerability was rated most severe web application security flaw in 2007. It has since lowered to rank 7 in the newer 2017 report. (OWASP org. 2007; OWASP org. 2017h)

2.1.8 Insecure Deserialization

Insecure deserialization refers to a vulnerability where Application and API's deserialize hostile data supplied by an attacker, which can then lead to application logic modification, arbitrary remote code execution. Exploiting this vulnerability is usually quite difficult, as exploits often need to be changed or tweaked in order to have an effect and thus require frequent human assistance in order to be validated. (OWASP org. 2017i) When an application deserializes untrusted data without sufficiently validating it, the application enables the attacker to gain control of the code execution. In order to perform an attack of this type, an attacker would require detailed knowledge of the source code of an application in order to determine how to exploit insecure deserialization. (Netsparker ltd. 2019b)

2.1.9 Using Components with Known Vulnerabilities

This vulnerability is the result of applications being constructed on top of different frameworks and components provided by 3rd parties, which can easily lead to cases where the development teams do not fully understand which components they use in their application or how to keep all of them updated. (OWASP org. 2017j) It's typical that automated security solutions can have checks for known vulnerable frameworks, platforms and applications, but maintaining a database containing all the vulnerable components and applications is virtually impossible. (Netsparker ltd. 2019b) Determining the exploitability and vulnerabilities of these components requires manual work and should always be on the responsibility of the developers. (OWASP org. 2017j)

2.1.10 Insufficient logging and monitoring

Occasionally, the developers of the software have not included accurate enough logging and monitoring systems related to the system. This enables the attackers to conduct vast attacks against systems without the administrators being aware of the situation and therefore being not able to respond to the threat. An example of insufficient monitoring would that a system does not monitor or log the user's log in credentials with related IP addresses, enabling the attackers to perform a brute-force attack to find out possible user credentials. (OWASP org. 2017k)

2.2 Penetration testing

Penetration testing in the context of web applications is a commonly used black-box testing method which resolves around passing multiple malicious inputs to the target system from the attacker’s perspective in order to reveal vulnerabilities in the target web application. Penetration testing as an approach does not require access to the actual source code, and as such is much less platform-dependant than other security testing methodologies. (Antunes & Vieira 2014, p. 30)

While being one of the most commonly applied method out of all the software security best practices, penetration testing is also the one that is being misapplied most commonly. Traditionally it is most often performed by security consultants assessing the software as a part of the final acceptance phase. However, a late lifecycle penetration testing tends to uncover underlying security problems such as architectural mistakes or common vulnerabilities too late, severely affecting the budget and development time of the software in case they must be dealt with before the software can be delivered. As a tool for measuring the security of the system, penetration testing is most efficient once it is integrated into the development process in a way that the results of the tests can be used to improve the practices of software design, deployment and implementation. (Arkin, Stender & McGraw 2005, p. 84).

Penetration testing as a manual process is very complex and tedious process, as the security specialists conducting the testing are often required to have various years of relevant experience and a wide skillset. The specialists typically create their own exploits, which they modify accordingly to the target being tested. Conducting a comprehensive and through-out manual penetration testing usually requires multiple of specialists with different skillsets, rendering most organizations unable to maintain an in-house team or contract a manual penetration testing team often. (Stefinko et al. 2016, p. 490)

The usual process of penetration testing is conducted in three parts. First part focuses in gathering information about the target web application. Second part focuses in using the gathered information in order to conduct the vulnerability analysis, which is directed towards various parts of the web application such as business logic, authentication, authorization, session management, data validation, denial of service etc. The last phase is the vulnerability analysis step, where the uncovered vulnerabilities are targeted by attacks in order to identify the possible exploitations and gather information about the vulnerability’s overall effect for the system. (Aarya et al. 2018, p. 124)

2.3 Automated penetration testing

As manual penetration testing is often seen as ineffective in terms of both money and time, automated penetration testing is often considered as a more effective alternative solution. Automated penetration testing tools are often created by a team of security experts which complete a through-out research of the system for vulnerabilities, build custom tests to run against the system for testing purposes and finally combine these tools into a single executable package. (Stefinko et al. 2016, p. 190) Another solution for this problem in the context of web applications is using a web vulnerability scanner, which is essentially an automated tool for testing web applications for the presence of common vulnerabilities and security problems. (Aarya et al. 2018, p. 124)

Web vulnerability scanners are essentially automated programs designed to examine web applications for security vulnerabilities and possible evidence of software coding errors. Web scanners operate by exploring an application by crawling through every web page of the application and performing automated penetration tests against the uncovered access points in the discovered pages, which involves generation of malicious inputs and subsequent evaluation of the target application’s response to these inputs. (Fong et al. 2008) It’s important to understand that the web vulnerability scanners mainly focus in identifying vulnerabilities from the web application which is the second phase of penetration testing process.

Automation in software development has become topical recently due to the transformation from traditional software development methods towards agile methodologies. The agile methodology has decreased the duration of development cycles drastically, and the change can be clearly seen in the emphasis of automation in traditional software quality assurance. According to Crispin et al., rather than relying solely on the tester, software quality is emphasized to be everyone’s responsibility and a continuous process executed alongside the development process. This is achieved through a heavy emphasis on automation. (Crispin & Gregory 2009, Foreword section, p. XXIII)

	Manual penetration testing	**Automated penetration testing**
Testing process	– Manual process, non-standard – Time consuming – Expensive – High amount of customization required to work with target application	– Fast, standardized process – Repeatability – Good compatibility without much customization
Vulnerability / Attack vector database management	– Manual maintenance – Need of customizing attacks to different platforms	– Attack database automatically maintained – Preconstructed attack codes for multiple platforms
Reporting	– Often requires manual data-collection	– Automated reporting capabilities
Cleanup	– Manual undo of changes to the system every time vulnerabilities are found, as tests can damage system data	– Can offer clean-up solutions
Training	– Testers need to learn a vast amount of different ways of testing – Testing can be customized and is time-consuming – Difficult to adopt by other than skilful testers	– Training for automated tools is easier, as all tools can be executed using the same UI – Easily adoptable by other than just testers

Cost	– Cost consists mainly from the work that it takes to conduct a comprehensive penetration test, conducted by either in-house testers or external security consultants.	– Cost consists mainly of the software licenses or monthly fees. Initial required work is much lower than in manual testing.
Accuracy	– Very accurate if conducted correctly by professionals with required skillsets – Customization of test cases towards target system and human factor provide greatly increased accuracy	– Possible false alarms – Most often finds the “low hanging fruits” when it comes to vulnerabilities – Results may vary greatly depending on the target system architecture and configuration

Table 1: Comparison between manual and automated penetration testing methods

Table 1 is a comparison between automated and manual penetration testing methods based on a similar table presented in a study conducted by Stefinko et al. (2016) which suggests that automated penetration testing methods have multiple advantages over traditional manual methods. These advantages are emphasized even more due to transformation towards agile methods requiring more speed, flexibility and integration possibilities from the testing tools. Automated penetration testing tools also have some drawbacks, and in order to integrate them into the development process, these drawbacks must also be understood and considered in order to avoid unwanted results.

In comparison to manual penetration testing, automated penetration testing does not provide as extensive test-data coverage as the manual penetration testing, increasing the false-positive and false-negative findings rates of the used tools. (Huang et al. 2017 p. 82). The core problem in automated penetration testing tools is that in in order to identify vulnerabilities in practice, the tools must analyse the output of the web application. Not being able to access or analyse the application’s internal behaviour decreases the reliability and accuracy of these tools. (Antunes & Vieira 2014, p. 32) The rates are often improved over time as the tools are taught to handle system specific cases and better filter the vulnerabilities. This does, however, lead to larger amount of work required from the personnel and can potentially harm trust towards the scanning system reports, causing negative effects in terms of improving system security.

The detection coverage rate is another main point of interest when it comes to the web application security scanners. In a research conducted by Antunes and Vieira (2014), four different web vulnerability scanners were compared between each other and the results displayed that none of the four was able to detect even one third of the total number of known vulnerabilities in the test site. The detection coverage rates differ greatly between the scanners, and in addition to differences between scanners, the detection coverage rate also depends highly on the system being tested. (Stuttard & Pinto 2011, 781-784) The low detection coverage rate, however, does not tell the entire truth, as the web vulnerability scanners often find undetected vulnerabilities which are preceded by other similar vulnerabilities, which can more easily be detected once the first has been fixed. (Antunes & Vieira 2014, 34) The deficiency of vulnerabilities web vulnerability scanners can find is good to keep in mind, as using one might otherwise provide a false sense of security to the development personnel.

3 STATE-OF-THE-ART WEB APPLICATION SCANNERS: SAMPLING AND EVALUATION CRITERIA

The study will focus on web vulnerability scanners, that can be considered state-of-the-art. State-of-the-art can be understood as ‘very modern and using the most recent ideas and methods’ (*Cambridge Advanced Learner’s Dictionary & Thesaurus* 2020, state-of-the-art entry). When applying this definition into the context of web vulnerability scanners, they are ‘very modern’ in cases where they have recently been or are still being actively developed. Respectively, web security scanners are considered to use ‘the most recent ideas and methods’ in cases where they are capable of identifying and responding to the most recent attack vectors and providing support to modern website development frameworks and trends. In the context of this study, web application scanners are considered state-of-the-art when they fulfil the aforementioned criteria.

Today, various vulnerability scanners exist, including both commercial and open source tools. These tools differ from each other in many ways, such as methods that the use to identify threats, cost of the software, integration possibilities and even in terms of threat detection effectiveness. The large number of different available tools has led to numerous studies focusing in analysing and comparing the effectiveness of various web scanners, which will be used in this study to identify current state-of-the-art web scanners.

3.1 Web application scanner studies

OWASP has created a benchmarking platform consisting of thousands of test cases for evaluating the speed, coverage and accuracy of both SAST and DAST tools. The benchmark has gathered results for various open-source tools, presented in table 2 and also commercial tools presented in table 3, but due to the fact that each commercial tool has licenses defining when their produced results can be released/made public, they have not revealed exact results of these commercial tools as it might be against the tool’s licenses. The OWASP benchmark also states that they haven’t been able to get a clean run against the benchmark from various tools and therefore the results are not available as a comparison chart, but rather they provide instructions for the user on how to run the benchmark on a selection of tools. (OWASP org. 2020b)

Table 2: Commercial tools supported by the OWASP Benchmark

Application	**Vendor**
Acunetix web vulnerability scanner	Acunetix ltd.
Burp Pro	Portswigger
HCL (Formerly IBM) AppScan	HCL Technologies ltd.
Fortify WebInspect	Micro Focus (Formally HPE)
Netsparker	Netsparker ltd.
Qualys web app scanner	Qualys Inc.
AppSpider	Rapid7 ltd.

Table 3: Open source tools supported by the OWASP Benchmark

Application	**Source**
Zed Attack Proxy (ZAP)	www.zaproxy.org (OWASP project)
Arachni	www.arachni-scanner.com

The OWASP also provides a listing of vulnerability scanning tools which has multiple tools presented which are missing from the benchmarking study. OWASP states on the listing page that they are aware of the Web Application Vulnerability Scanner Evaluation Project, while they do not directly endorse the results of the study or any DAST tool that the study evaluates. However, they admit that the WAVSEP study has far more detail on DAST tools and their features than their DAST page and encourages anyone interested in researching or selecting DAST tools for their projects to inspect the results of the study. (OWASP org. 2020f)

WAVSEP DAST benchmark test for automated penetration testing tools is a recurring study conducted by Shay Chen, an independent information security researcher and analyst. The study analyses and compares various features of both commercial and open-source tools for performing automated penetration testing, providing a good overview of the current level of development in the field of web scanners. (Chen 2017) The web application scanners examined in the WAVSEP DAST Benchmark 2017/2018 consist of a selection of the most popular and recent web application scanners and the results of this study are widely referenced throughout the infosec community.

Table 4: Commercial tools analyzed in WAVSEP DAST Benchmark 2017/2018

Application	**Version**	**Vendor**
Appsider	v6.14.060	Rapid7 ltd, acquirer of NTO
Netsparker	v4.8	Netsparker ltd.
Acunetix	v11.0.x, build 171181742	Acunetix ltd.
Burpsuite	v17.10XX	Portswigger
WebInspect	v17.10XX	HPE
WebCruiser	v3.5.4	Janusec
AppScan	v9.0.0.999 build 466	IBM

When comparing the sampling of commercial scanners analysed in the WAVSEP DAST benchmark and the OWASP benchmark, the uniformity is obvious, with the exception that the OWASP listing is more up-to-date as it also presents some of the tool ownership changes that have occurred after the 2017/2018 WAVSEP DAST benchmark.

Table 5: Open source tools analyzed in WAVSEP DAST Benchmark 2017/2018

Application	**Version**	**Source**
Zed Attack Proxy (ZAP)	2.6.0	www.zaproxy.org (OWASP project)
Arachni	1.5-0.5.11	www.arachni-scanner.com
IronWASP	0.9.8.6	ironwasp.org
WATOBO	v0.9.22	watobo.sourceforge.net
W3AF	1.6	w3af.org
Vega	1.0	subgraph.com/vega
Wapiti	2.3.0	wapiti.sourceforge.io
Skipfish	2.1.0	tools.kali.org/web-applications/skipfish
XSSer	1.6.1	xsser.03c8.net/

Regarding the analysed open source tools presented in both the WAVSEP DAST and OWASP Benchmark, the WAVSEP sampling is much larger than the one used in the

OWASP benchmark, and the common tools examined are the Zed Attack Proxy and the Arachni. Mburano & Si (2018) have further investigated these scanners in their recent evaluation study in 2018. The study focuses in analysing both scanners using the results from OWASP and WAVSEP benchmarks. The study notes, that while Arachni and ZAP are not considered to be on par with the commercial scanners, they provide consistent results, have a wide number of contributors and have grown to be quite popular among penetration testers due to their easy obtainability and virtually zero cost.

