WHITE PAPER

Invicti

How to Secure Thousands of Websites with a Small Security Team

EXECUTIVE SUMMARY

This whitepaper examines how you can use integration and automation of enterprise web security to combat the growing cybersecurity talent shortage, which, according to Forbes, currently stands at approximately 3 million job openings. We focus on four key factors that may be addressed using such an approach: awareness, responsibility, protection, and validation.

Technologies that are now available in selected web security solutions enable enterprises to automate many processes that were traditionally manual:

- **Web assets may be automatically identified and inventoried using crawler technology pioneered by search engines.**
- **Vulnerabilities may be automatically proven using safe exploits to significantly reduce the number of false positives, thus reducing time and resource costs, as well as substantially improving scalability.**
- **Proven vulnerabilities may be automatically assessed on the basis of both technical and organizational factors.**
- **Assessed vulnerabilities may be efficiently managed using integration with tools that are already used to manage tasks and issues during software development.**
- **Web security processes may be implemented at the earliest possible stage of software development using integration with DevOps solutions, thus significantly reducing the cost and improving the ease of remediation.**

The whitepaper explains the details of these methods and technologies, as well as shows the benefits that they bring.

INTRODUCTION: CHALLENGING THE CYBERSECURITY TALENT GAP

The cybersecurity talent gap is not going anywhere. Quite the opposite; it's getting worse with time. This calls for immediate solutions.

The cybersecurity talent gap is being addressed in two primary ways. The first way focuses on increasing available resources. Universities are promoting more cybersecurity programs for IT students. Businesses are trying to find incentives to convince independent security professionals to work for them full-time. Governments are encouraging more young professionals, especially women who are underrepresented in the industry.

The second way to address the cybersecurity talent gap is by lowering the resource requirements. Enterprises shift the responsibility for cybersecurity from dedicated teams onto other roles – not only administrators or developers but non-technical roles as well, all of which have the potential to increase risk significantly. Alongside this trend of enlisting underskilled professionals and people who have other day jobs in security management, researchers are exploring ways to use innovative technologies such as artificial intelligence to fill in for unavailable humans. While machine learning and artificial intelligence hold promise for security in some use cases, they are no more a substitute for a skilled security professional than people underskilled in security.

These approaches are insufficient to solve the problem today. Organizations must also rely on automation coupled with integration to expand the capacity of small security teams. Large organizations need innovative and comprehensive solutions built specifically for them to help address the cybersecurity talent gap. This eases the problem today, alongside increasing the number of security graduates in universities and exploring the potential of artificial intelligence.

AWARENESS, RESPONSIBILITY, PROTECTION AND VALIDATION

The basis for a successful cybersecurity strategy is educating the organization about potential dangers and how to avoid them. However, even the best training won't suffice if employees don't feel responsible for security. And even the best efforts in promoting awareness and responsibility fail unless there is a way to validate the effectiveness of those efforts and support them with automated tools for validation and protection.

Comparison of security approaches for two major cybersecurity threats

THREAT	PRIMARY CAUSE	AWARENESS	RESPONSIBILITY	PROTECTION	VALIDATION
Phishing and malware (including ransomware)	Users are careless and have excessive trust in the received information	How do I recognize phishing and malware?	I am responsible for reporting and potential consequences	Antivirus/ antispam tools (partial protection only)	Simulated attacks – fake phishing
Web vulnerabilities	Developers are careless and have excessive trust in users	How do I avoid introducing vulnerabilities?	I am responsible for avoiding vulnerabilities and potential consequences	Web application firewalls (partial protection only)	Simulated attacks – vulnerability scanning and penetration testing

For example, in the case of phishing, organizations should first teach users how to recognize such attacks. Then, they should make sure that every user realizes their responsibility to avoid and report phishing attempts. Finally, organizations should carry out exercises to test how well users can react to a fake phishing attempt. Additionally, organizations should use automated systems such as antispam and antivirus tools, which have the ability to spot and neutralize the majority of phishing attacks.

