tenable.io®
web app scanning

GETTING STARTED WITH WEB APPLICATION SCANNING

THIS STEP-BY-STEP GUIDE OUTLINES THE KEY CONSIDERATIONS AND BEST PRACTICES THAT CAN POSITION YOUR WAS PROGRAM FOR SECURITY SUCCESS

Table of Contents

1. Introduction

Scanning for vulnerabilities in web applications is significantly different from scanning for traditional vulnerabilities with Nessus, Nessus Agents or Nessus Network Monitor. As a result, using Tenable.io® Web Application Scanning (WAS) requires a different approach to vulnerability assessment and management.

WAS offers significant improvements over the legacy Nessus-based web app scanning policy. This legacy scanning template for Nessus is incompatible with modern web application frameworks such as Javascript, HTML 5, AJAX or single page applications (SPA), among others, potentially leaving you with an incomplete understanding of your web application security posture.

WAS provides comprehensive vulnerability scanning for modern web applications. Its accurate vulnerability coverage minimises false positives and false negatives, ensuring that security teams understand the true security risks in their web applications. It offers safe external scanning that ensures production web applications are not disrupted or delayed.

If you are using WAS for the first time, you can go to our welcome page to get started.

This document explores the differences between WAS and Nessus scanning, various approaches to launching web application scans and prioritizing vulnerabilities, and a detailed view of the architecture and deployment methodology underlying WAS.

2. Important Differences Between Nessus and WAS

Nessus looks for known vulnerabilities. WAS uses Dynamic Application Security Testing (DAST) to find unknown vulnerabilities.

Nessus vulnerability scanning typically identifies Common Vulnerabilities and Exposures (CVEs), Bugtraq ID's and other pre-disclosed vulnerabilities. WAS uses both OWASP Top 10 and other third-party web components to dynamically test how web applications behave as the code is running and typically finds previously undiscovered vulnerabilities by using similar techniques that attackers may use against your website. WAS can also detect some CVE-based vulnerabilities related to web components, such as servers, frameworks or JavaScript libraries.

Nessus can be push-button. DAST can never be fully push-button.

Dynamic testing means evaluating the code as it is executing, potentially with varying inputs. As a result, effective web app security cannot be fully push-button; you must tune for the website or code you're testing. In general, vulnerability management teams usually have limited understanding of the code that makes up each web application. While this level of detail is not required, a few general principles can provide improved results and faster scans, or simply teach the scanner how to authenticate to the website. For example, if a vulnerability administrator is able to read some core web code syntax—such as input fields, field names and general page makeup—they'll have a much easier time conversing with developers and to better discover vulnerabilities and tune scans.

Hosts scanned by Nessus are typically fair game for scrutiny. Websites scanned by WAS may encounter resistance from developers.

Hosts scanned with Nessus tend to be "cattle," while web applications tend to be "pets." The vulnerabilities that Nessus finds don't usually have champions, but with web applications, developers have been known to comb through a 100-page DAST report for the one and only false negative, cry foul and push to throw out the whole report as invalid or wrong. Make sure you bring in the product owners and other sponsors to support the developers, and help them understand the importance of a WAS program in the overall business security requirements. Help the developers prioritize which web-related vulnerabilities are critical to address first.

Authentication is typically more complex in WAS compared to traditional Nessus targets.

Nessus authentication is predominantly OS- or protocol-based (e.g., SMB, SSH), whereas WAS authentication is custom to the developer that made the site. Additionally, with Nessus, authentication occurs once when first scanning the host; with WAS, authentication can occur at any point—or repeatedly—during the scan, plus you have the complication of CAPTCHA tests and similar anti-bot measures and protections. Furthermore, Nessus allows you to scan many hosts with the same authentication based on the underlying OS, but that is not scalable in WAS, as different input validation & forms are likely to be required for access privilege within each application.

Rather than identifying vulnerabilities associated with an OS, WAS identifies a variety of web application flaws as the program runs in a live environment.

