WHITEPAPER

INDUSFACE™

Why **Manual Pen-Testing** is a must have for comprehensive application security testing

ABOUT THE WHITEPAPER

According to Gartner, the Application Security Testing (AST) Market, as it's called, will be about USD 1.33 bn by the end of 2020 with a CAGR of 10% year on year. In the same breath, other researchers suggest that the Penetration Testing Market was valued at USD 1.12 bn in 2019 and is projected to reach USD 5.6 bn by 2027, growing at a CAGR of 23.95% from 2020 to 2027. With the advent of DevSecOps and shifting left concepts, though automated testing is touted as the way forward, then why is manual pen-testing estimated to grow at a higher rate than automated scanning? This paper tries to address this question and talks about why manual pen-testing is very important and how it should be complemented with automated scans to ensure comprehensive detection.

This whitepaper will address the following questions:

- What is manual pen-testing, and how is it different from automated pen-testing
- Is manual pen-testing required
- How can automated scans be complemented with manual pen-testing
- Compliance requirements
- The Application Pen-testing process

Who should be reading this whitepaper?

Anyone looking for a comprehensive way to find vulnerabilities in their application should read this paper. If they are confused between automated scans vs manual pen-testing, and how they are different, this paper will provide them some clarity. Further, it will also help in their buying decisions.

SUMMARY

The Application Security Testing (AST) market is very diverse, with multiple tools and means available. The advent of rapid development has drastically changed the way application security testing is approached, with tighter collaboration of security engineers in the development cycle. Automation plays a key role in bringing in efficiency, especially with the adoption of DevSecOps.

Though automation is important and automated tools are critical for efficiency, the importance of manual pen-testing cannot be ignored. Manual intervention is primarily required to:

- Identify business logic flaws. Attackers may exploit flaws in business rules to perform attacks or cause harm. Automated scanners are good at identifying vulnerabilities in code, but any flaws in business rules need human intervention
- Identify complex vulnerabilities like IDOR or complex chain attacks. Often, human intervention is required to find possibilities of complex attacks, including chain attacks wherein certain information gained from low level attacks is used in combinations to launch far more sophisticated attacks
- Understand the context of the application and identify the actual severity of the vulnerabilities. The accurate risk to an application can only be determined by manual intervention. Though automated scanners will find vulnerabilities, the severity of those vulnerabilities is bound to vary depending on the application context
- Every Pen-tester uses automation and automated scanners, but they use it more in the context of improving the efficiency, and ideal automated scanners are those which work with and provide options for manual pen-testers to automate stuff based on application context, and also do it on scale so that pen-testers can concentrate on high value stuff

In a nutshell, anything encountered commonly like frameworks or attack patterns will be automated but it takes human intervention to figure out new threat scenarios. A combination of automated and human intervention is ideal - the human gets to 'think' like an attacker, while automation can do fuzzing; cover much greater volumes of attacks; and run frequently.

It is recommended that as a part of the application testing strategy, manual pen-testing is not to be completely replaced by an automated scanner but used to complement them! And, depending on the criticality of the application and the frequency of change in it, pen-testing can be done in regular/periodic intervals.

Note - *If you are not aware of common terminologies like DAST, black box , grey box testing etc, Please refer to our* *Appendix section.*

DYNAMIC APPLICATION SECURITY TESTING (DAST)

The Application security market is diverse with many complementary techniques and methodologies. For effective strategy, organizations may choose to employ more than one technique or methodology. In context of this whitepaper, we will be concentrating on Dynamic Application Security Testing (DAST).

As DAST is closer to life, the application is tested on real run-time to see if there are any vulnerabilities in application that are exploitable. This is done by mimicking the techniques that hackers would employ to find the vulnerabilities. With we use agile methodology and constant deployment of codes, other techniques become time consuming or just bottlenecks. And even if they are employed, it becomes very important to identify vulnerabilities that are present at run-time using DAST techniques, as these are the vulnerabilities that will be exploited by hackers.

DAST techniques are employed either through completely automated scanners or with manual intervention. When manual intervention is employed, it is more commonly called as manual pen-testing, wherein, the expertise of humans/ security is used to hack the applications to see if any vulnerability is exploitable. Quite often, such techniques are labelled as 'ethical' hacking. Many vendors do claim or use the term automated pen-testing, but it is not the same as manual pen-testing. Automated pen-testing is a more glorified name for automated scanners.