The same rules apply to web attacks that are the other most prominent cause of security breaches next to phishing. Everyone involved in developing web resources must be aware of potential vulnerabilities and feel responsible for eliminating them. There are limited protection methods such as web application firewalls, but they do not get to the root of the problem.

In the case of web assets, selective validation is not enough. Every web asset in the company can and should be verified thoroughly. Such verification, if done manually, would be impossible due to the sheer number of such assets and potential vulnerabilities. That is why in the case of web security, the only efficient solution is to rely as much as possible on automation and integration.

ASSET IDENTIFICATION

To protect your assets, you must first know them. In the case of websites and web applications, asset identification and inventory cannot be limited to primary production websites. They are only the tip of the iceberg. The following additional web assets must also be included:

1

TEMPORARY WEB ASSETS

This includes assets created for one-time marketing campaigns, demo assets for customers, etc. Such assets are the most elusive, and they can easily escape identification because they are often perceived as low-risk.

2

PRE-PRODUCTION WEB ASSETS

This includes staging assets, UAT (user acceptance testing) assets, OAT (operational acceptance testing) assets, QA (quality assurance) assets, even development assets. Organizations may perceive these assets as low-risk, but vulnerabilities in such assets ultimately end up in high-risk ones (production), too.

3

THIRD-PARTY WEB ASSETS

This includes assets maintained by third parties that are still associated with the organization (for example, use a subdomain of the organization's top-level domain). These assets are challenging to include in organizational processes, but they may cause major harm to the company's reputation if exploited.

Legacy approach: manual asset identification

To identify as many assets as possible (preferably all of them), organizations first and foremost need to build a data source. This must be a central database with information on all identified assets. If this information is scattered across teams, departments, or offices, it cannot be used efficiently.

The data source is not only crucial for awareness, but it must also serve as the origin of information for tools that follow up on security. Of course, in a large organization, such a data source may need to be continuously fed with information from secondary data sources.

The existence of the central data source is not enough to ensure that assets are identified. The more significant issue is making sure that the data source is not only initially filled but regularly updated. This may mean even several updates a day in a large organization – if you maintain thousands of websites, new ones may even appear daily.

The traditional approach to maintaining such a data source is via organizational processes. Such processes require that all personnel involved in developing and maintaining websites, especially those formally responsible, keep the data source updated. However, if this is a manual process, it is prone to human error. And even one error might mean that an utterly unprotected website slips past the process. An attacker needs just one such website to wreak havoc on company assets and reputation.

Additionally, such manual processes often fail if they include external entities. If third parties are involved in asset development or maintenance, they are frequently either unaware of reporting procedures or don't pay enough attention to them.

The identification process for websites may be partially streamlined using custom-built scripts, specialized inventory software, or manual scanning using tools such as network scanners. All these must also support the data source's format, making the process even more challenging to orchestrate.

The modern approach: automated asset identification

The most efficient way to ensure the highest possible level of identification is through the use of software that is explicitly designed to build an inventory of web assets, and that is tightly integrated with a vulnerability scanning solution (preferably, an integral part of such a solution). Such software has enormous advantages:

+ It provides a single central data source for all web assets. It also includes import mechanisms for secondary data sources. All in all, this guarantees that all the information is available in one authoritative location.
+ It has mechanisms to discover all the web assets automatically. Such mechanisms include both name-based discovery (for example, Internet scans, public certificate repositories, relying on open-source intelligence) and location-based discovery (for example, network scanning). These mechanisms automate identifying all types of assets mentioned above: production, pre-production, temporary, and third-party assets.
+ Data may also be entered manually for any assets that are impossible to identify automatically. Therefore, it may be included in manual processes, thus providing complete coverage.

If such a solution is integrated with tools for vulnerability scanning, vulnerability assessment, and vulnerability management, it means that the identification process works in unison with further stages.