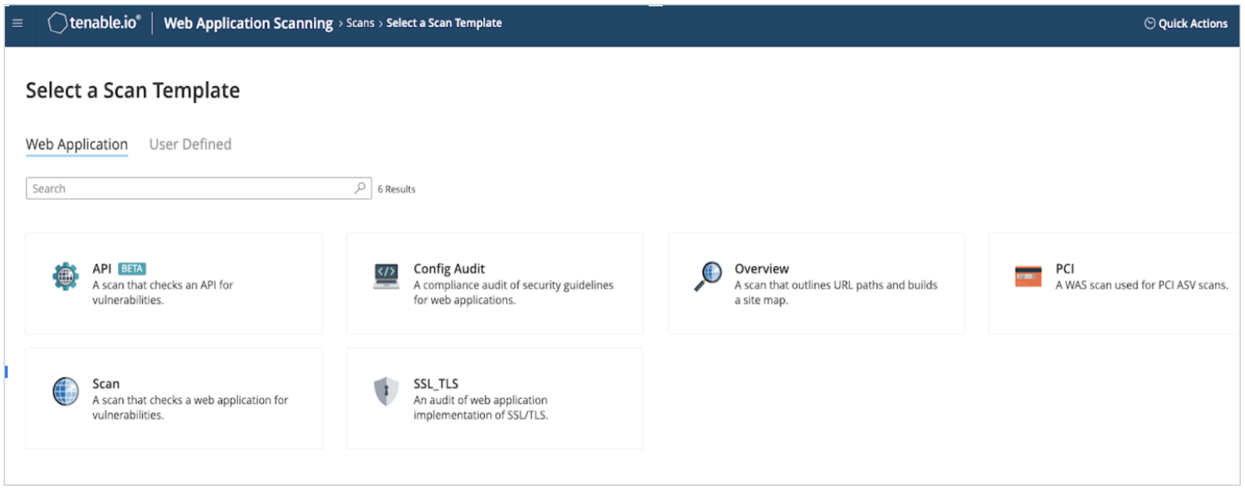
WAS, for example, may find an available SQL injection flaw, which can occur as much due to database misconfiguration as much as a lack of limiting input fields. Fixing a Nessus vulnerability typically requires a patch; fixing a web application flaw could include any combination of patches, configuration changes and code updates.

While you can readily patch vulnerabilities found by Nessus, resolving web application flaws may take weeks or even months.

It is not unusual for a complex web app vulnerability, such as an SQL injection, to take organizations up to 130 days before it's fully addressed. While this reality is less than desirable, there are multiple industry reports validating this standard timeline. Therefore, mitigation may be the first step, in addition to working with the development team to remediate the underlying vulnerability.

3. Types of Web Application Scanning Programs

There are several viable ways to operate a web application scanning program based on DAST technology. Most programs use some combination of each approach to meet different needs for each site.



Select a Scan Template

API — A scan that checks an API for vulnerabilities.

Config Audit — A compliance audit of security guidelines for web applications.

Overview — A scan that outlines URL paths and builds a site map.

PCI — A WAS scan used for PCI ASV scans.

Scan — A scan that checks a web application for vulnerabilities.

SSL_TLS — An audit of web application implementation of SSL/TLS.

Summary of WAS Scan Templates

- **Scan:** The complete set of available checks; all other pre-built templates are a subset of this template, other than the API scan.
- **Overview:** A simplified version of the "Scan" template with several of the active tests removed to lower its impact and speed up the scan.
- **PCI:** A special template used as part of the attestation offering that Tenable provides for the Payment Card Industry (PCI) security standards. Note: Only submissions to attestation consume PCI licenses, otherwise this operates as a simplified version of the "Scan" template.
- **SSL/TLS:** A health check scan focused on the current state of the web server encryption settings and certificate state (e.g., remaining time on the certificate).
- **Config Audit:** A compliance audit that provides detection of externally viewable web server settings that external audit providers commonly review to evaluate the health of a security program.
- **API Scan:** A special template requiring additional configuration to describe the application programming interface (API), so that the scanner can successfully detect relevant vulnerabilities. This includes some of the same tests as the "Scan" template but adds others unique to API endpoints.

Quick Surface-level Checks

These scans typically use the "SSL_TLS" or "Config Audit" templates to provide a rapid test, often lasting only minutes, to give vulnerability management teams an overview of any certificate- and encryption-type issues with a given site, or commonly exposed configuration parameters that are not set to best practice. It is somewhat common for teams to run these templates with greater regularity than in-depth scans in order to validate these surface-level items.

In-depth Assessments

- **Untuned Detailed Scans**

  This approach uses the "Scan" template which has a high probability of finding most vulnerabilities, and also simulates drive-by style attacks that sites commonly experience. This is done with minimal effort on vulnerability management teams. Since there's no tuning or scan refinement required, these scans deploy quickly and obtain valuable incremental visibility back from the scan target (though basic validation will help ensure there are no obvious scan errors). The drawback of this approach is that it may hit its timeout (e.g., 8-hour default in Tenable.io) or miss more complex sections of a site that require authentication or other intervention to ensure they are scanned correctly. These blind spots are common for sites with forums, blogs, large numbers of products, multiple languages and several other scenarios that are complex or involve a site that operates a significant number of pages.