3.2 Web Application Scanner selection

Based on the scanners evaluated and included in the benchmarks and studies discussed above, the base sampling of scanners for this study will consist of the scanners present in both the examined benchmark studies. Tools that have been selected for this study are presented in table 6. In order to reduce the overall work required for completing this study, the scope was narrowed down to examining four different commercial scanners and two different open-source scanners.

When examining the commercial scanners further, it was most often important to decide the version to examine, as most of the commercial tools offer a variety of versions for supporting different types of clients: security professionals which most often only need strong single user tools, organizations which often need possibilities to integrate the scanners into their SDLC software. In the context of this study the focus is on the organizational or enterprise versions of these software whenever possible as they provide support for various SDLC integrations and enhanced threat detection features which are some of the largest advantages achievable by web application vulnerability scanners.

When examining the open source scanners, the two scanners present in both the OWASP and WAVSEP DAST benchmarks were the ZAP and Arachni. The selection of these two was also supported by the study by Mburano & Si (2018) as they redeemed these scanners to have a good status among the web application security tester community.

Table 6: Tools qualified for the study

Application	Vendor / source

Acunetix Premium	Acunetix ltd.
Burp Suite Enterprise	Portswigger
NetSparker Team	Netsparket ltd.
AppSpider Enterprise	Rapid7 ltd.
Zed Attack Proxy	OWASP project
Arachni	www.arachni-scanner.com

3.3 Threat detection capabilities

A web application scanner's main task is to scan a system for multiple types of security problems, both vulnerabilities and architectural weaknesses. The OWASP Top-10 listing of vulnerabilities is a good baseline for representing the most critical security risks to web applications (OWASP org. 2017a), but in the context of web application security scanners it's hard to use as a baseline since the scanners are not able to tackle all of the presented issues due to their nature as vulnerabilities. (Netsparker ltd. 2019b) Another similar project aimed at identifying critical security flaws with an even broader perspective is the Web Application Security Consortium project. The project is a highly cooperative effort to clarify the threats to the security of a web application, and the authors consist of application developers, security professionals, software vendors and compliance auditors. (WASC 2009q) The following list of targeted vulnerabilities is based on the WASC criteria for assessing web application security scanners, which in turn is essentially extracted from the WASC Threat classification 2.0 enlisting.

3.3.1 Authentication

Authentication aims mainly at replying to the question "who are you?" Essentially authentication policy aims at describing certifications for users that identify them from the system perspective. Authentication is often used in conjunction with authorization policy and privacy policy in order to solve the fundamental problem of access control and privacy protection. (Yongsheng et al. 2010, p. 236) Authentication issues in web application systems open a possibility for attackers to gain access to the system through an existing account which they may obtain through various methods.

Brute force attack is a method where an unknown value such as password is determined through an automated process which tries a vast quantity of possible values. Passwords are the classic example of brute forcing, but the technique also applies to various other identification methods such as guessing session identifiers, finding possible hidden directories and files within the system, or even guessing credit card information. Web application scanners can be used to detect and verify brute force attacks by identifying insufficient account lockdown policies within the system, as well as issues with unthoughtful logging measures that for example return a different login failure messages for valid and invalid usernames. (WASC 2009a)

Insufficient authentication happens when a web application allows the attacker to access sensitive information or functionality without having to properly authenticate within the system. Traditionally some resources of a website, such as administrative resources are protected by simply hiding their location within a system by not linking it to anywhere on the main web site. It’s important to understand that while the resource is unknown to the attacker, it can still be accessed through a direct URL path, which can be discovered through various methods, such as crawling through error messages, referrer logs, brute forcing etc. (WASC 2009h)

Weak password recovery validation refers to a case where an application allows an attacker to obtain, change or recover another user’s password through the mechanism designed to provide means for the user to gain access to their account in cases they have forgotten password. This is most often possible in cases where the information required to validate a user’s password is either easy to be guessed or bypassed. Main methods of web application scanners for performing this action are directed brute force attacks towards the recovery system. (CWE 2020; WhiteHat Security Inc. 2020)

Lack of SSL on login pages is easily detectable by web application scanners and refers to cases where the login pages are not supporting the HTTPS protocol. When an application does not use any sort of SSL or TLS protection for their access, it opens multiple possibilities for attackers, such as server spoofing, man-in-the-middle types of attacks. Failure to use the SSL protection on the website also enables communication phishing since the communication between the server and user remains unencrypted. (SSL.com 2019)

Auto-complete not disabled on password parameters refers to a case where password input field values do not have the AutoComplete -attribute of the HTML input fields set to off. This enables scenarios where the browser caches the input passwords enabling them to be re-used without the user's consent. This is something that is often easily detectible by web application scanners by examining the html structure of the target application. (OWASP org. 2014b)

3.3.2 Authorization

Authorization or access control most often answers to the question "What can you do?" It acts in conjunction with the authentication policy and determines what processes, features and documents the authenticated user may access and use. (Yongsheng et al. 2010) Authorization issues in a web application context can often leads to the display of data to the user that they should not see, or to complete processes that they're not intended to be able to do. This is somewhat tricky to detect with automated penetration testing tools, as authorization and security misconfiguration often require vast knowledge of the target system and it's functionalities, as the context of what is allowed and what isn't is sometimes hard to pass to the scanner software. There are however some cases that the scanners are often able to detect. (Netsparker ltd. 2019b)

Credential / session prediction refers to a case where the unique identifier value of a user is guessed by the attacker, enabling them to issue web site requests with the compromised user's privileges. This is in some cases easily detectable by the scanners, as the session ID's are often generated using proprietary algorithms and then stored in a cookie, hidden form-fields or URL parameters. This allows the web scanner to either brute force or calculate a session ID and switch the value of its current session to the new one in order to compromise the system. (WASC 2009c)

Insufficient authorization results in cases where an application does not perform vast enough authorization checks in order to confirm that the user is performing an action or accessing data in an intended manner. Checking this vulnerability with a web application scanner is tricky due to fact that modeling system privileges into the web scanner is often impossible. However, for example in some cases where documents are accessed by altering URL parameters such as ID’s, modifying the parameters might grant access to a document otherwise inaccessible and this can be detected using a scanner. (WASC 2009i)

Insufficient session expiration vulnerability occurs in a case where a web application allows an attacker to reuse old session identifiers or credentials for authorization purposes. This is enabled when a web application does not correctly handle session expiration and exposes a site to attacks that steal or reuse user’s authorization tokens. Session should be invalidated by the web application after a predefined time of inactivity has occurred, usually referred to as timeout, or by enabling the user to invalidate their own session as a result of logging out. (WASC 2009j)

Session fixation is a vulnerability where an attacker forces a user’s session ID to an explicit value. The fixing of the user’s session ID is achieved through Cross-site scripting exploits or reusing previously made HTTP requests, vastly depending on the target site’s architecture. After the attacker has successfully fixed a user’s session identifier, they will simply wait for the user to log back to the system and after they’ve done so, use the set identifier to hijack their session and privileges to the system. (WASC 2009n)

3.3.3 Client-side attacks

Client-side attacks refer to attacks that target the users of a web application through their own system. Client-side attacks allow the attackers to gain information without having to worry about the heavily protected server-side application, as some of the web application users are prone to client-side attacks due to lack of proper anti-virus, firewall or anti-spyware on their client-side hardware. The traditional examples of introducing spyware or keylogging methods are not something that the scanner are generally unable to detect, but on the other hand the scanners can be useful in detecting some of the more advanced types of client-side attacks which take place in between the client and the server. (Oriyano & Shimonski 2012, p. 25-26)

Content spoofing is an attack where the attacker injects a malicious payload to the client that is treated as legitimate content of a web application. This is most often enabled by either text only content spoofing, in which the attacker is able to alter the text content of a target page to their liking, or cases of markup reflected content spoofing, where web pages that are using dynamically built html are attacked in a way that the generated html is compromised by injecting malicious code into the parameters. This attack essentially exploits the trust relationship between the user and the web application and is often used in order to create fake web pages including login forms, false press releases etc. This vulnerability is often tied to the cross-site scripting vulnerability. (WASC 2009b)

Cross-site scripting is an attack where malicious code is echoed to and executed by the client’s browser. When the code gets executed, it will run within the security context of the hosting web site and thus has the ability to read, modify and transmit multiple types of sensitive data of the browser, such as cookie information or even lead to browser redirection to harmful websites for even more malicious content and exploitation. Cross-site scripting can be classified into three types: Persistent, Non-persistent and DOM-based attacks. Persistent attacks refer to cases where the attack payload is stored on the server side, Non-persistent to the cases where the attack payload is stored on an external URL, and the DOM types to cases where the attacker abuses the runtime embedding of attacker data in the client side from within a page that is served via the web server. (WASC 2009d) XSS vulnerabilities are ranked 7th on the OWASP top-10 listing of 2017, and as such are one of the most severe threats to modern web application security. (OWASP org. 2017a)

HTML injection is an attack where an attacker can control an input point through which they can inject malicious HTML code into a vulnerable web page. This can lead to multiple various consequences such as theft of cookies, or more generally the alteration of the legitimate page content. This most often occurs when the user input is not correctly sanitized and the output is not encoded, which is essentially easily detectible by the web application scanners. (OWASP org. 2014a)

Cross-site request forgery is an attack where an end user is forced to execute unwanted actions on a web application where they currently have an active session. CSRF attacks target specifically state-changing request and as such do not involve data thievery, since the attacker has no access to see the server response to the forged request. It’s sometimes possible that the CSRF stack is stored on the vulnerable site, accomplished by simply storing and IMG or IFRAME tag in a field that accepts html, or by a more complex XSS attack. This amplifies the severity of the attack as it increases the likelihood of the victim visiting and viewing the malicious page while also ensuring that the victim is authenticated to the site already. (OWASP org, 2018)

3.3.4 Command execution

Command execution vulnerabilities refer to cases where the attackers find a way to execute arbitrary code on your servers in order to compromise them. Remote code execution is one of the most severe security issues, as it can be used to obtain remote control over a machine and even the entire system. After the attackers gain access to the system, they will thrive to escalate their privileges on the server in order to install malicious scripts or backdoors for later exploitation. (Sommestad et al. 2012; Hacksplaining 2020)

Format string attack occurs when string formatting library features are used to access other memory space, through which the flow of the application can be altered. This vulnerability occurs when user-supplied data is used directly as a formatting string input for certain C or C++ functions, such as “fprintf”, “printf”, “sprint” etc. An attacker may pass a format string consisting of “printf” conversion characters, such as “%f”, “%p”, “%n” etc. as a parameter value to the web application, and this may lead to execution of arbitrary code on the server side, reading other data values off the execution stack or even cause segmentation faults and software crashes. Attacks of this type are easily producible and can be easily executed by a web application scanner. (WASC 2009f)

LDAP injection is an attack that aims to exploit web sites that construct LDAP statements based on inputs supplied by the users. Lightweight Directory Access Protocol (LDAP) is an open-standard protocol that is used for querying and manipulating X.500 directory services. As the protocol is being run over internet transport protocols, web applications can use user-supplied inputs in order to create custom LDAP statements for dynamic web page requests. The vulnerability occurs in cases where the user-supplied input is not sanitized strictly enough, allowing the attackers to alter the construction of an LDAP statement. The forged statement then runs with the same privileges as an unaltered one, leading to compromise of rights to manage anything inside the LDAP tree. The advanced exploitation techniques available in SQL injection can also be applied to LDAP injection to some extent. (WASC 2009k)

OS command injection is an attack that aims to execute unauthorized operating system commands. It’s essentially a result of mixing untrusted code and untrusted data, and possible only when an application accepts untrusted input data in order to build OS commands in an insecure manner without proper data sanitization or proper calling of external programs. Since the malicious commands produced by the attacker are executed under the privileges of the component that primarily executes these commands, the attacker can leverage this vulnerability in order to gain access or damage unreachable parts of the software, such as OS directories and system files. (WASC 2009l)

SQL injection is an attack that is aimed to exploit applications that construct SQL statements based on user-supplied input data. The vulnerability compromises the logic of SQL queries sent to the database, and as such can grant the attacker control of all database resources that are accessible by the user, including the ability to execute commands on the host system. SQL injection can be categorized into two subcategories: traditional SQL injection where the errors provided by the SQL backend give information to the attackers and give them valuable information regarding their exploits, and cases of Blind SQL injection where detailed error messages are not provided to the attacker and the information about the success of the query must be gathered by other means, such as differential timing analysis or the manipulation of the user-visible state of the application. Multiple reliable exploits for SQL injection attacks already exist, and web application scanners generally have a very good level of detection related to uncover underlying SQL injection flaws in the system. (WASC 2009o)

SSI injection is a server-side attack that enables an attacker to send malicious code into a web application that is later executed locally by the web server. It essentially exploits the web application’s lack of sanitization of user-supplied data before it is inserted into a server-side interpreted HTML file, and is most common in websites where a web application inserts user-supplied data into the source of the web page, such as message boards or content management systems. This vulnerability essentially enables the attacker to execute arbitrary OS commands or include a restricted file’s contents to the page once it is served the next time. (WASC 2009p)

XPath injection is an attack that focuses in exploiting applications that construct XML Path language queries based on user input in order to query or navigate XML documents. The syntax of XPath is somewhat similar than that of SQL and it’s possible to form SQL-like queries using XPath. In case the unsafe user input is embedded to the XPath query, this may cause data injection to the query resulting it to perform unintentionally. Much like the SQL injection, XPath injection can be easily detected using a web application scanner that includes a pre-configured set of reliable XPath injection attacks. (WASC 2009r)