VULNERABILITY ASSESSMENT AUTOMATION

In a large organization, there are too many errors and vulnerabilities regularly discovered to be able to fix them all immediately. Therefore, the order of fixing is a matter of prioritization. And with a large number of assets, those priorities may be quite complex. There are multiple aspects that the organization must consider when deciding which vulnerabilities to focus on first:

1

THE SEVERITY OF THE VULNERABILITY

Some vulnerabilities pose little threat, while some may allow the attacker to take over the whole system or give them access to an entire database full of sensitive data.

2

THE CRITICALITY OF THE ASSET

An asset with simple marketing information obviously has a lower priority than a mission-critical web application that stores sensitive data.

3

THE SCOPE OF ACCESS

For example, an external web application is more apt to be attacked than an internal system. And a vulnerability that requires the attacker to be authenticated is less critical than one that can be exploited by an anonymous visitor.

4

THE POTENTIAL FOR ESCALATION

A vulnerability in a system that is not interconnected with other systems is less critical than a vulnerability in a system from which an attacker may escalate to other assets.

Manually assessing vulnerabilities in thousands of web assets is impractical. Therefore, a large organization must have a way to automate the process by assigning weights to each factor and applying them to each vulnerability right after it is identified.

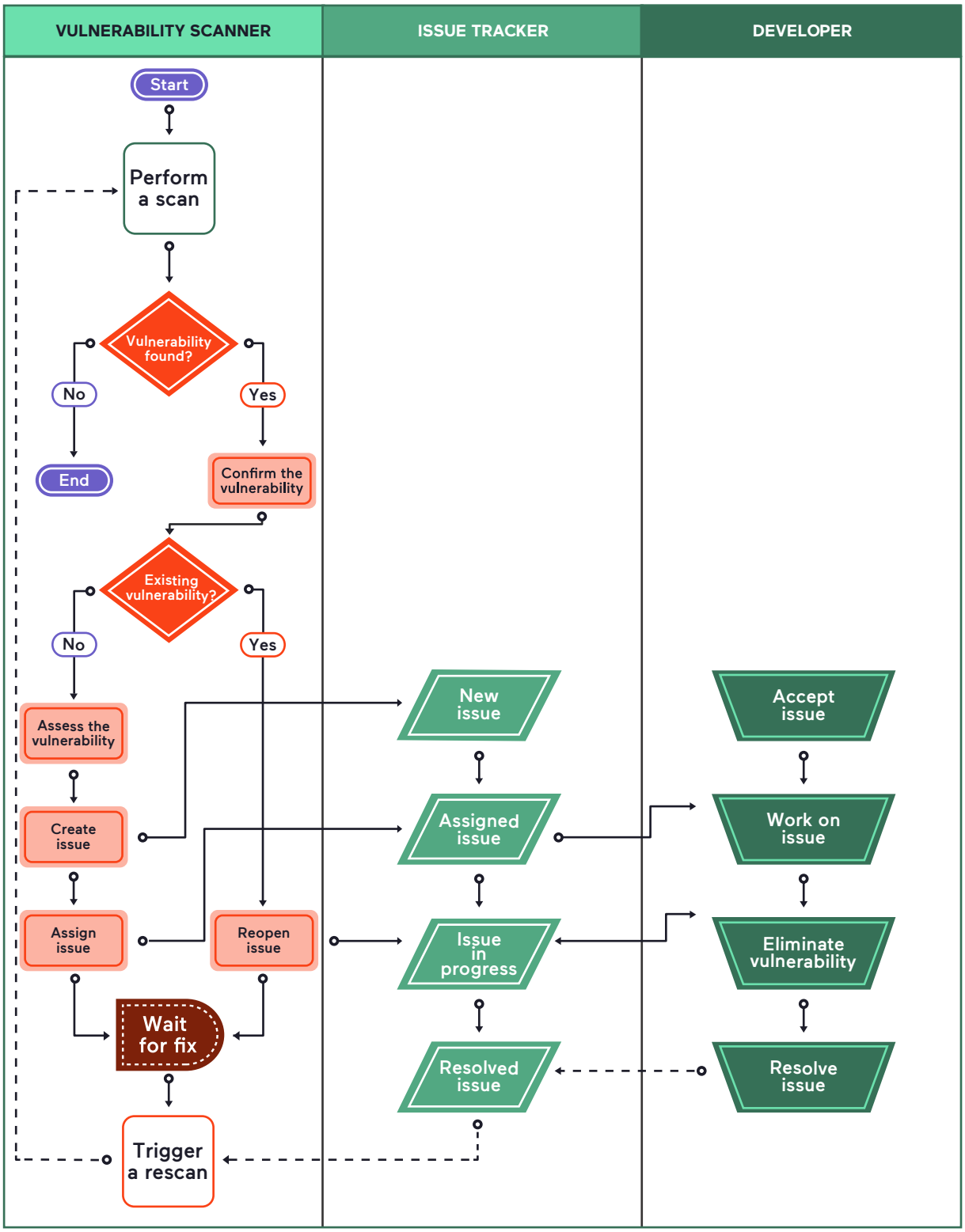
Obviously, even if this process is automated, it must also be centralized and closely coupled with the identification process. Each identified asset must be scanned for vulnerabilities, and then those that are found must be automatically assessed. It is possible with a multitude of interconnected tools, but it works much better if all the functionality is part of a single integrated solution.