- **Authenticated Detailed Scans**

  Similar to the above scan, this approach incorporates one of the quicker forms of tuning: authentication. You can perform this within the Tenable.io scan configuration page or via the Chrome extension provided by Tenable. In addition to the benefits of an untuned scan, authenticated scans log on as a user and test what (if any) issues are available to a typical user of the site. Note: We recommended that you never log on as an admin user, especially in production (for more information, see the "Key Considerations" section). Authentication requires slightly more effort to create and maintain the test user account, as well as update any unique configurations, since each site is built by a different developer or team and may implement or update authentication in subtly different ways.

- **Tuned Detailed Scans**

  In addition to authentication, vulnerability managers can use other methods to optimise their scans, such as speed or complexity (as detailed further in "Key Considerations"). These refinements involve a small time investment to get things up and running, and may require semi-regular adjustments depending on how frequently developers update the site.

Pre-production Scanning

Some sites are not fully scannable due to scanner impact on a production site, or excessive caution to maintain 100 percent uptime. In these cases, you may need to consider integrating scans from the build tools to trigger a scan via the Tenable.io API based on a weekly or monthly build, or a pre-production location on a regular schedule. It's typically desirable to validate production as much as possible, as this is the most exposed site and it may differ from internal builds for many reasons. This scanning approach occurs with most mature organizations to varying degrees and is often based on site criticality and resource availability.

API Scanning

In addition to web applications, organizations are increasingly adopting APIs to power web applications, B2B transactions, mobile applications and automation scenarios. You can assess these potential exposures by making use of the API scanning template within WAS, providing critical visibility into additional cyber risks. Generally speaking, APIs are scanned more frequently in mature programs or organizations where high risk and exposure is driving requirements. Ultimately, as the security program develops, many organizations try to proactively identify all vulnerable locations to ensure full coverage. This type of scan may require additional input from development staff and rely on a Swagger file to provide the endpoint definitions, so the scanner knows how to communicate to the API itself.

4. Deciding Which WAS Program to Use

Most programs start with a few scans based on the "SSL_TLS" or "Config Audit" templates to familiarise vulnerability managers with how to establish scans and review results. Then, they progress into running an untuned scan using the WAS scan template.

Timeouts are common when you first build out your program. The default scan completion timeout in Tenable.io is eight hours, and extending this may not "complete" the scan; this may only be achievable via tuning for greater speed.

It's viable to run a program based on untuned scans while accepting the timeout. As many web application vulnerabilities span multiple pages containing the same vulnerability, there is a high likelihood that a significant proportion of vulnerabilities are automatically detected within the first several hours of a scan. Our own monitoring suggests this to be the case. Tuned scans typically improve scan efficiency and accuracy by only a small additional percentage of findings, and this comes at the cost of time investment into refining the scan configuration.

Most mature organizations at least tune the scans on their most critical sites. This typically involves 10-20 minutes of effort per site but gets quicker with operator experience. An organization's level of knowledge and resource availability typically determines the percentage of sites that undergo detailed tuning. It's rare to see all sites tuned, especially in organizations with a large number of websites. This is due in part to the dynamic nature of websites; they are often expanded or significantly changed every few years, and some of these changes may require additional review of scan settings to meet with the pace of development of the site being tested.

In summary:

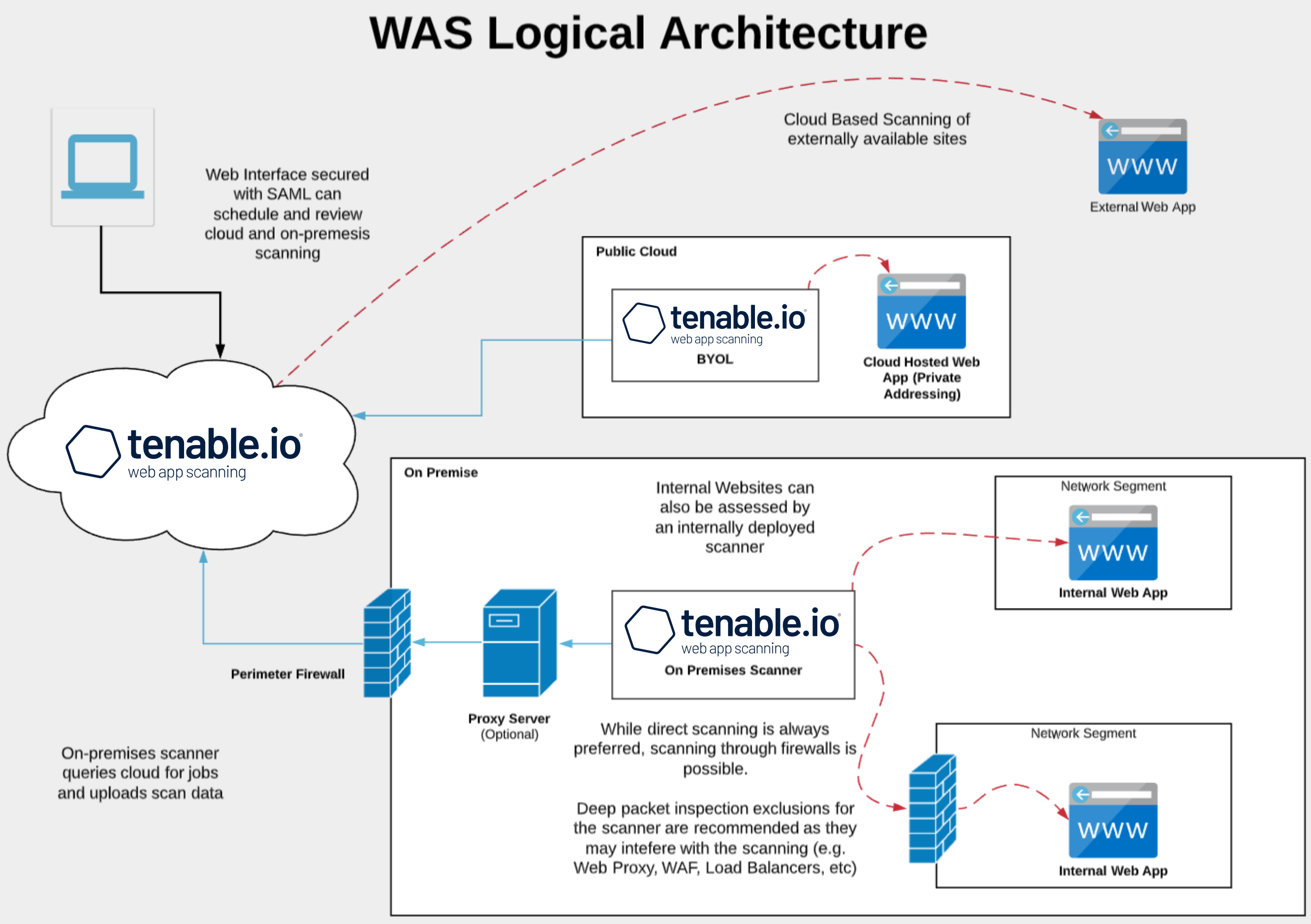
- **Focus on the process first.** Start with the WAS "Scan" (a complete set of checks) or an " Overview" scan (fewer checks but lower impact) templates. Familiarise yourself with the scanner output and work with your teams to incorporate the findings into your workflows. Develop your mitigation and resolution programs.
- **Dig deeper into critical areas.** Once you've established some of the baseline procedures and identified the right owners within your organization for the output from the scanner, start investing time in more advanced tuned scans to gain better visibility into your most important sites.
- **Take action.** There will be a significant amount of data found during any scan that should drive organizational action. Consider all of the potential consumers of the data. Developers will want details so they can easily identify what needs fixing and show improvement over time. Management will likely be concerned with which sites contribute the greatest risk to the business, and therefore where to allocate resources. Security leadership will also want general category information such as the OWASP vulnerability categories for all sites to determine if an education program for the business is warranted on a specific classification of vulnerabilities.

Tenable Professional Services offers a highly recommended quick-start program for those new to WAS scanning to help establish the mechanics of developing a new program. Additionally, the ProServe team runs a Cyber Exposure workshop for establishing the internal processes and initial goals of developing a broader vulnerability management program. These services offer the best possible way for organizations to get a solid foundation and understanding of effective cybersecurity programs and familiarize themselves with their latest investment. Contact your Tenable sales rep if you are interested. sales@tenable.com.

5. Key Considerations to Optimize Your Scan Results

1. Identify where the web application is located.

    - Public Websites
      External websites can be scanned from Tenable.io using the internet-based Web Application Scanner or an on-premises scanner.
    - Private Websites
      Internal or intranet web applications can be scanned from Tenable.io using an on-premises Web Application Scanner.



WAS Logical Architecture
Web Interface secured with SAML can schedule and review cloud and on-premesis scanning
Cloud Based Scanning of externally available sites
External Web App
Public Cloud
tenable.io web app scanning
BYOL
Cloud Hosted Web App (Private Addressing)
tenable.io web app scanning
On Premise
Internal Websites can also be assessed by an internally deployed scanner
Network Segment
Internal Web App
Perimeter Firewall
Proxy Server (Optional)
tenable.io web app scanning
On Premises Scanner
While direct scanning is always preferred, scanning through firewalls is possible.
Deep packet inspection exclusions for the scanner are recommended as they may intefere with the scanning (e.g. Web Proxy, WAF, Load Balancers, etc)
Network Segment
Internal Web App
On-premises scanner queries cloud for jobs and uploads scan data

2. **Ensure the scanner has a route, network-wise, to the target.**
   If the scanner cannot reach the web application, or cannot deliver an input and retrieve results, then scanning will fail. Network constraints can affect scanning (e.g., latency) or network controls (e.g., host-based firewalls, network firewalls, network segregation, etc.). Always include internal web application scanners on your allowlist.