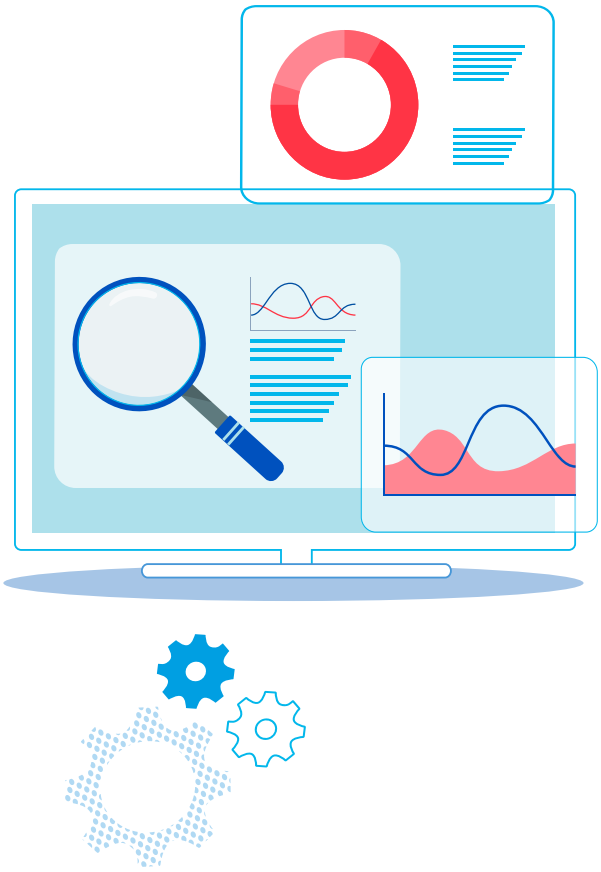
As we see it, there are 3 kinds of vulnerabilities:

- Vulnerabilities that are effectively found through automated scanners like documented vulnerabilities, vulnerable components, software configuration etc. (explained in the previous section)
- Vulnerabilities that can be found **only with** manual intervention, like business logic vulnerabilities, chain attacks, bypassing WAF etc.
- Vulnerabilities that can be identified only **through a combination of automated scanners & manual pen-testing.** Automated scanners cannot be relied to find these vulnerabilities without humans, and they will have to rely on automated tools to successfully exploit the same. For example, in the case of IDOR, DOM-based XSS etc., human and automated tools should work together to identify the vulnerabilities.

Automated Scanners

Before delving into manual pen-testing and its need, let us try and understand what automated scanners are and how they work. Automated scanners are good tools to identify standard, documented vulnerabilities in the application. DAST scanners, simulate attacks against an application at run-time and analyses the application reaction to determine if they are vulnerable. Generally, there are two steps to finding vulnerabilities through automated tools - crawling, where real time user actions are mimicked by crawlers; and generating requests which are then simulated/replayed with malicious attack patterns to see how the application reacts.

Such scanners are good at finding vulnerabilities like XSS (Self, stored and reflected), SQL injections, sensitive information disclosure, missing configurations, components with known header etc. It should be noted that automated scanners, though they are good at identifying vulnerabilities in an application, will not be able to accurately judge the risk/severity of it. For this, 'context' of the application is needed which is usually not available with automated scanners. Automated scanners should be used as a good starting point as it helps in identifying well documented vulnerabilities. With agile development processes being rapidly adopted worldwide, automated scanners provide the necessary automation to bring in efficiency in the security testing process. Hence, regular daily/weekly scans and sanitization through automated scanners is a must and should be incorporated as a critical part of the security strategy of any company. Identifying the right automated scanner is the key, especially with single page applications (as they are dynamic). Crawlers needs to be JS aware and understand the browser context.