VULNERABILITY MANAGEMENT AUTOMATION

Finding vulnerabilities is just the beginning of a complex process. For a large organization, manually managing thousands of such processes is impossible. Again, the only efficient solution is automation.

The vulnerability management process involves the following primary stages:

- It begins with finding the vulnerability and confirming that it's not a false alarm.
- The vulnerability must then be assessed based on multiple factors.
- It must then be treated as an issue and assigned to a person or a team responsible for managing it.
- The issue must then be monitored for state changes resulting from other processes and manual input.
- When the issue is marked as resolved, the vulnerability must be retested. This must not be optional.
- Only when the vulnerability scanner confirms that the vulnerability is no longer present can the issue be closed. If not, it must be kept open.
- Even when the issue is marked as resolved, it must not be forgotten. If the scanner finds the same vulnerability in the same application in the future, the new find must be linked to the previous issue. This saves a lot of resources on identifying the cause and helps resolve the reoccurring problem much quicker.



Some parts of the above process may be automated, but some will remain manual. A large organization will want to automate as much as possible, and therefore the only steps that should remain manual are those associated with fixing the vulnerability. In an agile environment, the product/service owner or team leader will have to oversee the issue management process, for example, manually allocate the vulnerability to a sprint and reassign it to a particular developer.

The vulnerability management process requires that the software solution is

adapted to the needs of development teams. Development teams already heavily rely on issue management solutions, both for adding new functionality and fixing bugs. Vulnerabilities should be treated exactly the same way that bugs are treated (those discovered by automated and manual testing). It would be highly inefficient if developers were forced to use different issue management systems for different types of issues.

Therefore, the key to successful vulnerability management is tight integration with issue management systems. The vulnerability management system must be able to create and manage issues in issue management systems and must be able to react to issue changes. If so, the process is transparent to all parties involved and highly efficient.