3. **Scanner location can impact latency or server response times.**
   During a scan, if there are too many timeouts, the session will terminate. You'll want to choose a scanner that is located as close as possible to the targets. Review the sitemap plugin attachments and review to see if there are long page load times or timeouts. This can be caused by too many concurrent tests on a slower server, a scanner that's not close enough to the web application (e.g., scanning Australia from a US scanner) or simply the way the site is built, which may lead to longer load times. Adjusting your scanner location may help prevent having to adjust the more advanced settings that slow the scanner down. Counter-intuitively, slowing the scan speed settings may in fact speed up results on a site that responds slowly, since it will lower the number of queries per second and add less variability to the returned queries.

4. **The scanner acts as a user.**
   The scanner will follow links, press buttons and simulate the actions of a user based on what it can gain access to. It's always possible that the scanner has an undesired interaction on the site as a result of its site discovery phase. For example, if a user can send an email, the scanner will likely fill out forms and press the "send email" button potentially more than once. The scanner itself has no context over what a specific button action may be unless you teach it or exclude either the whole page or page element to avoid it pressing a button you may not want it to. (For more information, view our documentation on Scope Settings.) Keep in mind that excluding page elements to prevent such actions lowers the accuracy of the scan, so consider plans to scan sites like this in pre-production on a regular schedule.

5. **The scanner acts as many users.**
   With its default settings, the scanner effectively operates as several users navigating the website at the same time. On generally well-behaved servers with good capacity, there is typically minimal impact from this activity. However, if the state of the server is unknown, de-tuning the speed of the scan, at least for the first test, will help you understand if there is any potential site impact from simultaneous sessions. For more details on configuring such a test, view our documentation on Advanced Settings.

6. **Tuning is bespoke to each site; it requires effort but may be optional.**
   Bespoke tuning is generally needed for most websites because each web application is different. There are unique structures, sitemaps, third-party libraries, components and custom code that tie everything together. Your investment in tuned scans will depend on resource availability, criticality of the site and impact to the business.

7. **When tuning for authentication, never run a WAS scan as a web site administrator in production – only in test or pre-production environments.**
   Running a web application scan with administrator credentials could create or delete users, or perform other undesired administrative functions.

8. **When tuning for speed, a rudimentary understanding of your sites may help accelerate DAST scans.**

   - First, review the sitemap plugin and associated file attachment.
   - Then, configure your settings: Increase "Network Timeout," or lower "Max Simultaneous Requests" and "Requests per Second," if you experience significant page timeouts, or discover higher than five-second average page response times in the sitemap attachment.
   - Consider speeding up your scan settings if you're obtaining sub one-second responses and only minimal impact to the web server.
   - Deduplicate site content: From the scanner's perspective, site text, image and video content are not tested – input fields and interactions with the page are. If you have redundant pages, such as a site that uses multiple languages, and the code behind the text and images is exactly the same, then you likely need only test one language version of the site.
   - Add additional binary exclusions: WAS doesn't "test" text, images or videos and decide which file extensions to exclude. A default set is provided in the scan scope section but may need to be improved for the site being accessed.
   - Prioritize critical URLs: Identify the critical portions of the application, the ones with forms that likely could reach back to sensitive data. Add those URLs to the scope of your testing, either via "include" in the scan scope section or through manual crawl script. Consider also that these sites may require testing in pre-production.

9. **When tuning for complexity, use session recordings to train the scanner.**
   Performed either by the Tenable Chrome extension or Selenium IDE, & adding within the scope section of a scan configuration. With his process, you can perform manual crawling to ensure that the scanner is able to test a highly complex location within a site. For example, it's possible a site requires a specific series of button presses and/or a specific set of correct input values to get to a page that isn't available any other way. Performing a recording of such a session so that the scanner can later play it back may be the only way to achieve this.

10. **Map out whether there is a web application firewall (WAF), web proxy or load balancer between the scanner and the target.**
    These network devices at best interfere with the scanning and at worst completely invalidate the results. You may think it's sufficient to receive only the "remote" view of results filtered by the firewall; however, it's possible the WAF's built-in protections only prevent one or two methods of executing the flaw. Gaining a full picture of the true state of the site is imperative to being able to make risk-based decisions. Configure your WAF to support bypass functionality, allowing specific IPs or a combination of IP and agent header strings so that the incoming scan can be proven and authorized. A list of Tenable scanner IP ranges are available here.