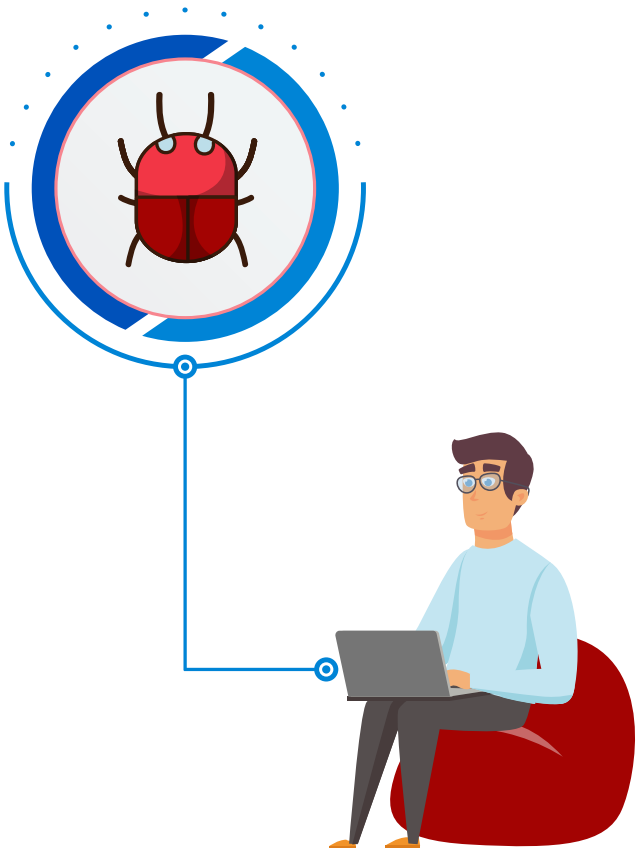
Note - *to learn more, check out our Scanner whitepaper.*

Manual Pen Testing

Manual Pen-Testing, as the name implies, is human intensive testing. Modern web applications have a lot of intricacies and complex vulnerabilities that can only be found with manual intervention. It should be noted that all pen-testing is done by humans with the help of automated scanners/tools. As a first step, testers rely on wide-spectrum application scanning tools, and from the report of these scanners, testers get an idea of the potential vulnerabilities in the application. They then use the intelligence gathered to further exploit them, based on their application context. While exploiting, testers may use automated scripts to effectively exploit the same and see if the application is actually vulnerable.

Vulnerabilities found only by Manual-PT:

These vulnerabilities can only be found using human intervention. Some common examples of these types of vulnerabilities are chain attacks, business logic vulnerabilities etc.



Chain attacks are exploits, where multiple common/low severity vulnerabilities are used in conjunction to identify a crown jewel. Human intervention then needs to understand the context of the application and vulnerabilities found and then decipher how certain vulnerabilities can be used together to escalate the attack. This depends on the human skill set and experience. Automated scanners cannot be taught to do this.

Another common example is business logic vulnerabilities, wherein testers try and understand the business workflows and manipulate them and gauge if any exploits are possible. Such vulnerabilities and their levels of risk cannot be found without human intervention.

Some common example are vulnerabilities like price manipulation, privilege escalation etc.

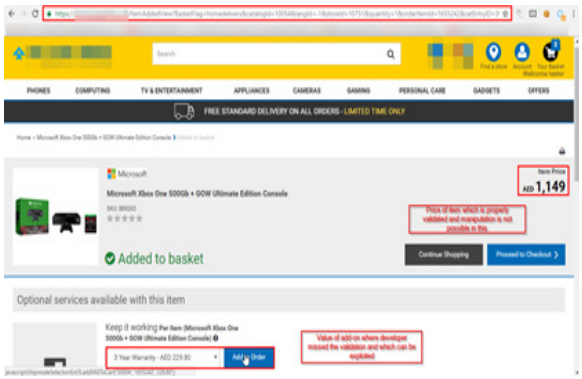
Price Manipulation Attack:

This is a logical vulnerability and affects e-commerce and transaction-based applications the most. It has many sub-categories, but all related to change in prices like,

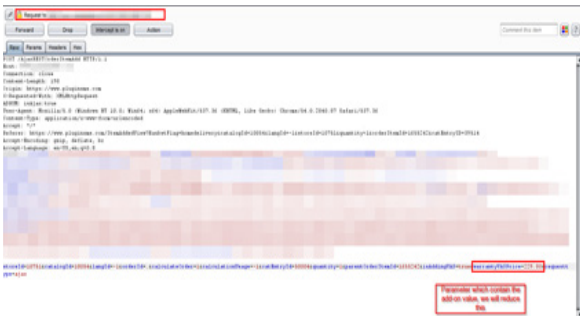
- Buying a product for an amount lesser than the defined amountBuying more quantity of same or different product for lesser amount
- Buying more quantity of the same or different product(s) for a lesser amountUsing some other user's data to purchase any product
- Selling any product for an amount more than the system-defined one
- Using some other users data to purchase any product