Automation in web security

To keep pace with growing attack surfaces and increasing threat levels, we must automate everything that can be practically automated. However, some processes still need to be handled manually:

+ **Automatic and continuous identification of all web assets**
+ **Automatic and comprehensive vulnerability scanning for all identified web assets**
+ **Automatic vulnerability confirmation (proving) to eliminate all false positives**
+ **Automatic vulnerability assessment based on multiple factors**
+ **Automatic creation of issues for development teams**
  + Manual sprint management by product/service owners or team leaders
  + Manual code repair by developers to eliminate the vulnerability
+ **Automatic vulnerability retesting**
+ **Automatic issue management: closing, reassigning, reopening**
+ **Automatic vulnerability archival**
+ **Automatic linking of reoccurring vulnerabilities with original issues**

AUTOMATICALLY PROVING VULNERABILITIES

One of the critical aspects of introducing automation to vulnerability assessment and management is being able to trust the automatic tool. The vulnerability management process consumes a lot of resources. If the vulnerability is ultimately found to be invalid (a false positive), the resources are wasted.

When a developer attempts to fix a false positive, they usually need much more time than for a real vulnerability. First, they have to try to replicate the vulnerability. If they cannot do so after several attempts, they have to authoritatively decide that there is no vulnerability and take responsibility for this

decision. This may also involve additional resources, for example, a dedicated penetration test to confirm the diagnosis. Therefore, resources consumed by a false positive are substantially greater than in the case of real vulnerability.

In smaller organizations with few assets and few vulnerabilities, the false positive rate is often not a significant problem because the total number of false positives is relatively small. However, in a large organization with thousands of assets, even a seemingly negligible positive rate may mean several false alarms appearing in every sprint. This, in turn, causes the security and development teams to lose trust in the vulnerability scanner. It can cause issues to be treated less seriously, or it may ultimately lead the organization to stop vulnerability scanning altogether due to the associated cost.

Real issue vs. false positive

REAL ISSUE	VS	FALSE POSITIVE
The vulnerability scanner detects an issue		**The vulnerability scanner detects an issue**
The developer fixes the issue		**The developer tries to find the issue and fails**
The vulnerability scanner confirms the issue as fixed		**The developer challenges the vulnerability scanner**
The problem is solved ✓		**The security team might get involved in a discussion**
		The vulnerability scanner still detects the issue
		The security team trusts the vulnerability scanner less
		The problem is not solved, and a lot of time is wasted ✗

Therefore, an enterprise-class vulnerability scanning solution should ideally minimize the false positive rate. In theory, this seems impossible, but it depends on how the vulnerability scanner is built. If a scanner performs its diagnosis based on signatures or simple patterns, you can never reduce false positives. The only viable approach is for every vulnerability to be actually exploited. If the scanner sends a payload and, for example, gains access to unauthorized data (such as the ***/etc/passwd*** file in Linux/UNIX), the vulnerability is one hundred percent certain. And this means that no time and effort is ever wasted by developers or security researchers to try to prove that such a vulnerability exists. However, the most significant advantage of proving that a vulnerability is genuine and not a false positive is the feeling of certainty. If the scanner can prove/confirm that, for example, 94% of direct-impact vulnerabilities - issues that could get your websites and applications hacked right away - are undoubtedly athentic, these vulnerabilities do not require any manual retesting, and you can send them straight to the developer. On the other hand, the remaining 6% may be retested manually before being assigned to the developers to ensure that there are no false positives in the production cycle.

EARLIEST POSSIBLE DETECTION

All organizations, independent of their size, should try to identify vulnerabilities as early as possible. Whenever a fragment of code is added or changed, it is consecutively introduced into the application on various systems, for example:

1. **On the developer's local machine**
2. **In the build instance created by the CI/CD system**
3. **In the internal testing/QA environment**
4. **In the staging/OAT/UAT environment**
5. **In the production environment**

It is impractical for every developer to install and run a vulnerability scanner on their own machine after every code change and compilation. It would also be impossible to enforce. However, in agile environments, changes are committed by the developer to the repository, and the CI/CD system immediately builds the application to test if it can be compiled correctly. The CI/CD system then performs a series of automated tests. This is the perfect spot to include a vulnerability scan as well.