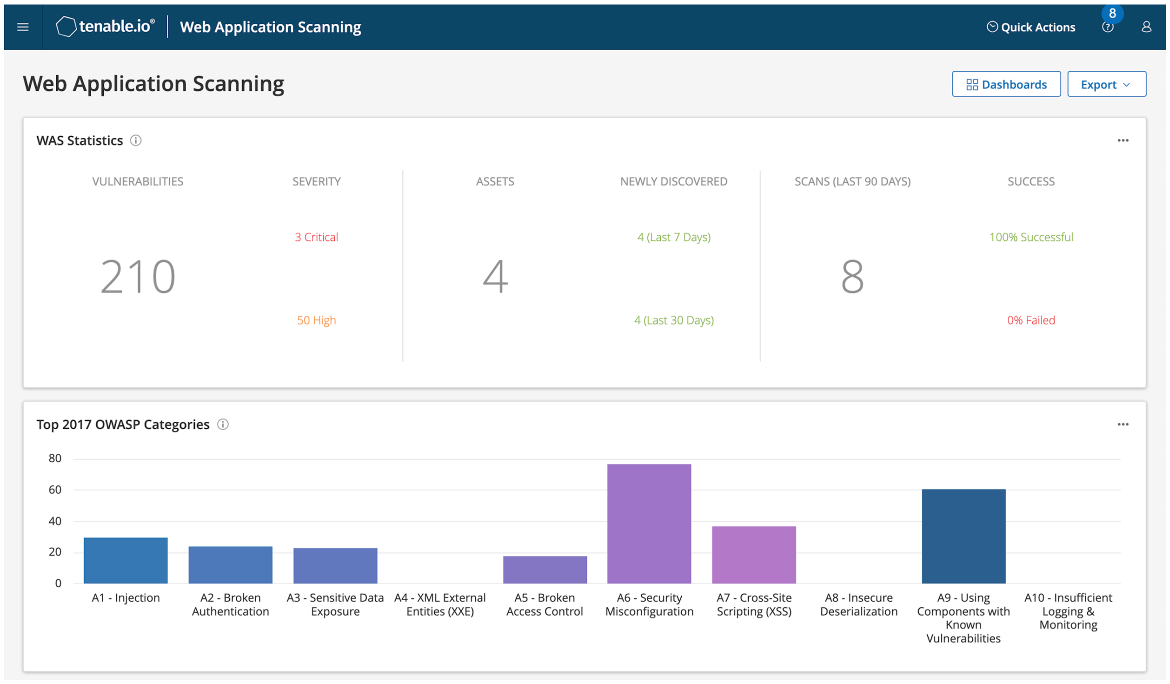
11. **Specific browser identities may be required for some sites.**
    Check whether the application is compatible with the default user agent (configured as "WAS/%v" by default). If not, it may need a specific or commonly available header from a standard browser, such as Mozilla/5.0. It's not unusual for some server-side protections or a web application firewall to require a specific set of results. In this case, you can copy the user agent string from a known browser that can successfully access the site.

12. **Target critical sites with greater care at the outset.**
    Is the target site production-facing, or in any other way critical? What will be the business impact if the web application scanner causes a service impact (e.g., the site falls over)? Always perform the first scan of a site in a controlled manner, either with staff on-hand or within a pre-production environment. Once you understand the nature of the site, then full automation can begin.

For more information and guided product walk-throughs, visit our Tenable Product Education YouTube channel. These short, instructional videos explain how to make the best use of WAS, including the authentication and tuning procedures mentioned above, so that you can secure your vulnerable web applications.

6. What Is the OWASP Top 10?

The Open Web Application Security Project (OWASP) is an international non-profit organization dedicated to web application security. The OWASP Top 10 is a benchmark report, the current version was released in 2017 and slated for an update in 2021, that outlines the most critical security risks for web applications. The OWASP Top 10 is put together by a team of security experts from all over the world and contains recommendations that all companies can incorporate into their processes to minimise and mitigate security risks. In short, anything on the Top 10 should be considered the most commonly abused web app security flaws, and therefore the greatest risks that should be tackled with priority. These OWASP Top 10 categories are prominently displayed in various widgets within the Tenable.io interface.


tenable.io
Web Application Scanning
Quick Actions
8
Web Application Scanning
Dashboards
Export
WAS Statistics
VULNERABILITIES
210
SEVERITY
3 Critical
50 High
ASSETS
4
NEWLY DISCOVERED
4 (Last 7 Days)
4 (Last 30 Days)
SCANS (LAST 90 DAYS)
8
SUCCESS
100% Successful
0% Failed
Top 2017 OWASP Categories
80
60
40
20
0
A1 - Injection
A2 - Broken Authentication
A3 - Sensitive Data Exposure
A4 - XML External Entities (XXE)
A5 - Broken Access Control
A6 - Security Misconfiguration
A7 - Cross-Site Scripting (XSS)
A8 - Insecure Deserialization
A9 - Using Components with Known Vulnerabilities
A10 - Insufficient Logging & Monitoring

On its website, OWASP describes its current Top Ten application security risks as follows:

1. **Injection**
   Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorisation.