HTTP header injection / response splitting, also referred to as CRLF (Carriage Return and Line Feed) vulnerability occurs when data enters a web application through an untrusted source, most often a HTTP request, after which the data is included in an HTTP response header which is sent to the user without being validated for malicious content. To mount a successful exploit, an application must allow input that contains both CR and LF characters into the header, while also having a vulnerable underlying platform. The CR and LF characters enable attackers to control the remaining headers and body of the response that the application tries to send, while also allowing them to create additional responses entirely constructed by them. Successful HTTP header injection and response splitting causes a flaw for executing more related attacks, such as XSS and page hijacking. (OWASP org. 2020a)

Remote file includes refers to a vulnerability that is designed to exploit dynamic file include mechanisms of web applications. Almost all web applications support file inclusion, and in cases where the application receives a path to a file as input for a web page and fails to sanitize it properly, an attacker can modify the value to direct the application into executing remote files with malicious code. As the malicious code is then run by the server, in case the file include is not properly protected, the malicious code can compromise the entire system. (WASC 2009m)

Local file includes is a vulnerability that occurs when a web application downloads and executes a remote file. This is most common in cases where the application uses a path to a file located on the server as an input. If an attacker manages to upload a malicious file to the server, modifying the target file path value in the code can trick the server into executing the malicious file. In a worst-case scenario, this can lead to information disclosure, remote code execution or even XSS vulnerabilities. (Netsparker ltd. 2019a)

Potential malicious file uploads present a serious threat to various applications. The first step in multiple attack vectors is to be able to store malicious code on the target server, after which the only task needed is a way to get the code executed. Unrestricted file uploads open a way to achieve the first part without much effort, and the consequences of this vulnerability vary from complete system takeover to client side attacks, and the sheer range of the possible consequences is in-line with the high impact rating of this vulnerability. In order to protect against this kind of vulnerability, an application should always analyze every interaction a web application has with files and carefully consider about what kind of processing and interpreting is safe. (OWASP org. 2020e)

3.3.5 Information disclosure

Information disclosure or sensitive data exposure has been one of the most common impactful attack vectors over the last few years. This vulnerability is possible in all cases of software where sensitive data is being handled, such as passwords, credit card numbers, health records or other personal or business information. This kind of data needs to be properly protected, particularly in cases it’s data that’s treated as sensitive data in terms of the EU GDPR regulation. When assessing if an application is vulnerable for information disclosure types of attack, it’s important to make sure that all of the used transmission protocols are encrypted and all traffic between the system hardware such as servers, load balancers and backend systems is verified. Other points of interest are the used cryptographic algorithms, the use of crypto keys throughout the system and the enforcing of encryption through user agent security directives and headers. This vulnerability is ranked as the 3rd most severe in the most recent OWASP top 10 listing from the year 2017

(OWASP org. 2017a), and as such is on high priority list when it comes to assessing the features of a web application scanner.

Directory indexing is an automated web server function, that lists all of the files located within the requested directory in case the normal base file, such as index.html is not present. Traditionally when the user requests the main page of a website, they type in the URL using the domain name and the requested page name. The web server then processes this request and searches the root directory for the specified document and display it to the client. In cases that this page is not present, the server will run a dynamic directory listing and send the result to the client, equivalent to that of a Unix ls command within the directory and showing all of the results in the resulting HTML. This can unintentionally lead to leakage of possible sensitive data within the directory to the attacker that they aren’t meant to be able to see, such as backup, temporary or hidden files within the system. (WASC 2009e)

Information leakage is an application vulnerability where the application unintentionally leaks sensitive data, such as technical details, environment or user specific data of the system. This sensitive data can then be used by the attacker in order to exploit the application, the running environment or the system users. This is most often a result of HTML or script comments which contain sensitive information about the system’s functionality, misconfiguration of the server or application or cases where the page responses for valid and invalid data are different. (WASC 2009g)

Path traversal, also known as dot-dot-slash, directory traversal or backtracking is an attack that aims to gain access to files outside the web root folder. This is often achieved by manipulating the load file path variables with various “dot-dot-slash” sequence variations in order to access files otherwise inaccessible through the system. These files could include system source code or otherwise critical information such as configuration files, which give the attacker valuable information about the application’s functionality in order to exploit the system further. (OWASP org. 2020d)

Predictable resource location vulnerability, also referred to as insecure indexing occurs when server-side resources, such as admin pages, file backups, uploaded files or system logs are located in easy-to-guess locations on the server. Since the data resources are referenced through auto-incremented primary keys, it’s sometimes easy for the attacker to guess other valid values by brute force methods and gain access to these critical information files. Once again, the information within these files is valuable to the attacker, as it gives them valuable information about the structure and functionality of the web application based on which they can construct better attacks. (OWASP org. 2013)

Insecure HTTP methods is a vulnerability that takes advantage of the methods supported by the HTTP that have a possibility to be used for nefarious causes. PUT method allows a client to upload files to the web server and as such opens a possibility for uploading malicious files. DELETE method allows a client to delete files located on the web server, which can be exploited by the attacker in order to deface a website. CONNECT method could allow the attacker to use the website as a proxy and lastly the TRACE method reports back to the client every string that has been sent to the server and is intended as a tool for debugging process, but it could be used by the attackers in order to create a Cross Site Tracing attack. (OWASP org. 2020c)

Default web server files often refer to default installation or welcome pages which are pre-installed on servers such as the Microsoft IIS or Apache. This most often indicated that the server is newly installed and is yet to be properly configured before the actual use. In often cases, these servers are rarely monitored or patched, and provide attackers with an easy target that is unlikely to be spotted by the corresponding administrators. (Acunetix 2020b; Beyond Security 2020)

Testing and diagnostics pages refer to a vulnerability, where the testing and diagnostics pages aimed strictly at helping developers in testing their code or debug sections of application are visible and accessible through the directory. These pages often contain a lot of sensitive information related to the system’s functionality and as such the access to these pages should always be restricted. One of the common examples for this kind of page are the ASP.NET diagnostics pages, which are aimed at identifying environment problems after production deployment. (Acunetix 2020a)

Internal IP address disclosure, sometimes also referred to as private IP address disclosure is a vulnerability that occurs when an attacker manages to determine the private internal IP address of the system which is usually a lot more difficult to find as the public IP address which is typically protected by a firewall using NAT protocol to direct the public address to the server's internal address. The internal IP address is often used by the internal system but leaking it to a remote system and from there to an attacker can aid in executing various network-layer attacks in order to compromise the organization's internal infrastructure. The private IP address is most often leaked in scenarios where it's included in HTTP responses, debug messages or used in back-end server load balancing systems, and as stated before it should always be masked in order to prevent these types of attacks. (Chapple & Seidl 2018, pp. 166; Portswigger 2020)

3.4 Development environment support

The development environment support in this case refers to the web application security scanner's ability to function across various types of web applications as well as different development environments. While the scanners can detect a vast amount of cross-platform vulnerabilities, their actual functionality is often defined by their support for various web application architectures. In addition to being able to analyze the system, it's also vital for the scanners to be able to be integrated to the development lifecycle, as this is essentially one of the most fundamental advantages of automated penetration testing over traditional methodologies.

3.4.1 Architectural support

Architectural support focuses in examining the web application scanner's support for different architectures and so-called scan barriers. Scan barriers are barriers that prevent the scanner from working in the target environment, for example security measures such as CAPTCHA codes and CSRF tokens designed to prevent various types of attack vectors from occurring. The architectural support points selected for this study are based mainly on the scan barriers enlisted in the WAVSEP DAST 2017/2018 benchmark with a few additions and modifications. (Chen 2017)

Single page application support, or support for multiple domains is crucial as the current trend of development leans towards creating single page applications. SPAs consist from an individual page that is updated independently on each user’s action, which removes the need to reload the entire page as in classic web application. (Joseph 2015, p. 29-30) Instead of reloading the site, the state is updated by AJAX and detecting the change on the application can be tricky even for the modern web application scanners.

Custom authentication headers and cookies refer to cases where an application requires the user to be identified by customized headers or cookies, as the same applies for scanners in order to scan and test the system. These are often session related information which is managed in web applications through various different methods such as URL based and embedded session ID’s, hidden post fields which store the session ID information within the form fields that are submitted to the application and lastly cookies which preserve knowledge of the client browser and can technically also store information beyond single dynamic sessions, referred to as “persistent cookies”. (Kumar et al. 2014) The web application scanner’s ability to set custom cookie and header values is crucial as it enables it to use the target application with a customizable level of access and authority defined through the pre-defined authentication values.

CAPTCHA support is required in cases where the application is protected by a CAPTCHA verification designed to identify whether the user is a human or an automated system. (Von Ahn et al. 2008, p. 1465) At it’s very core, It’s the purpose of a CAPTCHA check to prevent the scanner from submitting a form or accessing a feature of the application, and as such it’s often a hard barrier for the scanner and entirely prevents it from assessing protected parts the target application. Cases where the checks are present can be managed for example by manually flagging the sites containing CAPTCHA checks and allowing a user to manually enter the required values or patterns during a scan. (Acunetix 2009)

Field value autofill support is tightly coupled with the web application scanner’s ability to handle websites with various forms. Form fields contain custom checks in many cases, requiring the input to fulfil certain criteria and this can be difficult for the scanners to handle. For example in case of Netsparker, the default set of preconfigured values is enough for traversing most of the forms, but in cases where the required form values require custom data types such as IBAN numbers or different kinds of ID's, the user can easily configure some additional values to the library that the scanner will then be able to use. (Netsparker 2020)

3.4.2 Usability

Usability assessment criteria in this thesis focuses around the features of the web application security scanner that are related to the actual process of analysing a web application with the scanner apart from actual scanner configuration to overcome scan barriers. This aspect mainly focuses in the scanner's ability to provide the using personnel with valuable information about the state of the system, integrate with various tools related to the software development lifecycle. The basis for this evaluation is mainly in the WASC web application security scanner evaluation criteria with some additions. (WASC 2009r)

Scanners often present the results of the scans in their user interface, but the ability to output different types of reports is essential for spreading the knowledge about the state of security of the software throughout the developing company. There are various types of reports that the scanner should be able to output: Firstly, the Executive summary which provides a concise recap of the results of a scan and allows the tester to determine the severity of the results easily and without much effort. Secondly, a technical detail report which provides technical information related to the issues, including all of the used request and provided response data, as well as a list of all URL and host information so that the developers know exactly how to reproduce the identified issue, and lastly the Delta report which essentially focuses in comparing results of past scans and provides trends and development data over time. All these reports should also be exportable from the system user interface in some form that is easy for the user to handle, such as pdf or html. (WASC 2009r)

Another key aspect to assess is the actual scanning procedure. Scheduled scanning is often the most convenient way to maintain a steady monitoring framework. In addition to scheduled scanning, the users should also be able to pause and resume an ongoing scan so that the tester can resume the scan at later date instead of having to start all over again. Real-time scan status monitoring is also an important feature, as this could include information valuable to the tester such as the types of tests being conducted as well as the completion percentage of a scan and the overall scan history of the target application. Another important aspect to look for is scan logging, as logs produced by the scanner help the testers debug the scanner's behaviour when encountering problems. The tools should also have support for running multiple simultaneous scans since organizations often have multiple web applications or in cases where different testers want to assess different parts of the same system simultaneously. Lastly, the scanner should support multiple users, since scanning the web application could easily be distributed across the company to different testers. The way the application handles multiple users depends on the nature of the application, as some require the user to have the web application scanner installed on their local computer, while others provide a centralized web-based interface which multiple users can use simultaneously. (WASC 2009r)

Regarding the user experience point of view, it's desirable that the web application scanner has a client application with a graphical user interface. In addition to a graphical user interface of the desktop-application or web-based application, some tools provide a traditional command line interface for the more experienced users with advanced command options and configuration files used to define the scan settings. (WASC 2009r) These interfaces vary a lot between different types of applications and assessing their usability could be considered as a preference question and as such is not assessed in the context of this study.

One of the best ways to integrate the web application scanner into the development process is by extending the scanner's functionality by integrating it with other systems used in the development process, including ticketing and bug-tracking systems such as Jira, browser automation systems such as Selenium and continuous integration systems, such as Jenkins. In addition to these external system integrations, having a dedicated API for the web application scanner provides the company with a powerful interface for carrying out custom integration projects. (WASC 2009r)

Last parameter considered in the section of usability is the actual cost of the scanner. Most often commercial web scanners have quire complex pricing models, differentiating between various license types and the amount of target websites planned for scanning procedures. However, this is an interesting aspect to include especially when reflecting the results of commercial web scanners on the open source alternatives.

4 RESULTS

The study was conducted using the assessment criteria described in section 3 of this thesis and is entirely based on the documentation available from the product, provided by the vendors or related communities. The results for each individual tool assessment with corresponding sources and an overview of the tool are available in the appendix 1 of this thesis. In cases where information related to supporting specific assessment criterion was not found (N/A), criterion is treated as unsupported in the context of this study. This is due to fact that product documentation was used as a main source of information and it does not provide any information related to unsupported features by default. The table legend is as follows: ✓: Supported, ✓: Partially or insufficiently supported, ✗: N/A or Not supported.

Table 7: Results of the study.