Here are a few screenshots that are an example of Price manipulation, where the value of extended warranty is changed:

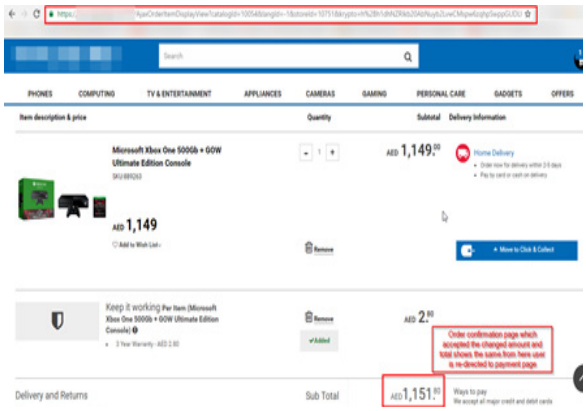
POC 1: We add an item from an e-commerce application to our cart and add an Add-on also, along with the same



POC 2: Now, when we confirm the request and go ahead to the next page, the same request is intercepted using a proxy tool which will show the add-on value in clear text. We will reduce this add-on value here.



POC 3: Post this step, the user will be taken to a confirmation page where the changed amount is accepted by the system and the user is re-directed to the payment page where he/she can enter his/her card details to purchase the product



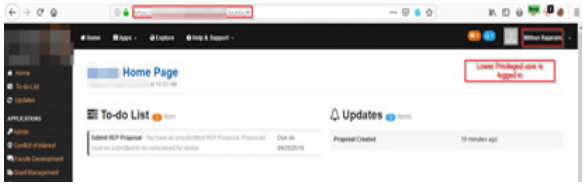
POC 4: In this screenshot, please note that the values are accepted, the payment is confirmed, and the user is provided with a valid Order ID along with the purchased add-on for a lesser amount that was listed on the application. This price is manipulated, and the misconfiguration can be exploited by any user.

Privilege escalation:

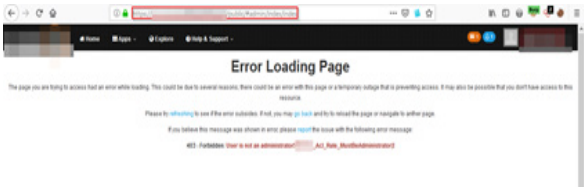
Often, single applications are used by a privileged user and a non-privileged user like admin/guest, admin/normal user, maker/checker etc. In such cases, the server/system identifies every user using some unique parameter/id and the response contains the details related to that user. When any non-privileged user escalates his privilege to view/edit any data in the application which is assigned to a privileged user, it is known as Privilege Escalation.

Here is an example of Privilege Escalation:

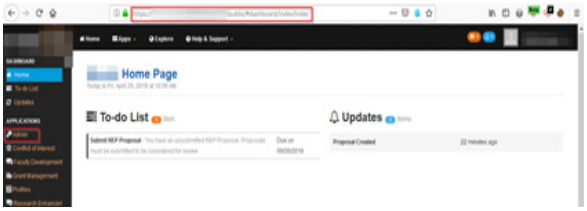
POC 1: In this screenshot, we are logged in as a non- privileged user.



POC2: In this screenshot, the current user does not have access to "Admin" and encounters an error while trying to access the Admin module.



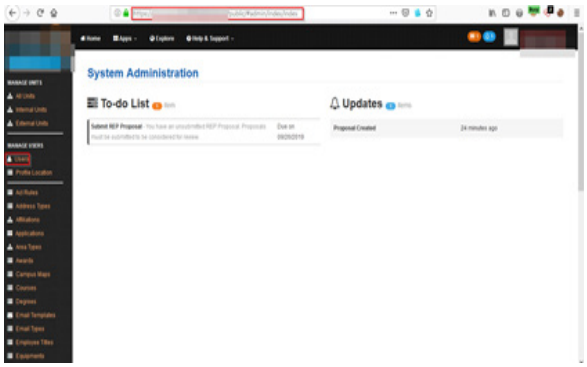
POC3: In this screenshot, we again try to access the "Admin" with the current user and intercept the request with any Proxy tool.