There are many advantages to finding vulnerabilities at such an early stage. First of all, if a vulnerability is found at any later stage, it must go back to the developer, and the build process must be repeated. Therefore, it consumes unnecessary resources and delays the release. In the worst possible case, if a vulnerability is found on a production system, it may even require the release to be reverted and delay the re-release by several days or weeks (depending on the deployment procedures and resource availability).

Example of late scanning vs. early scanning

LATE SCANNING	VS	EARLY SCANNING
1. Application deployed on the local machine (day 1)		1. Application deployed on the local machine (day 1)
2. Application deployed in the CI/CD system (day 1)		2. Application deployed in the CI/CD system (day 1)
3. Application deployed in the QA environment (day 2)		3. A vulnerability scan performed, and issues found (day 1)
4. Application deployed in the staging/OAT/UAT environments (day 5)		4. Repeat steps 1-3 (day 2)
5. Application deployed in the production environment (day 14)		5. Safe application deployed in the QA environment (day 3)
6. A vulnerability scan performed, and issues found (day 15)		6. Safe application deployed in the staging/OAT/UAT environments (day 6)
7. Repeat steps 1-6 (day 30)		7. Safe application deployed in the production environment (day 15)
8. Safe application deployed in the production environment (day 31)		
TOTAL DELAY: 15 DAYS		TOTAL DELAY: 1 DAY

The additional advantage of early discovery is that the code is fresh in the mind of the developer. If a developer worked on a piece of code, committed it to the repository, and then receives a notification about a particular vulnerability within a very short time, they still remember the code that they wrote and have no doubt about where the vulnerability could have been introduced. On the other hand, if a vulnerability is found on a production system, the code might have been originally committed even weeks before, which means that even the original developer doesn't know what code changes caused it.

To achieve top efficiency, you must integrate enterprise-class vulnerability scanning/assessment/management solutions with CI/CD systems. The more configurable the integration, the better. DevSecOps should be able to define thresholds for warnings and failures. The scanner should also be able to perform incremental scans. If only a tiny code piece was added to one module of an application, scanning the whole application from scratch consumes unnecessary resources and takes a lot of time. In enterprise environments, applications may even require several builds a day, so every minute saved on scanning is very valuable.

CONCLUSION: EFFICIENCY, NOT PERFECTION

Some organizations may have a misconception that automated web security solutions are supposed to find every possible vulnerability. This is not true, and it will never be true. While automated tools have a very high success rate, they won't be able to replace security researchers and independent white hat hackers. However, they are excellent at replacing mundane and repetitive tasks that are a waste of time for professionals.

This was not the initial role of web vulnerability scanners. At first, they were built as tools that would help security researchers with their manual work, and many of them are still meant for that role. On the other hand, modern web security solutions encompass tasks in the whole organization, which would otherwise be performed by different roles, including product/service owners and team leaders, developers, and operations.

Leading-edge comprehensive web security solutions aim to eliminate simple tasks and support decision-making (for example, by pre-assessing and pre-assigning vulnerabilities). The ultimate goal is for the valuable cybersecurity talent not to be wasted on something that a machine can do. With the support of such solutions, you can easily keep the web assets secure with a small team, even in a huge organization.

Invicti

Invicti Security is transforming the way web applications are secured. An AppSec leader for more than 15 years, Invicti enables organizations in every industry to continuously scan and secure all of their web applications and APIs at the speed of innovation. Through industry-leading Asset Discovery, Dynamic Application Security Testing (DAST), Interactive Application Security Testing (IAST), and Software Composition Analysis (SCA), Invicti provides a comprehensive view of an organization's entire web application portfolio and scales to cover thousands, or tens of thousands of applications. Invicti's proprietary Proof-Based Scanning technology is the first to deliver automatic verification of vulnerabilities and proof of exploit with 99.98% accuracy, returning time to development teams for critical projects and innovation. Invicti is headquartered in Austin, Texas, and serves more than 3,500 organizations all over the world.

www.invicti.com