2. **Broken Authentication**
   Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

3 **Sensitive Data Exposure**
Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

4 **XML External Entities (XXE)**
Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

5 **Broken Access Control**
Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorised functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

6 **Security Misconfiguration**
Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

7 **Cross-Site Scripting XSS**
XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

8 **Insecure Deserialisation**
Insecure deserialisation often leads to remote code execution. Even if deserialisation flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

9 **Using Components with Known Vulnerabilities**
Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

10 **Insufficient Logging & Monitoring**
Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

7. Implementation Methodology

1. Preparation for Deployment

(a) **Confirm requisite access to the Tenable.io platform and WAS application.** Users must be created with appropriate access to the Tenable.io WAS product for scanning and viewing of results. You can configure Role-Based Access Control (RBAC) to allow this access. Administrative credentials are required for configuration.

(b) **Determine whether a local scanner is needed.** Local or cloud-based scanners can be deployed and connected to Tenable.io. You can use these scanner(s) on internet-facing web applications, as well as development or pre-production environments (if firewall rules are suitably configured).

The Tenable Core + WAS scanner supports installation on VMware (.ova), Hyper-V (.zip) or a physical machine (.ISO). It can be deployed locally on-premises or within a cloud-based development environment to scan non-internet-facing web applications.

The local scanner can be downloaded here, and will need:

- Outbound access to https://cloud.tenable.com via port 443 to communicate with Tenable.io.
- Inbound access via https on port 8000 for web browser access to the management interface.

2. Identification and Planning

(a) **Define the security objectives.** Why are we scanning, what do we hope to achieve and what does success look like?

(b) **Determine scanning priorities.** Identify which target web applications are within the scope of quick scanning and which require more detailed scanning.

(c) **Ensure full coverage.** Determine whether there are any other (possibly unidentified) web servers, services or applications that may need to be scanned, and how they may be found.

3. Scanning, Tuning and Optimization

(a) **Establish a baseline.** Configure some quick scans to provide a high-level assessment of the target with respect to component vulnerabilities, configuration audits, SSL/TLS templates and web application vulnerabilities. For guidance on creating scans, refer to this document.

(b) **Achieve a more detailed view.** Run a scan based on either the "Overview" or "Scan" template, where the results will start providing you more detailed insight. Use this first scan to help tune if that's part of your program:

- analyse the findings;
- use the sitemap crawled as an input to detailed scanning, tuning and optimization, reviewing for page timeouts, length of time to access a page, errors or opportunities to remove repetitive content;
- review the "Scan notes" for any higher priority concerns, which may provide suggestions for scan improvement.

For guidance on viewing scans, refer to this document.

(c) **Experiment with advanced settings.** Perform scan tuning in a few locations based on the data gathered in the previous step. You can then update and deploy the scan for the targeted web applications. Click on the relevant topic below for further information on scan settings:

- Scope Settings
- Assessment Settings
- Advanced Settings

(d) **Consider further bespoke adjustments.** Every application is unique. Running scans and analyzing the results will provide experience in which particular techniques can help optimize your assessments to run most efficiently and ensure coverage of all areas of the application. Depending on the size or complexity of the web application, the scan may complete and the results can be further analysed and potentially optimized further. For example, regularly reviewing the "scan notes" after a scan completes & reviewing the attachment to the sitemap plugin on a regular basis is highly recommended.