	Acunetix Premium	Burp Suite Enterprise	NetSparker Team	AppSpider Enterprise	Arachni	ZAP
Vendor / Source	Acunetix	PortSwigger ltd.	Netsparker ltd.	Rapid7 ltd.	Sarosys LLC	ZAP dev team
Authentication	4/5	1/5	3/5	4/5	3/5	3/5
Brute force	✓	✗	✓	✓	✓	✓
Insufficient authentication	✗	✗	✗	✓	✗	✓
Weak password recovery validation	✓	✗	✗	✗	✗	✗
Lack of SSL protection on login pages	✓	✗	✓	✓	✓	✓
Auto-complete not disabled on password fields	✓	✓	✓	✓	✓	✗
Authorization	1/4	0/4	2/4	4/4	1/4	3/4
Credential and session prediction	✗	✗	✓	✓	✗	✓
Insufficient authorization	✗	✗	✓	✓	✗	✓
Insufficient session expiration	✗	✗	✗	✓	✗	✗
Session fixation	✓	✗	✗	✓	✓	✓
Client-side attacks	7/7	4/7	5/7	5/7	4/7	4/7
Content spoofing	✓	✗	✓	✗	✗	✗

Reflected XSS	✓	✓	✓	✓	✓	✓
Persistent XSS	✓	✓	✓	✓	✓	✓
DOM-based XSS	✓	✓	✓	✓	✓	✓
Cross-Frame Scripting	✓	✗	✗	✗	✗	✗
HTML Injection	✓	✗	✗	✓	✗	✗
CSRF	✓	✓	✓	✓	✓	✓
Command execution	11/11	10/11	7/11	11/11	8/11	9/11
Format string attack	✓	✗	✗	✓	✗	✓
LDAP injection	✓	✓	✗	✓	✓	✓
OS command injection	✓	✓	✓	✓	✓	✓
SQL injection	✓	✓	✓	✓	✓	✓
Blind SQL injection	✓	✓	✓	✓	✓	✓
SSI injection	✓	✓	✗	✓	✗	✓
XPath injection	✓	✓	✗	✓	✓	✓
HTTP header injection / response splitting	✓	✓	✓	✓	✓	✓
Remote file includes	✓	✓	✓	✓	✓	✓
Local file includes	✓	✓	✓	✓	✓	✗
Potential malicious file uploads	✓	✓	✓	✓	✗	✗
Information disclosure	7/8	5/8	7/8	6/8	6/8	6/8
Directory indexing	✓	✓	✓	✓	✓	✓
Information leakage	✓	✓	✓	✓	✓	✓
Path traversal	✓	✓	✓	✓	✓	✓
Predictable resource location	✗	✗	✓	✓	✓	✓
Insecure HTTP methods enabled	✓	✓	✓	✗	✓	✓
Default web server files	✓	✗	✓	✗	✗	✗
Testing and diagnostics pages	✓	✗	✗	✓	✗	✗
Internal IP Address disclosure	✓	✓	✓	✓	✓	✓
Architectural support	5/5	4.5/5	5/5	5/5	5/5	4/5

SPA support	✓	✔	✓	✓	✓	✓
Custom authentication headers	✓	✗	✓	✓	✓	✓
Custom authentication cookies	✓	✓	✓	✓	✓	✓
CAPTCHA support	✓	✗	✓	✓	✓	✗
Field value autofill support	✓	✓	✓	✓	✓	✓
Usability	16/17	15/17	15.5	17/17	14/17	12/17
Executive summary	✓	✓	✓	✓	✓	✓
Technical detail report	✓	✓	✓	✓	✓	✓
Delta report	✓	✓	✓	✓	✗	✗
Compliance reports	✓	✗	✓	✓	✓	✗
Report exporting	✓	✓	✓	✓	✓	✓
Scheduled scanning	✓	✓	✓	✓	✓	✔
Scanning pause and resume	✓	✓	✓	✓	✓	✔
Real-time scan monitoring	✓	✓	✓	✓	✓	✓
Scan logging	✓	✗	✓	✓	✓	✓
Multiple simultaneous scans support	✓	✓	✓	✓	✓	✗
Multi-user support	✓	✓	✓	✓	✓	✗
GUI	✓	✓	✓	✓	✓	✓
CLI	✗	✗	✗	✓	✓	✓
Ticketing / bug tracking system integration	✓	✓	✓	✓	✗	✓
Browser automation integration	✓	✓	✔	✓	✗	✓
CI integration	✓	✓	✓	✓	✓	✓
API	✓	✓	✓	✓	✓	✓
Cost	$6995/year	$3999/year	On premise	On premise	Free	Free
Overall	50/56 89%	35,5/56 63%	43,5/56 78%	51/56 91%	40/56 71%	41/56 73%

5 DISCUSSION AND CONCLUSIONS

Overall results have quite significant differences, and the two scanners to stand out are the Acunetix and the AppSpider which have the highest overall scores. When looking at the results for assessing threat detection capabilities, most of the tools have quite similar results when examining client-side attacks, command execution and information disclosure, but this is simply due to the fact that the criterion included in these sections consist mainly from various attack vectors. In comparison, when assessing Authorization and Authentication, there is a lot more dispersion between the results. It is understandable, as detecting the authentication and authorization issues through web scanners is generally treated as impossible, but some of these tools do provide methods for automating parts of the work required to uncover vulnerabilities related to these topics. The overall level of architectural support in the results is very similar across all assessed tools despite their initial cost or provider, but it’s obvious that the free tools fall short of the commercial scanner’s results in this category.

One of the more interesting takes related to an individual scanner was the division of features and tools between different Burp Suite editions. The Enterprise version offers only the web application scanner without the manual tools and therefore lacks support for various methods as they are intended to be uncoverable using some of the manual tools. If the enterprise edition would also provide these manual tools for the purposes of testing personnel in the company, the coverage of assessment criteria would resulted in at 51 points which is on-par with the AppSpider that boasted the top-score of this assessment. If one would be interested in purchasing an overall solution for both scanning and manual testing the Burp Suite would be a very interesting product, but it’s a shame that the enterprise version lacks the manual tools essential for conducting a throughout penetration testing process.

Information availability was vastly different throughout the assessed tools. Most of the commercial tools provided very large amounts of information through their documentation and websites, but in order to assess the tool’s support for an individual feature proved to be cumbersome and very time-consuming, as the documentation was often very decentralized. Acunetix, Burp Suite and Netsparker provided a convenient enlisting of vulnerabilities that the tool can identify, while the AppSpider relied on more brochure-like concept where all vulnerabilities detectable by the tool are listed on a simplified pdf document. When studying the architectural support, the information was even more hard to find and combine and often required using google search towards site contents in order to find relevant information. The free tools, on the other hand surprised with how well the features are documented. Both assessed tools had very good level of documentation related to both the general vulnerability detection features and the architectural support features of each tool through the official pages, but also through the provided GitHub pages as they provide excellent insights on individual feature development and ticketing history. Arachni also referenced the results of the WAVSEP DAST benchmark in the documentation to provide a quick overview of the tool's functionalities and features.

The most difficult part of the assessment process was determining if a feature is not supported opposed to the fact that it's just not documented. was difficult. There was rarely any information directly pointing out that the feature is not supported or planned, and generally this information was only available by inspecting support forums for questions related to the topics and analyzing the staff responses. This proved to be quite challenging as well, as the support forum questions generally are marked as solved based on the information available at that point and are not updated accordingly even if the feature is added later.

Overall, comparing different tools is very hard, and as such work required to determine the most suitable for an application and the corresponding development environment can prove to be difficult. There are few significant reasons behind this: Firstly, many of the assessed scanners use different vocabulary for describing similar features. For example, HTTP header injection can also be referenced to as Response-splitting or CRLF injection, which in the context of the scanner documentation all refer to the same type of vulnerability. Secondly, the sheer amount of information related to a product is often decentralized to different forms of documentation, such as online documentations, brochures, guidebooks, videos and blog posts making it difficult to get a clear overlook of the supported features. And lastly, it's very hard to establish a functional criteria of evaluation for this type of assessment. The WASC assessment criteria for DAST tools that was used as a reference for this study and is also widely used throughout other similar studies such as the OWASP and the WAVSEP DAST benchmarks, actually dates back to 2009 and as such some of the assessment criterion presented in that framework are helplessly outdated today. (WASC 2009r; OWASP org. 2020f; Chen 2017) A good example related to this is the 'Auto-complete not disabled on password fields' assessment criteria, which is essentially useless today since majority of browsers will override any use of 'autocomplete=”off”' since early 2014 with regards to password forms and as a result it's commonly not recommended to disable this feature. (OWASP org. 2014b)

In conclusion to DAST scanner comparison being both difficult and exhaustive, the simplest way to compare these tools is through examining the results of a benchmark or comparison study such as this one. The WAVSEP DAST benchmark is one of the most comprehensive studies available related to the topic and provides a very vast criteria of evaluation for covering and comparing as many aspects of these tools and their functionalities as possible. The results of the WAVSEP are also conveniently gathered to the sectoolmarket.com webpage (Chen 2015) and are very easy to compare. The main problem with this kind of study is however that the tools are constantly evolving, and the results presented in the most recent WAVSEP DAST benchmark are collected during a time period from 2011 to 2016. (Chen 2017) The results presented in this thesis are not as encompassing as the results of WAVSEP study, but they provide a good overview of the assessed scanners and indirectly reflect the state and availability of documentation related to them.

6 SUMMARY

The attacks against web applications have increased drastically in the past years and the severity of breaches has increased as more and more of our sensitive information is being handled through web applications and services. Penetration testing is an effective measure to test web applications for such vulnerabilities, but as it’s often tedious and time-consuming manual labor, dynamic application security scanners have been introduced to gain a quick overall assessment about the current state of security of a web application and can be easily integrated to the SDLC for easy monitoring. Today, there are multiple different DAST tools available, both commercial and open source and while all of them are developed for the same purpose, their set of features in terms of functionality and usability is often very different from each other and need to be closely examined as a part of the process of selecting such tool. However, assessing and comparing these tools is very hard as universal assessment framework does not exist, searching information related to these tools in order to compare them takes a lot of time and dedication, and even so, the results of the search are not valid after a couple of years as the tools improve current and develop new features to respond to the evolving set of web application security vulnerabilities.

When assessing and comparing the currently available DAST solutions, they do provide excellent possibilities for automating traditional penetration testing processes and provide easy means for integrating them to the SDLC. While there are differences among the assessed tools, the results suggest that a clear classification between these assessed tools is hard to provide. While one of the evident classifications present in the selection of assessed tools is the commercial and open-source diversion, the results suggest that this classification is not reflected in the results of the study. It must be noted that this study’s results are based solely on the existing materials and documentation and the actual results are not based on benchmarking the subject applications. It would be an interesting topic for a future study to investigate how well these results are reflected in a benchmark study. Another interesting view would be to conduct a case-study in order to examine the process of selecting any of the examined DAST tools for a company, and the process of integrating the selected tool into an actual software development environment and the overall effects of this integration after it’s been deployed.

REFERENCES

Aarya, P., Rajan, A., Sachin, K., Gopi, R. & Sreenu, G. 2018. Web Scanning: Existing Techniques and Future. *Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, doi: 10.1109/ICCONS.2018.8662934

Acunetix, 2009, New Acunetix WVS V6.5 build; better support for CAPTCHA and modern authentication mechanisms, online, available at <https://www.acunetix.com/blog/releases/better-support-for-captcha-and-modern-authentication-mechanisms/>, accessed 19.3.2020

Acunetix, 2020a, Web vulnerabilities index: ASP.NET diagnostic page, online, available at: < https://www.acunetix.com/vulnerabilities/web/asp-net-diagnostic-page/>, accessed 4.4.2020

Acunetix, 2020b, Web vulnerabilities index: Web server default welcome page, online, available at: < https://www.acunetix.com/vulnerabilities/web/web-server-default-welcome-page/ >, accessed 4.4.2020

Antunes, N. & Vieira, M. 2014. Penetration Testing for Web Services. *Computer, vol. 47*(2), pp. 30-36. doi:10.1109/MC.2013.409

Arkin, B., Stender, S., McGraw, G. 2005. Software penetration testing. *IEEE Security & Privacy, vol. 3*(1), pp. 84-87. doi:10.1109/MSP.2005.23

Beyond Security, 2020, Finding and Fixing Vulnerabilities in Microsoft IIS Default Page, a Low Risk Vulnerability, online, available at: < https://beyondsecurity.com/scan-pentest-network-vulnerabilities-microsoft-iis-default-page.html >, accessed 4.4.2020

Chapple, M., & Seidl, D. 2019. *CompTIA PenTest+ Study Guide: Exam PT0-001*, [eBook] John Wiley & Sons, Inc. ISBN:9781119549420 472 pages

Chen, S. 2015, Price and Feature Comparison of Web Application Scanners, online, available at <http://www.sectoolmarket.com/price-and-feature-comparison-of-web-application-scanners-unified-list.html >, accessed 6.4.2020

Chen, S. 2017, The 2017/2019 WAVSEP DAST Benchmark: Evaluation of Web Application Vulnerability Scanners in Modern Pentest/SSDLC Usage Scenarios, 2017, online, available at: <http://sectooladdict.blogspot.com/2017/11/wavsep-2017-evaluating-dast-against.html> accessed 22.01.2020

Crispin, L. & Gregory, J. 2009. *Agile testing: A practical guide for testers and agile teams.* Upper Saddle River, NJ: Addison-Wesley. ISBN: 9780321534460, 533 pages

CWE, 2020, CWE-640: Weak Password Recovery Mechanisms for Forgotten Passwords, online, available at <https://cwe.mitre.org/data/definitions/640.html>, accessed on 4.4.2020

Fong, E., Gaucher, R., Okun, V. & Black, P. 2008. Building a Test Suite for Web Application Scanners, *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, pp. 478-478. doi: 10.1109/HICSS.2008.79

Hacksplaining, 2020, Protecting against command execution attacks, online, available at < https://www.hacksplaining.com/prevention/command-execution >, accessed 4.4.2020

Huang, H., Zhang, Z., Cheng, H. & Shieh, S. W. 2017. Web Application Security: Threats, Countermeasures, and Pitfalls. *Computer*, *50*(6), pp. 81-85. doi: 10.1109/MC.2017.183

Joseph, R. 2015. Single Page Application and Canvas Drawing. *International Journal of Web & Semantic Technology, 6(1)*, pp. 29-37. doi: 10.5121/ijwest.2015.6103