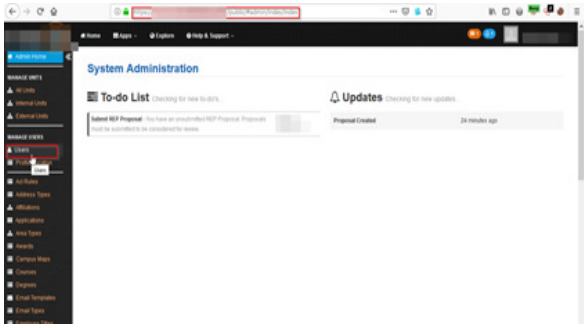
POC4: In this screenshot we get a "403 Forbidden" error in response for this Admin request from the non-privileged user.

POC 5: Observe that in this screenshot we change the response from “403 Forbidden” to “200 OK” and remove the error message from the body and forward the request from a proxy tool to a Browser.

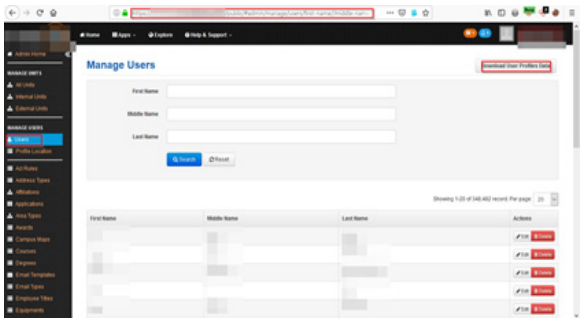
POC6: Now, we can access the Admin module of the application due to the misconfiguration of the Access control list.



POC 7: We now try to access Users under the Admin module.



POC8: While trying to access this module under the Admin tab, we again encounter the 403 Forbidden error which we again change to “200 OK” and remove the rest of the error details from the response body. By doing this, we can now access the User module also, and can edit/delete users from the system. We can automate the process of replacing 403 to 200 also, which will also reduce the time.



If you look at both the examples given above, it requires humans to intervene and understand how the application is behaving and manipulate the logic to exploit the vulnerability. It will be very hard to teach/program automated scanners to find these vulnerabilities.

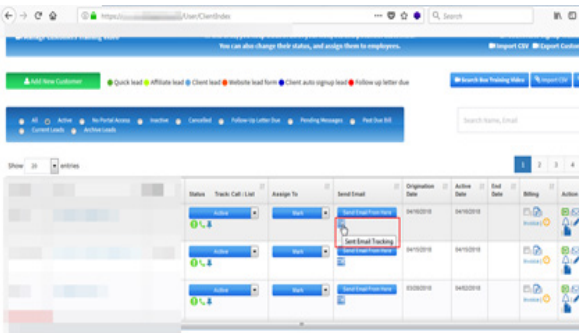
Vulnerabilities found only by combination of human & automated scanner:

These are vulnerabilities, as mentioned earlier, need both human & automated scanners to work in tandem. Example IDOR, DOM based XSS etc.

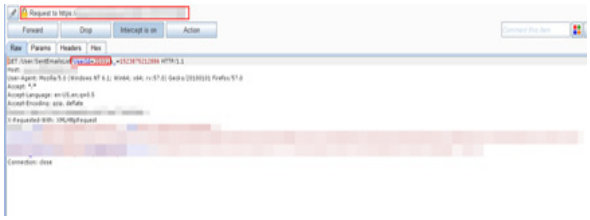
IDOR:

Applications frequently use the actual name or key of an object when generating web pages. Applications do not always verify if the user is authorized for the target object. This results in an insecure direct object reference flaw. Testers can easily manipulate parameter values to detect such flaws and a quick code analysis shows whether authorization is properly verified. Here, automated tools help the Manual Tester to extract valid users from the system.