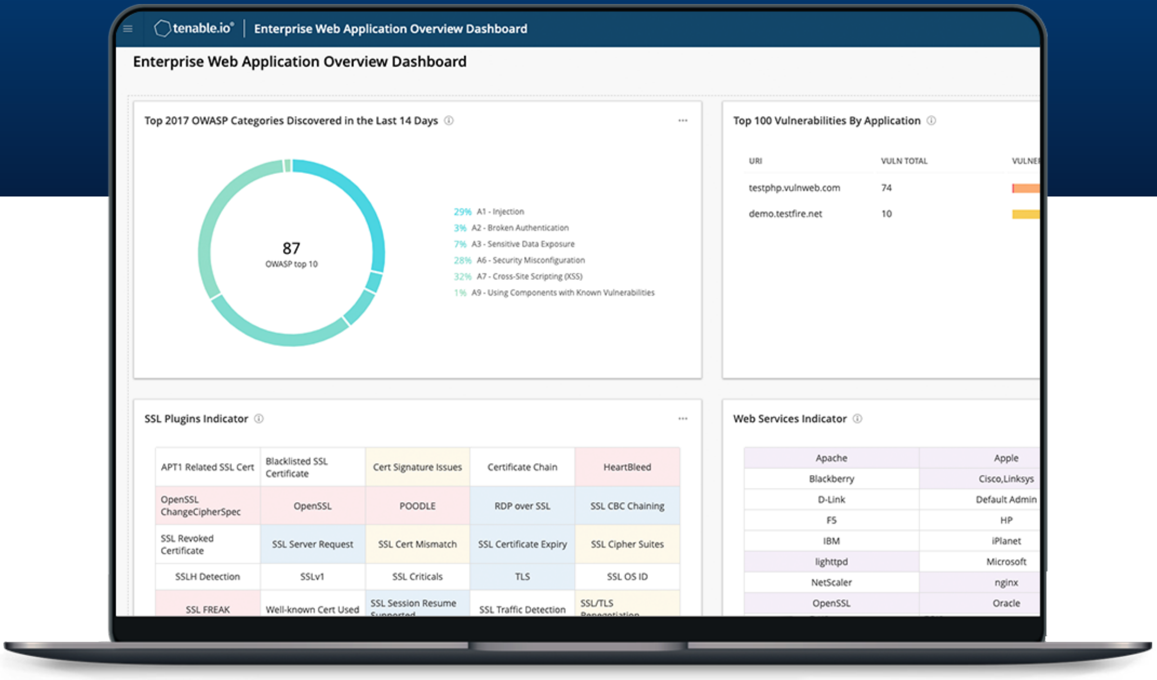
4. Documentation

(a) **Track everything.** Produce and manage documentation that captures full details of the deployment requirements, deployed scanner resources (if applicable), web applications identified for scanning and the tuning applied to the scans with accompanying rationale.

(b) **Communicate your findings.** Establish reporting requirements to identify: who gets reports, and at what level of detail, as well as how often the reports are distributed. Developers may need PDFs while ticketing systems require vulnerability details. Management often prefers a higher-level summary of overall exposure and risk reduction.

About Tenable

Tenable®, Inc. is the Cyber Exposure Company. Over 30,000 organizations around the globe rely on Tenable to understand and reduce cyber risk. As the creator of Nessus®, Tenable extended its expertise in vulnerabilities to deliver the world's first platform to see and secure any digital asset on any computing platform. Tenable customers include more than 50 percent of the Fortune 500, more than 30 percent of the Global 2000 and large government agencies. Learn more at www.tenable.com.



Enterprise Web Application Overview Dashboard

Top 2017 OWASP Categories Discovered in the Last 14 Days

29%	A1 - Injection
3%	A2 - Broken Authentication
7%	A3 - Sensitive Data Exposure
28%	A6 - Security Misconfiguration
32%	A7 - Cross-Site Scripting (XSS)
1%	A9 - Using Components with Known Vulnerabilities

Top 100 Vulnerabilities By Application

URI	VULN TOTAL	VULNEI
testphp.vulnweb.com	74	
demo.testfire.net	10	

SSL Plugins Indicator

APT1 Related SSL Cert	Blacklisted SSL Certificate	Cert Signature Issues	Certificate Chain	HeartBleed
OpenSSL ChangeCipherSpec	OpenSSL	POODLE	RDP over SSL	SSL CBC Chaining
SSL Revoked Certificate	SSL Server Request	SSL Cert Mismatch	SSL Certificate Expiry	SSL Cipher Suites
SSLH Detection	SSLv1	SSL Criticals	TLS	SSL OS ID
SSL FREAK	Well-known Cert Used	SSL Session Resume	SSL Traffic Detection	SSL/TLS

Web Services Indicator

Apache	Apple
Blackberry	Cisco,Linksys
D-Link	Default Admin
F5	HP
IBM	iPlanet
lighttpd	Microsoft
NetScaler	nginx
OpenSSL	Oracle

6100 Merriweather Drive
12th Floor
Columbia, MD 21044

North America +1 (410) 872-0555

www.tenable.com

051821 V01

COPYRIGHT 2021 TENABLE, INC. ALL RIGHTS RESERVED. TENABLE, TENABLE.IO, TENABLE NETWORK SECURITY, NESSUS, SECURITYCENTER, SECURITYCENTER CONTINUOUS VIEW AND LOG CORRELATION ENGINE ARE REGISTERED TRADEMARKS OF TENABLE, INC. TENABLE.SC, LUMIN, ASSURE, AND THE CYBER EXPOSURE COMPANY ARE TRADEMARKS OF TENABLE, INC. ALL OTHER PRODUCTS OR SERVICES ARE TRADEMARKS OF THEIR RESPECTIVE OWNERS.