Ko, M., Osei-Bryson, K. & Dorantes, C. 2009. Investigating the Impact of Publicly Announced Information Security Breaches on Three Performance Indicators of the

Breached Firms. *Information Resources Management Journal, 22*(2), pp. 1-21. doi:10.4018/irmj.2009040101

Kortelainen, H., Happonen, A., Hanski, J. (2019), "From asset provider to knowledge company - transformation in the digital era", In Lecture Notes in Mechanical Engineering, ISSN: 2195-4356, pp. 333-341, doi: 10.1007/978-3-319-95711-1_33

Kortelainen, H., Happonen, A., Kinnunen, S-K. (2016), Fleet Service Generation – Challenges in Corporate Asset Management, Lecture Notes in Mechanical Engineering, Springer, pp. 373–380, doi: 10.1007/978-3-319-27064-7_35

Mburano, B. & Si, W. 2018. Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark. *26th International Conference on Systems Engineering (ICSEng)*, Sydney, Australia, pp. 1-6. doi: 10.1109/ICSENG.2018.8638176

Mcgraw, G. 2004. Software security. *IEEE Security & Privacy, 2*(2), pp. 80-83. doi:10.1109/MSECP.2004.1281254

Netsparker ltd. 2019a, Local File Inclusion Vulnerability, online, available at <https://www.netsparker.com/blog/web-security/local-file-inclusion-vulnerability/>, accessed 22.3.2020

Netsparker ltd. 2019b, OWASP Top 10 Web Application Vulnerabilities, online, available at <https://www.netsparker.com/blog/web-security/owasp-top-10/>, accessed 26.3.2020

Netsparker ltd. 2020, Configuring Predefined Web Form Values in Netsparker Web Security Scanners, online, available at <https://www.netsparker.com/blog/docs-and-faqs/configure-predefined-web-form-values-web-vulnerability-scanner/>, accessed 23.3.2020

Oriyano, S. & Shimonski, R. 2012. *Client-Side Attacks and Defense,* [eBook] Syngress Publishing, ISBN:1597495905 296 pages

OWASP org. 2007, OWASP Top 10 2007, online, available at <https://www.owasp.org/index.php/Top_10_2007>, accessed 3.12.2019

OWASP org. 2013, OWASP Periodic Table of Vulnerabilities – Brute Force predictable Resource Location / insecure Indexing, online, available at <https://wiki.owasp.org/index.php/OWASP_Periodic_Table_of_Vulnerabilities_-_Brute_Force_Predictable_Resource_Location/Insecure_Indexing>, accessed 28.2.2020

OWASP org. 2014a, Testing for HTML Injection (OTG-CLIENT-003), online, available at <https://www.owasp.org/index.php/Testing_for_HTML_Injection_(OTG-CLIENT-003)>, accessed 15.3.2020

OWASP org. 2014b, Testing for Vulnerable Remember Password (OTH-AUTHN-005), online, available at <https://wiki.owasp.org/index.php/Testing_for_Vulnerable_Remember_Password_(OTG-AUTHN-005)>, accessed 15.3.2020

OWASP org. 2017a, OWASP Top Ten project, online, available at <https://owasp.org/www-project-top-ten/>, accessed 4.4.2020

OWASP org. 2017b, A1-Injection, online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A1-Injection>, accessed 02.12.2019

OWASP org. 2017c, A2-Broken Authentication, online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A2-Broken_Authentication>, accessed 02.12.2019

OWASP org. 2017d, A3-Sensitive Data Exposure, online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A3-Sensitive_Data_Exposure>, accessed 02.12.2019

OWASP org. 2017e, A4-XML External Entities (XXE), online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A4-XML_External_Entities_(XXE)>, accessed 02.12.2019

OWASP org. 2017f, A5-Broken Access Control, online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A5-Broken_Access_Control>, accessed 02.12.2019

OWASP org. 2017g, A6-Security Misconfiguration, online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A6-Security_Misconfiguration>, accessed 02.12.2019

OWASP org. 2017h, A7-Cross-Site Scripting, online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A7-Cross-Site_Scripting_(XSS)>, accessed 02.12.2019

OWASP org. 2017i, A8-Insecure Deserialization, online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A8-Insecure_Deserialization>, accessed 02.12.2019

OWASP org. 2017j, A9-Using Components With Known Vulnerabilities, online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A9-Using_Components_with_Known_Vulnerabilities>, accessed 02.12.2019

OWASP org. 2017k, A10-Insufficient Logging & Monitoring, online, available at <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A10-Insufficient_Logging%252526Monitoring>, accessed 02.12.2019

OWASP org. 2018, Cross-Site Request Forgery (CSRF), online, available at <https://wiki.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)>, accessed 22.3.2020

OWASP org. 2020a, HTTP Response Splitting, online, available at <https://owasp.org/www-community/attacks/HTTP_Response_Splitting>, accessed 24.3.2020

OWASP org. 2020b, OWASP Benchmark, online, available at < https://owasp.org/www-project-benchmark/#div-tool_support>,

OWASP org. 2020c, OWASP Web Security Testing Guide – Test HTTP Methods, online, available at <https://github.com/OWASP/wstg/tree/master/document>, accessed 28.3.2020

OWASP org. 2020d, Path Traversal, online, available at <https://owasp.org/www-community/attacks/Path_Traversal>, accessed 6.2.2020

OWASP org. 2020e, Unrestricted File Upload, online, available at <https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload>, accessed 22.3.2020

OWASP org. 2020f, Vulnerability scanning tools, online, available at <https://owasp.org/www-community/Vulnerability_Scanning_Tools>, accessed 27.3.2020

Portswigger ltd. 2020, Private IP addresses disclosed, online, available at: < https://portswigger.net/kb/issues/00600300_private-ip-addresses-disclosed>, accessed 4.4.2020

retire.js / E. Oftedal, 2019, Retire.js, online, available at: <https://retirejs.github.io/retire.js/>, accessed 6.4.2020

Sommestad, T., Holm, H. & Ekstedt, M. 2012. Estimates of success rates of remote arbitrary code execution attacks. *Information Management & Computer Security, 20*(2), pp. 107-122. doi:10.1108/09685221211235625

Sonicwall, 2019, Sonicwall Cyber Threat Report, online, available at <https://www.sonicwall.com/resources/white-papers/2019-sonicwall-cyber-threat-report/>, accessed 6.4.2020

SSL.com, 2019, What is SSL, online, available at <https://www.ssl.com/faqs/faq-what-is-ssl/>, accessed 24.3.2020

Stefinko, Y., Piskozub, A. & Banakh, R. 2016. Manual and automated penetration testing. Benefits and drawbacks. Modern tendency, *13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET),* Lviv, pp. 488-491 doi: 10.1109/TCSET.2016.7452095

Stuttard, D. & Pinto, M. 2011. *The web application hacker's handbook: Finding and exploiting security flaws, 2nd edition* [eBook]. Indianapolis, IN: John Wiley & Sons, Inc. pp. 878.

Symantec ltd. 2019, Internet Security Threat Report 2019 vol. 24, online, available at: https://resource.elq.symantec.com/LP=6819 accessed on 6.11.2019

Von Ahn, L, Maurer, B, Mcmillen, C, Abraham, D. & Blum, M. 2008. reCAPTCHA: Human-based character recognition via Web security measures. *Science (New York, N.Y.), 321*(5895), p. 1465-1468. doi:10.1126/science.1160379

WASC, 2009a, Brute Force, online, available at <http://projects.webappsec.org/w/page/13246915/Brute%20Force >, accessed 24.3.2020

WASC, 2009b, Content Spoofing, online, available at <http://projects.webappsec.org/w/page/13246917/Content%20Spoofing>, accessed 22.3.2020

WASC, 2009c, Credential and Session Prediction, online, available at <http://projects.webappsec.org/w/page/13246918/Credential%20and%20Session%20Prediction>, accessed 27.3.2020

WASC, 2009d, Cross Site Scripting, online, available at <http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting>, accessed 22.3.2020

WASC, 2009e, Directory Indexing, online, available at <http://projects.webappsec.org/w/page/13246922/Directory%20Indexing>, accessed 22.3.2020

WASC, 2009f, Format String Attack, online, available at <http://projects.webappsec.org/w/page/13246926/Format%20String>, accessed 22.3.2020

WASC, 2009g, Information Leakage, online, available at <http://projects.webappsec.org/w/page/13246936/Information%20Leakage>, accessed 20.3.2020

WASC, 2009h, Insufficient Authentication, online, available at <http://projects.webappsec.org/w/page/13246939/Insufficient%20Authentication>, accessed 24.3.2020

WASC, 2009i, Insufficient Authorization, online, available at <http://projects.webappsec.org/w/page/13246940/Insufficient%20Authorization>, accessed 22.3.2020

WASC, 2009j, Insufficient Session Expiration, online, available at <http://projects.webappsec.org/w/page/13246944/Insufficient%20Session%20Expiration>, accessed 25.3.2020

WASC, 2009k, LDAP Injection, online, available at <http://projects.webappsec.org/w/page/13246947/LDAP%20Injection>, accessed 23.3.2020

WASC, 2009l, OS Commanding, online, available at <http://projects.webappsec.org/w/page/13246950/OS%20Commanding>, accessed 24.3.2020

WASC, 2009m, Remote File Inclusion, online, available at <http://projects.webappsec.org/w/page/13246955/Remote%20File%20Inclusion>, accessed 25.3.2020

WASC, 2009n, Session Fixation, online, available at <http://projects.webappsec.org/w/page/13246960/Session%20Fixation>, accessed 22.3.2020

WASC, 2009o, SQL Injection, online, available at <http://projects.webappsec.org/w/page/13246963/SQL%20Injection>, accessed 25.3.2020

WASC, 2009p, SSI Injection, online, available at <http://projects.webappsec.org/w/page/13246964/SSI%20Injection>, accessed 25.3.2020

WASC, 2009q, The WASC Threat Classification v2.0, online, available at <http://projects.webappsec.org/w/page/13246978/Threat%20Classification>, accessed 24.3.2020

WASC, 2009r, Web Application Security Scanner Evaluation Criteria, online, available at <http://projects.webappsec.org/w/page/13246986/Web%20Application%20Security%20Scanner%20Evaluation%20Criteria>, accessed 22.3.2020

WASC, 2009s, XPath Injection, online, available at <http://projects.webappsec.org/w/page/13247005/XPath%20Injection>, accessed 25.3.2020

WhiteHat Security Inc, 2020, Weak Password Recovery Validation, online, available at < https://www.whitehatsec.com/glossary/content/weak-password-recovery-validation >, accessed 4.4.2020

Yongsheng, Z., Cuicui, S., Jing, Y. & Ying, W. 2010. Web Services Security Policy. *International Conference on Multimedia Information Networking and Security, Nanjing, Jiangsu.* pp. 236-239, doi: 10.1109/MINES.2010.223

APPENDIX 1. Results from the study

AppSpider Enterprise

AppSpider Enterprise is an on-premise, dynamic web application security scanner by Rapid7 ltd, designed for scanning web and mobile applications for vulnerabilities by DevSecOps teams and enterprise-wide users. The core of the AppSpider scanner is in the Universal Translator system, which interprets new technologies such as AJAX, HTML5 and JSON which are the most used technologies in today's web and mobile applications and sophisticated attack methodologies. [1]

The company behind the product also delivers other solutions for similar purposes: InsightAppSec offers similar threat detection features when compared to the AppSpider software but is an entirely cloud-based solution instead of an on-premise solution. Rapid7 also provides the Managed AppSec, which is essentially an all-round application security solution where the Rapid7 provides expert support for managing and running the scans and validating the vulnerabilities. [2]

On general level, the AppSpider had an easily accessible and vast documentation which eased the process of assessing this tool. The only problem encountered was that in some cases the documentation referred to the Professional version of the AppSpider scanner, which apparently is not available anymore, and in some cases where you follow links to the AppSpider material [3], the website would actually provide you with material that is labeled as InsightAppSec material [4]. A conclusion was made that since AppSpider Enterprise is the company's oldest and most developed product, it's very likely to support the same set of threat detection as the InsightAppSec software at parts, and also due to the fact that it's essentially the most valuable license available on the AppSpider software that it most likely also provides all of the features that the AppSpider professional does that in the context of this study, the focus will be in the provided material and if it's referenced as a set of features for the AppSpider Enterprise anywhere on the site, it will be treated as such despite the material not specifically stating to be that of AppSpider Enterprise.