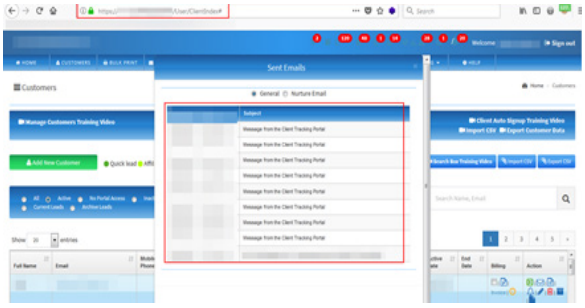
POC 1: In this screenshot, the application has an email tracking provision which provides details of sent emails.



POC 2: Now, we intercept the request of email tracking which contains a Parameter User ID



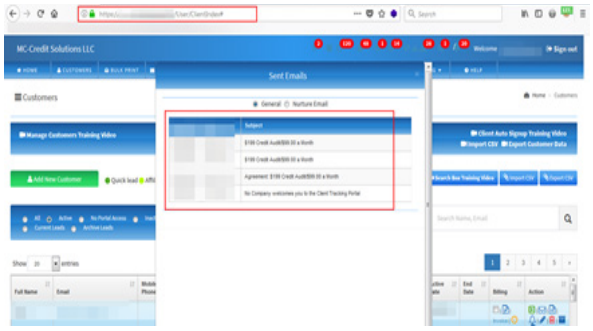
POC 3: The POC shows the email tracking for the provided User ID



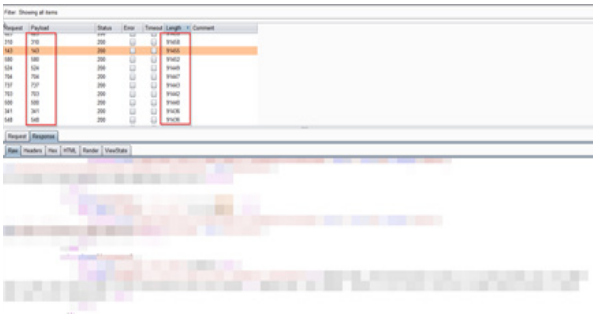
POC 4: Now in the same request, we try to access the details of some other random user IDs by sending User ID in the parameter of the same request.

POC 5: When we changed value to some random User ID, sent emails of provided

User ID is returned.



POC 6 : We can use an automated tool to derive these valid User ids from the system and to fetch further details from this request itself.



If you see in the above example, though a human is required to figure out how to intercept the request and get other user details, the User ID parameter is added in random, but, to identify which User ID is valid and get details of all the users an automated system is required to iterate the same. This is a good example of how automated and manual testing can come together seamlessly.

Compliance Need:

Today, every country & industry has its own data protection laws and regulations and most of them enforce regular pen-testing for web and mobile applications.

For example,

- PCI-DSS mandates ongoing pen-testing for applications whenever a change occurs
- SOX (the Sarbanes–Oxley Act of 2002) and HIPAA (the Health Insurance Portability and Accountability Act) requires application pen-testing to be done. Similar provisions are requested by other data protection standards, like GLBA, FISMA, and OWASP.
- ISO 27001 also has pen testing guidelines.
- GDPR recommends regular pen-testing for secure applications

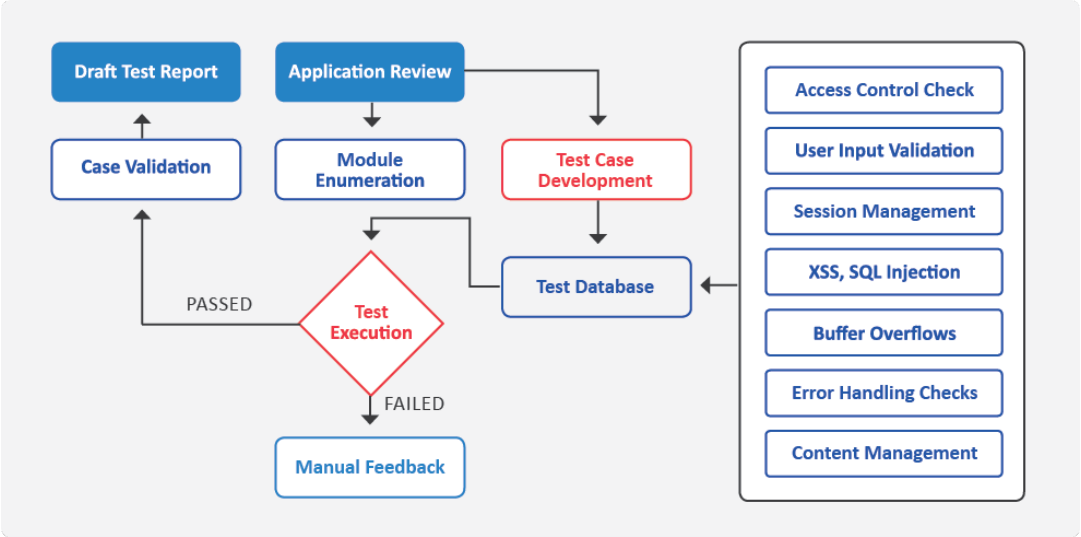
So, no longer is pen-testing just a good application strategy but is a must-have from a compliance perspective.