	AppSpider	Source

Overview sources		[1] https://appspider.help.rapid7.com/docs/welcome-to-appspider [2] https://www.rapid7.com/products/appspider/download/editions/ [3] https://www.rapid7.com/products/appspider/features/ [4] https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Authentication		
Brute force	Yes	Support for both HTTP and form-based authentication: https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Insufficient authentication	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
Weak password recovery validation	N/A	
Lack of SSL protection on login pages	Yes	Support for checking SSL Strength in general: https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Auto-complete not disabled on password fields	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
Authorization		
Credential and session prediction	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
Insufficient authorization	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
Insufficient session expiration	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
Session fixation	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Client-side attacks		
Content spoofing	N/A	
Reflected XSS	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Persistent XSS	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-

		datasheet.pdf
DOM-based XSS	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Cross-Frame Scripting	N/A	
HTML Injection	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
CSRF	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
Command execution		
Format string attack	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
LDAP injection	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
OS command injection	Yes	Referred to as OS Commanding: https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
SQL injection	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Blind SQL injection	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
SSI injection	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
XPath injection	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
HTTP header injection / response splitting	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Remote file includes	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
Local file includes	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
Potential malicious file uploads	Yes	Referred to as Arbitrary file upload: https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Information disclosure		

Directory indexing	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Information leakage	Yes	Response data and SQL errors: https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Path traversal	Yes	https://appspider.help.rapid7.com/docs/how-to-test-the-web-application-automated
Predictable resource location	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-insightappsec-appspider-attack-types-datasheet.pdf
Insecure HTTP methods enabled	N/A	
Default web server files	N/A	
Testing and diagnostics pages	Yes	ASP.Net misconfiguration: https://appspider.help.rapid7.com/docs/attack-module-details
Internal IP Address disclosure	Yes	https://appspider.help.rapid7.com/docs/attack-module-details
Architectural support		
SPA support	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-feature-brief-appsec-universal-translator.pdf
Custom authentication headers	Yes	https://appspider.help.rapid7.com/docs/http-headers
Custom authentication cookies	Yes	https://appspider.help.rapid7.com/docs/http-headers
CAPTCHA support	Yes	https://appspider.help.rapid7.com/docs/authentication
Field value autofill support	Yes	https://appspider.help.rapid7.com/docs/parameters-training
Usability		
Executive summary	Yes	https://appspider.help.rapid7.com/docs/reporting
Technical detail report	Yes	https://appspider.help.rapid7.com/docs/reporting
Delta report	Yes	https://www.rapid7.com/globalassets/external/docs/download/AppSpider_Enterprise_User_Guide.pdf
Compliance reports	Yes	https://www.rapid7.com/products/appspider/download/editions/

Report exporting	Yes	https://www.rapid7.com/globalassets/external/docs/download/AppSpider_Enterprise_User_Guide.pdf
Scheduled scanning	Yes	https://www.rapid7.com/globalassets/external/docs/download/AppSpider_Enterprise_User_Guide.pdf
Scanning pause and resume	Yes	https://www.rapid7.com/globalassets/external/docs/download/AppSpider_Enterprise_User_Guide.pdf
Real-time scan monitoring	Yes	https://appspider.help.rapid7.com/docs/scan-status
Scan logging	Yes	https://appspider.help.rapid7.com/docs/monitor-an-ongoing-scan
Multiple simultaneous scans support	Yes	https://appspider.help.rapid7.com/docs/installing-appspider-enterprise
Multi-user support	Yes	https://appspider.help.rapid7.com/docs/installing-appspider-enterprise
GUI	Yes	https://appspider.help.rapid7.com/docs/appspider-pro-quick-start-guide
CLI	Yes	https://appspider.help.rapid7.com/docs/installing-appspider-enterprise
Ticketing / Bug tracking system integration	Yes	https://www.rapid7.com/products/appspider/integrations/
Browser automation integration	Yes	https://www.rapid7.com/globalassets/_pdfs/product-and-service-briefs/rapid7-feature-brief-appsec-universal-translator.pdf
CI integration	Yes	https://www.rapid7.com/products/appspider/integrations/
API	Yes	https://appspider.help.rapid7.com/docs/attack-module-api-overview
Cost	N/A	Informed as on-premise at https://www.rapid7.com/products/appspider/download/editions/

Acunetix Premium

Acunetix premium is a web application security scanner provided by a company also titled as Accunetix. The premium version of the scanner is primarily intended for medium-to-large organizations that are required to secure multiple websites and web applications and wish to incorporate the scanning measures to their DevOps and issue management infrastructures Acunetix claims to maintain the best level of security in larger organizations through strong automation and integration and C++ based engine, deemed to be fast and optimized to discover vulnerabilities using as few requests as possible. [1]

Appendix 1

Acunetix provides documentation about their core features mainly in the forms of blog posts which are provided by the professional security engineers working for the company. There is not that much information publicly available regarding the attack modules. Instead, they have provided a list of vulnerabilities detectable by their system as an indexed list which does not differentiate between products. [2] However, according to the documentation, the premium version provides all of the available features of the vulnerability assessment engine, it’s fair to assume that the list contains vulnerabilities detectable by the Acunetix premium scanner. [3]

	Acunetix	Source
Overview sources		[1] https://www.acunetix.com/product/premium/ [2] https://www.acunetix.com/vulnerabilities/web/ [3] https://www.acunetix.com/ordering/
Authentication		
Brute force	Yes	Weak passwords detected using brute force: https://www.acunetix.com/vulnerabilities/web/tag/bruteforce-possible/
Insufficient authentication	N/A	
Weak password recovery validation	Yes	https://www.acunetix.com/blog/articles/password-reset-poisoning/
Lack of SSL protection on login pages	Yes	https://www.acunetix.com/vulnerabilities/web/sensitive-data-not-encrypted/
Auto-complete not disabled on password fields	Yes	https://www.acunetix.com/vulnerabilities/web/password-type-input-with-auto-complete-enabled/
Authorization		
Credential and session prediction	N/A	
Insufficient authorization	N/A	
Insufficient session expiration	N/A	
Session fixation	Yes	Manual confirmation required: https://www.acunetix.com/vulnerabilities/web/session-fixation/
Client-side		

attacks		
Content spoofing	Yes	Both XSS and PHP mail based: https://www.acunetix.com/vulnerabilities/web/microsoft-sharepoint-xss-spoofing-vulnerability/ and https://www.acunetix.com/vulnerabilities/web/php-mail-function-ascii-control-character-header-spoofing-vulnerability/
Reflected XSS	Yes	https://www.acunetix.com/vulnerability-scanner/xss-vulnerability-scanning/
Persistent XSS	Yes	https://www.acunetix.com/vulnerability-scanner/xss-vulnerability-scanning/
DOM-based XSS	Yes	https://www.acunetix.com/vulnerability-scanner/xss-vulnerability-scanning/
Cross-Frame Scripting	Yes	https://www.acunetix.com/vulnerabilities/web/cross-frame-scripting/
HTML Injection	Yes	https://www.acunetix.com/vulnerabilities/web/html-injection/
CSRF	Yes	Various examples, https://www.acunetix.com/vulnerabilities/web/tag/csrf/
Command execution		
Format string attack	Yes	https://www.acunetix.com/vulnerabilities/web/uncontrolled-format-string/
LDAP injection	Yes	https://www.acunetix.com/vulnerabilities/web/ldap-injection/
OS command injection	Yes	Detection for example Struts2, WordPress and Apache environments: https://www.acunetix.com/vulnerabilities/web/struts2-xwork-remote-command-execution/, https://www.acunetix.com/vulnerabilities/web/wordpress-2-1-1-command-execution-backdoor-vulnerability-2-1-1-2-1-1/ and https://www.acunetix.com/vulnerabilities/web/apache-struts2-remote-command-execution-s2-045/
SQL injection	Yes	https://www.acunetix.com/vulnerabilities/web/sql-injection/
Blind SQL injection	Yes	https://www.acunetix.com/vulnerabilities/web/blind-sql-injection/
SSI injection	Yes	https://www.acunetix.com/vulnerabilities/web/server-side-javascript-injection/
XPath injection	Yes	https://www.acunetix.com/vulnerabilities/web/xpath-injection-vulnerability/
HTTP header injection / response splitting	Yes	https://www.acunetix.com/vulnerabilities/web/crlf-injection-http-response-splitting/
Remote file	Yes	https://www.acunetix.com/vulnerabilities/web/tag/file-

includes		inclusion/
Local file includes	Yes	https://www.acunetix.com/vulnerabilities/web/tag/file-inclusion/
Potential malicious file uploads	Yes	https://www.acunetix.com/vulnerabilities/web/file-upload/
Information disclosure		
Directory indexing	Yes	https://www.acunetix.com/vulnerabilities/web/directory-listing/
Information leakage	Yes	https://www.acunetix.com/blog/web-security-zone/how-to-stop-backup-leaking-sensitive-information/
Path traversal	Yes	Detects multiple known path traversal vulnerabilities in various components: https://www.acunetix.com/vulnerabilities/web/cisco-adaptive-security-appliance-asa-path-traversal/, https://www.acunetix.com/vulnerabilities/web/path-traversal-via-misconfigured-nginx-alias/ and https://www.acunetix.com/vulnerabilities/web/tomcat-path-traversal-via-reverse-proxy-mapping/
Predictable resource location	N/A	
Insecure HTTP methods enabled	Yes	Detects OPTIONS, CONNECT and TRACE methods: https://www.acunetix.com/vulnerabilities/web/options-method-is-enabled/, https://www.acunetix.com/vulnerabilities/web/apache-proxy-http-connect-method-enabled/ and https://www.acunetix.com/vulnerabilities/web/trace-method-is-enabled/
Default web server files	Yes	https://www.acunetix.com/vulnerabilities/web/web-server-default-welcome-page/
Testing and diagnostics pages	Yes	https://www.acunetix.com/vulnerabilities/web/asp-net-diagnostic-page/
Internal IP Address disclosure	Yes	Manual confirmation is required: https://www.acunetix.com/vulnerabilities/web/possible-internal-ip-address-disclosure/
Architectural support		
SPA support	Yes	https://www.acunetix.com/vulnerability-scanner/web-application-security/
Custom authentication headers	Yes	https://www.acunetix.com/blog/docs/scan-http-authentication-protected-area/
Custom authentication	Yes	https://www.acunetix.com/blog/docs/scanning-for-vulnerabilities-using-custom-cookies/

cookies		
CAPTCHA support	Yes	https://www.acunetix.com/blog/releases/better-support-for-captcha-and-modern-authentication-mechanisms/
Field value autofill support	Yes	https://www.acunetix.com/blog/docs/acunetix-wvs-input-fields/
Usability		
Executive summary	Yes	https://www.acunetix.com/support/docs/types-reports/
Technical detail report	Yes	https://www.acunetix.com/support/docs/types-reports/
Delta report	Yes	https://www.acunetix.com/support/docs/types-reports/
Compliance reports	Yes	https://www.acunetix.com/support/docs/types-reports/
Report exporting	Yes	https://www.acunetix.com/support/docs/wvs/generating-reports/
Scheduled scanning	Yes	https://www.acunetix.com/blog/docs/how-to-schedule-future-and-recurrent-scans/
Scanning pause and resume	Yes	https://www.acunetix.com/blog/docs/can-i-pause-a-scan/
Real-time scan monitoring	Yes, indirect reference on faq	https://www.acunetix.com/blog/docs/my-scan-seems-to-be-stuck/
Scan logging	Yes	https://www.acunetix.com/blog/docs/enable-logging-scan/
Multiple simultaneous scans support	Yes	https://www.acunetix.com/blog/docs/how-to-scan-large-websites/
Multi-user support	Yes	https://www.acunetix.com/support/docs/wvs/configuring-users/
GUI	Yes	https://www.acunetix.com/support/docs/wvs/installing-acunetix-wvs/
CLI	N/A	
Ticketing / Bug tracking system integration	Yes	https://www.acunetix.com/vulnerability-scanner/acunetix-integrations/
Browser automation integration	Yes	https://www.acunetix.com/blog/docs/what-are-import-files/
CI integration	Yes	https://www.acunetix.com/vulnerability-scanner/acunetix-integrations/
API	Yes	https://www.acunetix.com/support/api-documentation/
Cost	$6995 / year for max 5 websites	https://www.acunetix.com/ordering/

Netsparker Team

Netsparker Team is a cloud-based web application scanner targeted towards medium and large organizations, providing a complete workflow solution for both assessing and managing security vulnerabilities. It provides various integration possibilities to the existing SDLC solutions and enables the teams to fully automate various processes that are otherwise handled manually. [1] Netsparker also provides the unique Proof-Based-Scanning™ technology, which simulates the activities of a penetration tester in order to verify the scan results and sort out possible false positives, resulting in a very low rate of false positives. [2] Netsparker Team includes access to both versions of the scanner, the Standard and the Enterprise version [3]

On a general level, Netsparker provides a good amount of information about their various products and features through their Support pages which contain various information related to using the actual products [4], but also valuable information about the vulnerabilities that the application is able to detect through the vulnerability index. [5] In addition to providing information about their products and it's capabilities, the company also runs various blogs focusing on covering various security vulnerabilities related to web applications, product updates and usage in addition to covering the role of web application security scanners in the field of cyber security from somewhat neutral perspective, which is definitely a nice add to the already extensive amount of information provided on their website. [6]

	Netsparker Team	Source
Overview sources		[1] https://www.netsparker.com/product/team/ [2] https://www.netsparker.com/features/advanced/accurate-proof-based-scanning-technology/ [3] https://www.netsparker.com/support/netsparker-editions/#Netsparker-Standard [4] https://www.netsparker.com/support/ [5] https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/ [6] https://www.netsparker.com/blog/
Authentication		
Brute force	Yes	https://www.netsparker.com/support/configuring-scan-policies-netsparker/

Insufficient authentication	N/A	
Weak password recovery validation	N/A	
Lack of SSL protection on login pages	Yes	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/ssltls-not-implemented/
Auto-complete not disabled on password fields	Yes	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/autocomplete-enabled-password-field/
Authorization		
Credential and session prediction	Yes	Supports basic authentication credentials: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/weak-basic-authentication-credentials/
Insufficient authorization	Yes	Tested using HTTP request builder tool: https://www.netsparker.com/blog/web-security/owasp-top-10/
Insufficient session expiration	N/A	
Session fixation	N/A	
Client-side attacks		
Content spoofing	Yes	In form of Frame injection to include spoofed content on the site: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/frame-injection/
Reflected XSS	Yes	https://www.netsparker.com/website-security-scanner/xss-vulnerability-scanner/
Persistent XSS	Yes	https://www.netsparker.com/website-security-scanner/xss-vulnerability-scanner/
DOM-based XSS	Yes	https://www.netsparker.com/website-security-scanner/xss-vulnerability-scanner/
Cross-Frame Scripting	N/A	
HTML Injection	N/A	
CSRF	Yes	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/cross-site-request-forgery/
Command execution		
Format string attack	N/A	
LDAP injection	N/A	
OS command	Yes	Both Out of band and blind types:

injection		https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/out-of-band-command-injection/ and https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/blind-command-injection/
SQL injection	Yes	https://www.netsparker.com/website-security-scanner/sql-injection-vulnerability-scanner/
Blind SQL injection	Yes	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/blind-sql-injection/
SSI injection	N/A	
XPath injection	N/A	
HTTP header injection / response splitting	Yes	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/http-header-injection/
Remote file includes	Yes	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/cross-site-scripting-via-remote-file-inclusion/
Local file includes	Yes	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/code-execution-via-local-file-inclusion/
Potential malicious file uploads	Yes	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/code-execution-via-file-upload/
Information disclosure		
Directory indexing	Yes	Supported on various server platforms: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/directory-listing-iis/, https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/directory-listing-aspnet-server/, https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/directory-listing-apache/ and more.
Information leakage	Yes	Detection of various information leak disclosure types: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/social-security-number-disclosure/, https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/username-disclosure-microsoft-sql-server/, https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/username-disclosure-mysql/, https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/credit-card-disclosure/ and more.
Path traversal	Yes	Included in the LFI checks: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/code-execution-via-local-file-inclusion/ referenced in https://www.netsparker.com/blog/news/comparison-web-vulnerability-scanners-netsparker-2013-2014/
Predictable	Yes	Forced browsing tool:

resource location		https://www.netsparker.com/support/common-directories/
Insecure HTTP methods enabled	Yes	Detects OPTIONS, TRACE/TRACK and openly redirected POST methods: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/options-method-enabled/, https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/tracetrack-method-detected/ and https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/open-redirection-in-post-method/
Default web server files	Yes	Detects for apache, CakePHP, Tomcat and various IIS versions: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/default-page-detected-apache/, https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/default-page-detected-cakephp-framework/, https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/default-page-detected-tomcat/ and https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/default-page-detected-iis-100/
Testing and diagnostics pages	N/A	
Internal IP Address disclosure	Yes	https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/internal-ip-address-disclosure/
Architectural support		
SPA support	Yes	https://www.netsparker.com/support/scanning-single-page-applications/
Custom authentication headers	Yes	https://www.netsparker.com/support/creating-new-scan-netsparker/
Custom authentication cookies	Yes	https://www.netsparker.com/support/creating-new-scan-netsparker/
CAPTCHA support	Yes	https://www.netsparker.com/support/creating-new-scan-netsparker/ and https://www.netsparker.com/support/configuring-scan-policies-netsparker/
Field value autofill support	Yes	https://www.netsparker.com/blog/docs-and-faqs/configure-predefined-web-form-values-web-vulnerability-scanner/
Usability		
Executive summary	Yes	https://www.netsparker.com/support/reviewing-scan-results-imported-vulnerabilities/ and https://www.netsparker.com/support/report-templates-netsparker/

Technical detail report	Yes	https://www.netsparker.com/support/reviewing-scan-results-imported-vulnerabilities/ and https://www.netsparker.com/support/report-templates-netsparker/
Delta report	Yes	https://www.netsparker.com/support/generating-viewing-reports-netsparker-enterprise/ and https://www.netsparker.com/support/report-templates-netsparker/
Compliance reports	Yes	https://www.netsparker.com/support/report-templates-netsparker/
Report exporting	Yes	https://www.netsparker.com/support/report-templates-netsparker/
Scheduled scanning	Yes	https://www.netsparker.com/support/scheduling-scans/
Scanning pause and resume	Yes	https://www.netsparker.com/blog/releases/september-2019-update-netsparker-enterprise/
Real-time scan monitoring	Yes	https://www.netsparker.com/support/introduction-dashboards-netsparker/
Scan logging	Yes	https://www.netsparker.com/support/introduction-dashboards-netsparker/
Multiple simultaneous scans support	Yes	For individual websites: https://www.netsparker.com/support/website-groups-netsparker-enterprise/
Multi-user support	Yes	https://www.netsparker.com/features/advanced/boost-security-team-collaboration/
GUI	Yes	https://www.netsparker.com/product/enterprise/
CLI	No	Only for standard version: https://www.netsparker.com/support/command-line-interface-netsparker-standard/
Ticketing / Bug tracking system integration	Yes	https://www.netsparker.com/support/category/integrations/
Browser automation integration	Partial	Supports selenium as part of manual crawling process: https://www.netsparker.com/support/manual-crawling-proxy-mode-netsparker/
CI integration	Yes	https://www.netsparker.com/support/category/integrations/
API	Yes	https://www.netsparkercloud.com/docs/index
Cost	On premise, more than 50 websites supported	https://www.netsparker.com/pricing/

Burp Suite Enterprise

Burp suite enterprise is essentially a web application scanner based on the technology empowering the Burp suite penetration testing toolkit loved by penetration testers worldwide. The enterprise version is fitted with simplified user interfaces, automation, scheduling, scaling and integration possibilities to support enterprise environment and make the DevSecOps a reality. [1]

Burp suite family consists of three products: the community version containing the essential manual tools for penetration testing, the professional version providing more advanced manual tools and the web vulnerability scanner, but lacking the scheduling and repeating, as well as scaling and CI integration capabilities, and lastly the Enterprise version, which unfortunately does not provide any manual tools for the testers. [2] This is a very interesting diversion between the available versions as many of the assessment criteria could be easily tested manually using the advanced tools and provided instructions, but the actual web vulnerability scanner of the enterprise edition is unable to detect or handle. The enterprise version also does not support community developed extensions available through the BApp store, [3] which provide support for some quite essential features outside of the functionalities provided by the burp suite. [4]

When assessing the Burp Suite Enterprise, the tests cases that could be manually tested by using the advanced penetration testing tools available through the burp professional edition or an extension available in the BApp store are also documented in the assessment criteria for later review in the discussions section of the actual thesis.

	Burp suite Enterprise	Source
Overview sources		[1] https://portswigger.net/burp/enterprise [2] https://portswigger.net/burp [3] https://forum.portswigger.net/thread/burp-enterprise-and-extensions-support-d9f0e6fb [4] https://portswigger.net/bappstore
Authentication		
Brute force	N/A	Manual testing available through the tools provided in professional version: https://portswigger.net/support/using-burp-to-brute-force-a-login-page
Insufficient authentication	N/A	Manual testing available through the tools provided in professional version: https://portswigger.net/support/using-burp-to-attack-authentication and https://portswigger.net/support/using-sql-injection-to-bypass-authentication in particular
Weak	N/A	Manual testing available through the tools provided in

password recovery validation		professional version: https://blog.appsecco.com/mass-account-pwning-or-how-we-hacked-multiple-user-accounts-using-weak-reset-tokens-for-passwords-c2d6c0831377
Lack of SSL protection on login pages	N/A	Is available through an extension, but extensions are not yet supported on the enterprise version: https://portswigger.net/bappstore/474b3c575a1a4584aa44dfefc70f269d
Auto-complete not disabled on password fields	Yes	https://portswigger.net/kb/issues/00500800_password-field-with-autocomplete-enabled
Authorization		
Credential and session prediction	N/A	
Insufficient authorization	N/A	Available through an extension, but extensions are not yet supported on the enterprise version: https://portswigger.net/support/using-burp-to-test-for-missing-function-level-access-control
Insufficient session expiration	N/A	Available through an extension, but extensions are not yet supported on the enterprise version: https://portswigger.net/bappstore/c4bfd29882974712a1d69c6d8f05874e
Session fixation	N/A	Manual testing available through the tools provided in professional version: https://portswigger.net/support/using-burp-to-hack-cookies-and-manipulate-sessions
Client-side attacks		
Content spoofing	N/A	
Reflected XSS	Yes	https://portswigger.net/web-security/cross-site-scripting/reflected
Persistent XSS	Yes	https://portswigger.net/web-security/cross-site-scripting/stored
DOM-based XSS	Yes	https://portswigger.net/web-security/cross-site-scripting/dom-based
Cross-Frame Scripting	N/A	
HTML Injection	N/A	Manual testing available through the tools provided in professional version: https://subscription.packtpub.com/book/networking_and_servers/9781789531732/9/ch09lvl1sec74/testing-for-html-injection
CSRF	Yes	https://portswigger.net/web-security/csrf
Command execution		

Format string attack	N/A	
LDAP injection	Yes	https://portswigger.net/kb/issues/00100500_ldap-injection
OS command injection	Yes	https://portswigger.net/web-security/os-command-injection
SQL injection	Yes	https://portswigger.net/web-security/sql-injection
Blind SQL injection	Yes	https://portswigger.net/support/using-burp-to-detect-blind-sql-injection-bugs
SSI injection	Yes	https://portswigger.net/kb/issues/00101100_ssi-injection
XPath injection	Yes	https://portswigger.net/kb/issues/00100600_xpath-injection
HTTP header injection / response splitting	Yes	https://portswigger.net/kb/issues/00200200_http-response-header-injection
Remote file includes	Yes	https://portswigger.net/kb/issues/00100b00_file-path-manipulation
Local file includes	Yes	https://portswigger.net/kb/issues/00100b00_file-path-manipulation
Potential malicious file uploads	Yes	https://portswigger.net/kb/issues/00500980_file-upload-functionality
Information disclosure		
Directory indexing	Yes	https://portswigger.net/kb/issues/00600100_directory-listing
Information leakage	Yes, support for various cases	https://portswigger.net/kb/issues/00600500_credit-card-numbers-disclosed, https://portswigger.net/kb/issues/00600550_private-key-disclosed, https://portswigger.net/kb/issues/00600400_social-security-numbers-disclosed, https://portswigger.net/kb/issues/00600300_private-ip-addresses-disclosed, https://portswigger.net/kb/issues/00600200_email-addresses-disclosed and more.
Path traversal	Yes	https://portswigger.net/web-security/file-path-traversal
Predictable resource location	N/A	Manual testing available through the tools provided in professional version with FuzzDB attack pattern dictionary: https://github.com/fuzzdb-project/fuzzdb
Insecure HTTP methods enabled	Yes	https://portswigger.net/kb/issues/00500a00_http-trace-method-is-enabled
Default web server files	N/A	

Testing and diagnostics pages	N/A	
Internal IP Address disclosure	Yes	https://portswigger.net/kb/issues/00600300_private-ip-addresses-disclosed
Architectural support		
SPA support	Partial	Improvements to the SPA scanning functionalities are included on the 2020 roadmap: https://portswigger.net/burp/documentation/scanner/crawling, https://forum.portswigger.net/thread/scan-a-single-page-application-with-enterprise-scanner-c8086510 and https://portswigger.net/blog/burp-suite-roadmap-for-2020
Custom authentication headers	N/A	Available through an extension, but extensions are not yet supported on the enterprise version: https://portswigger.net/bappstore/807907f5380c4cb38748ef4fc1d8cdbc
Custom authentication cookies	Yes	https://portswigger.net/support/manually-setting-a-cookie-for-burp-suites-crawl-and-audit
CAPTCHA support	N/A	Available through an extension, but extensions are not yet supported on the enterprise version: https://forum.portswigger.net/thread/bypass-racaptcha-on-website-login-7d4d792c and https://github.com/TimGuenther/burp-reCAPTCHA
Field value autofill support	Yes	https://portswigger.net/blog/mobp-custom-form-filling-rules and
Usability		
Executive summary	Yes	https://portswigger.net/burp/documentation/desktop/scanning/reporting-results#report-details and https://portswigger.net/burp/samplereport/burpscannersamplereport
Technical detail report	Yes	https://portswigger.net/burp/documentation/desktop/scanning/reporting-results#report-details and https://portswigger.net/burp/samplereport/burpscannersamplereport
Delta report	Yes	https://portswigger.net/burp/releases/enterprise-edition-1-0-10beta
Compliance reports	No	https://forum.portswigger.net/thread/owasp-top-10-reporting-fab55562f3bfa
Report exporting	Yes	Both XML and HTML: https://portswigger.net/burp/documentation/desktop/scanning/reporting-results#report-details
Scheduled scanning	Yes	https://portswigger.net/burp/documentation/enterprise/reference/scans
Scanning	Yes	https://portswigger.net/burp/documentation/desktop/dashb

pause and resume		oard/task-execution-settings
Real-time scan monitoring	Yes	https://portswigger.net/burp/documentation/enterprise/reference/scans
Scan logging	N/A	Available through an extension, but extensions are not yet supported on the enterprise version: https://portswigger.net/bappstore/1edf849a4df447158c04141e9a4e67db
Multiple simultaneous scans support	Yes	https://portswigger.net/burp/documentation/desktop/scanning and https://portswigger.net/blog/enterprise-edition-performing-scans
Multi-user support	Yes	https://portswigger.net/blog/enterprise-edition-configuring-your-team
GUI	Yes	Referenced in various documentations throughout the site such as https://portswigger.net/burp/documentation/enterprise/reference/settings/updates and https://portswigger.net/burp/documentation/enterprise/getting-started/system-requirements
CLI	N/A	Available through an extension, but extensions are not yet supported on the enterprise version: https://portswigger.net/bappstore/d54b11f7af3c4dfeb6b81fb5db72e381
Ticketing / Bug tracking system integration	Yes	https://portswigger.net/burp/documentation/enterprise/reference/settings/jira-integration
Browser automation integration	Yes	https://portswigger.net/support/using-burp-with-selenium
CI integration	Yes	Through native plugins : https://portswigger.net/burp/extender/ci-integration and https://portswigger.net/burp/documentation/enterprise/reference/rest-api
API	Yes	https://portswigger.net/burp/documentation/enterprise/reference/rest-api
Cost	$3999 / year for 1 site + $399 for each additional website	https://portswigger.net/pricing

Arachni

Arachni is a web application security scanner based on the open ruby framework, open-source development and public source code access for anyone. The scanner is multiplatform and supports all major operating systems, windows Mac OS X and Linux. The scanner is equipped with support for a multitude of vulnerabilities, supports multiple users and even provides a REST API for supporting custom integrations. [1] The scanner also provides WIVET scores that are top tier of the industry [2] and supports various modern web applications through an integrated browser engine.[1]