Pen-Testing Methodology

As a part of manual pen-testing, a security expert would assess the Web Application by performing a range of application vulnerability tests and checks using a combination of manual testing techniques and automated tools testing. At Indusface, we follow the OWASP, OSSTMM & SANS Top-25 guidelines for carrying out Web application vulnerability assessment.

The details are given below:

The Project is initiated with a kick-off meeting with the established point-of-contact of the Customer. The Consultants gather all relevant information with respect to the scope of work before initiating the actual audit. Following is the block representation of the Web/Mobile application testing process that we, at Indusface, take:



Module Enumeration the approach

- Understanding the module operations and its features
- Creating a map of the data and process flow
- Discovering the visible and hidden modules (manually or using tools)
- Understanding data paths (and in some cases, the initialization process as well)

Test Case Development – the approach

- Enumerate the various input, data, exchange fields used in each module
- Identifying the data types accepted by each of these fields
- Enlisting each permutation and combination that could be used in these fields
- Creating a case that could be used to test the application

Case Validation

Each test result will be further validated and verified by completing an attack cycle. This is done to reconfirm the process and to understand the flaws in the application. Validating a case may also be useful to recommend an accurate recommendation procedure.

Test Database – the approach

- Each test case created will be matched against the possible attacks in the list of attacks, in the attacks database
- A permutation of each case and attack is created and added to the test database
- Sample values and or ranges are entered in each of the test cases in the test database
- A test success criterion is also documented in each of the test cases

Application Assessment

In order to conduct a comprehensive Application Assessment, it is vital to have a broad and flexible testing methodology to uncover the most stubborn vulnerabilities.

The Indusface Application Testing methodology leverages dozens of grey-box and black-box tests to understand the workings of the applications better, while identifying vulnerabilities. Combined, these services offer the consultants the framework required to conduct the most thorough assessment possible.

It is important that testers fully understand the unique circumstances around each application, and what the primary concerns are. For example, the Indusface testing methodology examines risk related to each of the following areas (the engagement can be tailored to focus on any particular area):

- Exposure and integrity of confidential information
- Exposure and integrity of confidential employee information
- Denial of service risk to application or application components
- Network infrastructure exposure via application vulnerabilities

Detailed Application Risk Assessment Methodology

The standard Indusface Application Assessment methodology focuses primarily on the following testing classes:

- Architecture
- Business Logic
- Development procedures
- Authentication
- Transmission security
- Session management
- Information or data leakage
- Input validation
- Logic flow and authorization
- Data corruption
- Application deployment

This is a complete and overall examination of the applications deployed and security configuration from perceived threat models. Advice is given on secure deployment methodologies for the application type, based upon market trends, new vulnerability developments and attack methodologies.