Arachni provides a simple overview of the framework's features through their website, including most of the supported vulnerabilities, scanning features and setting up information. [3] Some of the features were also accurately documented within the source code comments and the official GitHub page for the application acted as a secondary source of information for conducting this study. [4]

	Arachni	Source
Overview sources		[1] https://www.arachni-scanner.com/ [2] http://sectoolmarket.com/wivet-score-unified-list.html [3] https://www.arachni-scanner.com/features/framework/ [4] https://github.com/Arachni/arachni
Authentication		
Brute force	Yes	Through dictionary attacker plugin for both HTTP and session authentication: https://www.arachni-scanner.com/features/framework/
Insufficient authentication	N/A	
Weak password recovery validation	N/A	
Lack of SSL protection on login pages	Yes	https://github.com/Arachni/arachni/blob/0428b9db1b8b15c2796692e646da21e27668676e/components/checks/passive/grep/unencrypted_password_forms.rb
Auto-complete not disabled on password fields	Yes	https://github.com/Arachni/arachni/blob/master/components/checks/passive/grep/password_autocomplete.rb
Authorization		
Credential and session prediction	N/A	
Insufficient authorization	N/A	
Insufficient session	N/A	

expiration		
Session fixation	Yes	https://github.com/Arachni/arachni/blob/master/components/checks/active/session_fixation.rb
Client-side attacks		
Content spoofing	N/A	
Reflected XSS	Yes	https://www.arachni-scanner.com/features/framework/, https://github.com/Arachni/arachni/blob/master/components/checks/active/xss_event.rb
Persistent XSS	Yes	https://www.arachni-scanner.com/features/framework/, https://github.com/Arachni/arachni/blob/master/components/checks/active/xss_event.rb
DOM-based XSS	Yes	https://www.arachni-scanner.com/features/framework/
Cross-Frame Scripting	N/A	
HTML Injection	N/A	
CSRF	Yes	https://www.arachni-scanner.com/features/framework/
Command execution		
Format string attack	N/A	
LDAP injection	Yes	https://www.arachni-scanner.com/features/framework/
OS command injection	Yes	https://www.arachni-scanner.com/features/framework/
SQL injection	Yes	https://www.arachni-scanner.com/features/framework/
Blind SQL injection	Yes	https://www.arachni-scanner.com/features/framework/
SSI injection		
XPath injection	Yes	https://www.arachni-scanner.com/features/framework/
HTTP header injection / response splitting	Yes	https://www.arachni-scanner.com/features/framework/
Remote file includes	Yes	https://www.arachni-scanner.com/features/framework/
Local file includes	Yes	https://www.arachni-scanner.com/features/framework/
Potential malicious file uploads	N/A	
Information disclosure	Yes	https://www.arachni-scanner.com/features/framework/

Directory indexing	Yes	https://www.arachni-scanner.com/features/framework/
Information leakage	Yes	Backup directories and files: https://github.com/Arachni/arachni/blob/0428b9db1b8b15c2796692e646da21e27668676e/components/checks/passive/backup_files.rb and https://github.com/Arachni/arachni/blob/0428b9db1b8b15c2796692e646da21e27668676e/components/checks/passive/backup_directories.rb
Path traversal	Yes	https://www.arachni-scanner.com/features/framework/
Predictable resource location	Yes	https://github.com/Arachni/arachni/blob/0428b9db1b8b15c2796692e646da21e27668676e/components/checks/passive/common_admin_interfaces.rb
Insecure HTTP methods enabled	Yes	https://www.arachni-scanner.com/features/framework/
Default web server files	N/A	
Testing and diagnostics pages	N/A	
Internal IP Address disclosure	Yes	https://www.arachni-scanner.com/features/framework/
Architectural support		
SPA support	Yes	https://www.arachni-scanner.com/ and https://www.arachni-scanner.com/features/framework/crawl-coverage-vulnerability-detection/#vulnerability-detection
Custom authentication headers	Yes	https://www.arachni-scanner.com/features/framework/
Custom authentication cookies	Yes	https://www.arachni-scanner.com/features/framework/
CAPTCHA support	Yes	Requires the use of proxy plugin: https://github.com/Arachni/arachni/issues/851
Field value autofill support	Yes	https://www.arachni-scanner.com/features/framework/
Usability		
Executive summary	Yes	https://rubydoc.info/github/Arachni/arachni#Reporters HTML version
Technical detail report	Yes	https://rubydoc.info/github/Arachni/arachni#Reporters XML / text versions contain lots of technical details
Delta report	N/A	

Compliance reports	Yes	https://rubydoc.info/github/Arachni/arachni#Reporters HTML version
Report exporting	Yes	https://rubydoc.info/github/Arachni/arachni#Reporters and https://www.arachni-scanner.com/features/web-user-interface/
Scheduled scanning	Yes	https://www.arachni-scanner.com/features/web-user-interface/
Scanning pause and resume	Yes	https://www.arachni-scanner.com/features/framework/
Real-time scan monitoring	Yes	https://www.arachni-scanner.com/features/framework/distributed-architecture/
Scan logging	Yes	http://support.arachni-scanner.com/discussions/questions/4991-arachni-log-files and https://github.com/Arachni/arachni/wiki/Command-line-user-interface#output-debug
Multiple simultaneous scans support	Yes	https://www.arachni-scanner.com/features/framework/distributed-architecture/
Multi-user support	Yes	https://www.arachni-scanner.com/features/web-user-interface/
GUI	Yes	https://www.arachni-scanner.com/features/web-user-interface/, https://www.arachni-scanner.com/features/framework/
CLI	Yes	https://www.arachni-scanner.com/screenshots/command-line-interface/, https://www.arachni-scanner.com/features/framework/
Ticketing / Bug tracking system integration	N/A	
Browser automation integration	N/A	
CI integration	Yes	Jenkins provides an integration plugin: https://plugins.jenkins.io/arachni-scanner/
API	Yes	REST and RPC support: https://github.com/Arachni/arachni/wiki/REST-API and https://github.com/Arachni/arachni/wiki/RPC-API
Cost	Free	

Zed Attack Proxy

OWASP Zed Attack Proxy, more commonly known as ZAP is entitled as the “world’s most popular free, open-source web security tool.” [1] There is a vast community behind the project, consisting from various developers from around the world. [2] The application provides tools for both manual and automatic penetration testing, focusing to derive the penetration testing process into three simple steps: exploring the target web application by crawler while running passive scans against the site, attacking the site using active testing tools and lastly reporting the results back to the user. [3] ZAP also provides various possibilities for external integrations through a dedicated API and existing plugin solutions for various other software [4], enabling various types of SDLC pipeline integrations. The software is also highly modular, as multiple features are available to be integrated to the core product through add-ons, making it easy to customize the feature set based on the requirements of the target application. [5]

There is generally a good amount of information available about the ZAP and it's usage through the documentation available on the home website [6], but also through a vast community of users maintaining active discussion forums [7] and lastly the issue tracking and management systems presented on the related GitHub pages. [8]

	ZAP	Source
Overview sources		[1] https://www.zaproxy.org/ [2] https://github.com/zaproxy/zaproxy/pulse [3] https://www.zaproxy.org/getting-started/ [4] https://www.zaproxy.org/docs/api/#introduction [5] https://www.zaproxy.org/docs/desktop/addons/ [6] https://www.zaproxy.org/docs/ [7] https://groups.google.com/forum/#!forum/zaproxy-users [8] https://github.com/zaproxy
Authentication		
Brute force	Yes	https://www.zaproxy.org/docs/desktop/addons/fuzzer/
Insufficient authentication	Yes	Username enumeration plugin (beta): https://www.zaproxy.org/docs/desktop/addons/active-scan-rules-beta/
Weak password recovery validation	N/A	
Lack of SSL protection on login pages	Yes	https://www.zaproxy.org/docs/desktop/addons/passive-scan-rules/
Auto-complete not disabled on password fields	N/A	
Authorization		
Credential and	Yes	https://www.zaproxy.org/docs/desktop/addons/token-

session prediction		generator/
Insufficient authorization	Yes	https://www.zaproxy.org/docs/desktop/addons/access-control-testing/
Insufficient session expiration	N/A	
Session fixation	Yes	Beta plugin available: https://www.zaproxy.org/docs/desktop/addons/active-scan-rules-beta/
Client-side attacks		
Content spoofing	N/A	
Reflected XSS	Yes	https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/
Persistent XSS	Yes	https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/
DOM-based XSS	Yes	https://www.zaproxy.org/docs/desktop/addons/dom-xss-active-scan-rule/
Cross-Frame Scripting	N/A	
HTML Injection	N/A	
CSRF	Yes	https://www.zaproxy.org/docs/desktop/addons/passive-scan-rules/
Command execution		
Format string attack	Yes	https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/
LDAP injection	Yes	Alpha plugin available: https://www.zaproxy.org/docs/desktop/addons/active-scan-rules-alpha/
OS command injection	Yes	https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/
SQL injection	Yes	https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/ and https://www.zaproxy.org/docs/desktop/addons/active-scan-rules-beta/
Blind SQL injection	Yes	https://www.zaproxy.org/docs/desktop/addons/advanced-sqlinjection-scanner/ and https://github.com/zaproxy/zap-extensions/blob/master/addOns/sqliplugin/src/main/java/org/zaproxy/zap/extension/sqliplugin/SQLInjectionPlugin.java
SSI injection	Yes	https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/
XPath	Yes	Beta plugin available:

injection		https://www.zaproxy.org/docs/desktop/addons/active-scan-rules-beta/
HTTP header injection / response splitting	Yes	CRLF injection: https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/
Remote file includes	Yes	https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/
Local file includes	N/A	
Potential malicious file uploads	N/A	
Information disclosure		
Directory indexing	Yes	https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/
Information leakage	Yes	Source code, backup file and various other information disclosure checks: https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/, https://www.zaproxy.org/docs/desktop/addons/active-scan-rules-beta/ and https://www.zaproxy.org/docs/desktop/addons/passive-scan-rules/
Path traversal	Yes	https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/
Predictable resource location	Yes	Forced browsing plugin: https://www.zaproxy.org/docs/desktop/addons/forced-browse/
Insecure HTTP methods enabled	Yes	Beta plugin available: https://www.zaproxy.org/docs/desktop/addons/active-scan-rules-beta/
Default web server files	N/A	
Testing and diagnostics pages	N/A	
Internal IP Address disclosure	Yes	https://www.zaproxy.org/docs/desktop/addons/passive-scan-rules/
Architectural support		
SPA support	Yes	https://www.zaproxy.org/docs/desktop/start/features/structparams/ and https://www.zaproxy.org/docs/desktop/addons/ajax-spider/ for native support and https://blog.xaviermaso.com/2018/10/01/Scanning-modern-

		web-applications-with-OWASP-ZAP.html#zap-and-modern-web-applications for improved support
Custom authentication headers	Yes	https://www.zaproxy.org/docs/desktop/start/features/session management/, https://www.zaproxy.org/docs/desktop/start/features/httpsessions/ and https://www.zaproxy.org/docs/desktop/start/features/authentication/
Custom authentication cookies	Yes	https://www.zaproxy.org/docs/desktop/start/features/session management/ and https://www.zaproxy.org/docs/desktop/start/features/httpsessions/
CAPTCHA support	N/A	
Field value autofill support	Yes	https://www.zaproxy.org/docs/desktop/addons/form-handler/
Usability		
Executive summary	Yes	https://www.zaproxy.org/docs/desktop/ui/tlmenu/report/, https://www.zaproxy.org/docs/desktop/addons/custom-report/, also supports exporting data to BIRT for visualization https://www.eclipse.org/birt/
Technical detail report	Yes	https://www.zaproxy.org/docs/desktop/ui/tlmenu/report/, https://www.zaproxy.org/docs/desktop/addons/custom-report/,
Delta report	N/A	
Compliance reports	N/A	
Report exporting	Yes	https://www.zaproxy.org/docs/desktop/addons/export-report/
Scheduled scanning	Partial	This can be achieved through Jenkins integration, API or CLI Quick start add-on: https://groups.google.com/forum/#!topic/zaproxy-develop/Vn63NRIsN6E, https://plugins.jenkins.io/zap/, https://www.zaproxy.org/docs/api/ and https://github.com/zaproxy/zaproxy/issues/5226 https://www.zaproxy.org/docs/desktop/addons/quick-start/
Scanning pause and resume	Partial.	Scanner and passive scans can be paused and resumed, but Scanner and passive scans can be paused and resumed, but active scans can only be paused or stopped by setting breakpoints, and cannot be resumed: https://www.zaproxy.org/docs/desktop/ui/tabs/spider/, https://www.zaproxy.org/docs/api/#using-spider, https://www.zaproxy.org/docs/desktop/ui/tabs/breakpoints/ and https://www.zaproxy.org/docs/api/#using-active-scan
Real-time scan monitoring	Yes	https://www.zaproxy.org/docs/desktop/ui/dialogs/scanprogress/
Scan logging	Yes	https://www.zaproxy.org/faq/how-do-you-configure-zap-

		logging/
Multiple simultaneous scans support	N/A	
Multi-user support	N/A	
GUI	Yes	https://www.zaproxy.org/getting-started/ and https://www.zaproxy.org/docs/desktop/ui/
CLI	Yes	https://www.zaproxy.org/docs/desktop/cmdline/
Ticketing / Bug tracking system integration	Yes	https://www.zaproxy.org/docs/desktop/addons/bug-tracker/
Browser automation integration	Yes	https://www.zaproxy.org/docs/desktop/addons/selenium/
CI integration	Yes	https://plugins.jenkins.io/zap/ and possible through the docker https://github.com/zaproxy/zaproxy/wiki/Docker
API	Yes	https://www.zaproxy.org/docs/api/
Cost	Free	