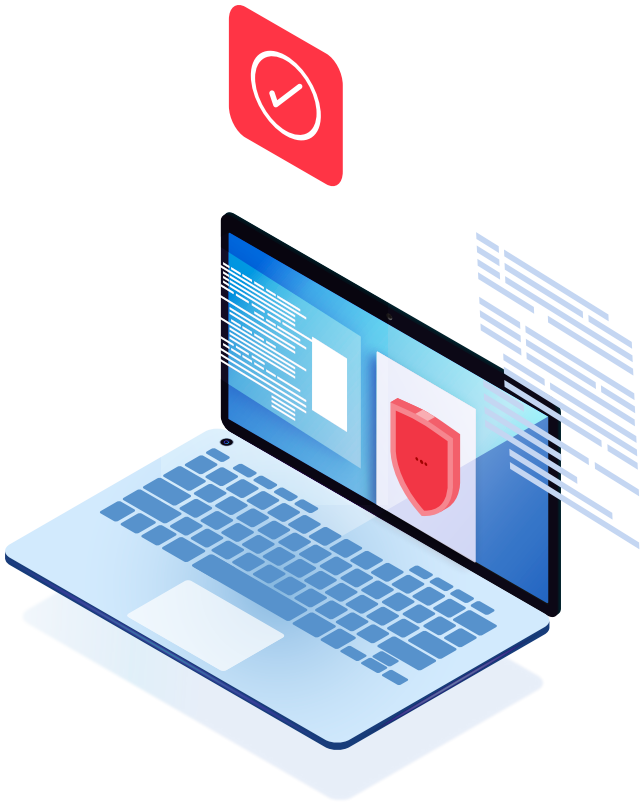
Reporting

At the end of the manual pen-testing process, the customer gets a detailed report of the vulnerabilities found in the application along with the methodology employed and recommendations. Unlike automated scanners, there are no chance of FPs through manual pen-testing. The report will have details on the following:

- Methodology used to carry our audit
- Standards followed
- Tools used
- List of vulnerabilities identified
- Descriptions of vulnerabilities
- Risk rating or severity vulnerabilities
- Proof of Concept
- Recommendations

Revalidation

Most pen-testers, including Indusface, provide revalidation service where customers can ask pen-testers to test again for the vulnerabilities once they are fixed. Generally, it is provided without any additional cost.



Conclusion:

Without pen-testing, no application security strategy is complete. As explained, there are vulnerabilities that cannot be found without manual interventions, and it is a must-have when it comes to companies looking to meet compliance needs. We highly recommended that regular pen-testing is included as a part of the application testing strategy. While choosing pen-testing service providers, do look for their reputation, methodologies employed, and certifications they have. For instance, Indusface pen-testers are Cert-In certified experts. Also, do not discount the importance of automated scanners and periodic daily or weekly scans. They are important in the current agile world. Choose solutions that can provide integrated manual pen-testing & automated scanning solutions like Indusface WAS.

ABOUT INDUSFACE:

Indusface is a SaaS company which secures critical Web applications of 2000+ global customers using it's award winning platform that integrates Web application scanner, Web application firewall, CDN and threat information engine. The company has been mentioned in the Gartner Magic Quadrant and Forrester Tech Now reports, is CERT-In empanelled as a trusted scanning vendor, and has been the recipient of many awards such as the Economic Times Top 25 , Nasscom DSCI Top Security Company of the year Award and is funded by Tata Capital Growth Fund.

Appendix

Application security testing is done to find security vulnerabilities found in applications to ensure they are addressed before hackers find them and exploit them. As part of solid application security strategy, it is important to have a well-planned mechanism to identify the risk posture of the application.

The three major methods for testing application security are:

White Box testing – Testing is done with unlimited access to the application, including source code and documentation. It is comprehensive, but time consuming.

Black Box testing – No information of the application is known before testing. This is closer to reality when one compares with how hackers will behave. Not very time consuming.

Grey Box testing – Testing is done with restricted access to application, with usually no access to source code. This lies somewhere in-between the white box and block box testing methods.

All these methods have their merits in the application security testing market (AST).

Similarly, various techniques can be employed for scanning.

Some of the major options available are:

Static Application Security Testing (SAST)- These are considered as White box testing options, where analysis is done to uncompiled code to find vulnerabilities.

Dynamic Application Security Testing (DAST) – These are considered as Black box or Grey box testing options, where analysis is done without prior knowledge or limited knowledge of the application. Testing is done on the running state of the application.

Interactive Application Security Testing (IAST) - This is a more hybrid approach, and here the combination of SAST and DAST techniques are employed. They try to test if the known vulnerability found in code can actually be exploited at runtime.

Software Composition Analysis (SAS) – This is similar to SAST but not the same. These tools focus on the origin of components and libraries and see if there are any known vulnerabilities found in the component. This is more useful in context of open source components where many applications use diverse components and it is hard to know how they are vulnerable.