EN

The Big Application Security Penetration Testing FAQ for Clients

Everything you should know before, during and after the commissioning of an Application Security Penetration Test

Authors: The pentest team of mgm security partners GmbH

Location
English version: https://www.mgm-sp.com/penetration-test-faq/
German version: https://www.mgm-sp.com/en/penetration-test-faq/

mgm security partners GmbH
Frankfurter Ring 105a | D-80807 Munich, Germany
Phone: +49 89 358680-880
office@mgm-sp.com
http://www.mgm-sp.com

The **penetration test** - or **pentest** for short - is still the most popular way to detect security deficiencies in web applications. The main reason for the prevalence of this test type is that it can be set up without great effort and carried out by an appropriate expert or a company specialising in it.

This simplicity easily disguises the fact that the success of a pentest depends on a number of factors to be taken into account. A pentest is successful if it leads to maximum coverage while minimizing the effort. Question 12 why the task cannot be defined as the identification of *all* vulnerabilities.

If optimal framework conditions are not guaranteed, there is a danger of weighing oneself in false security (question 5): One incorrectly assumes that the application is safe - except for the possibly found weak points - and becomes more negligent in the effort for comprehensive security.

An essential contribution to counteracting this is the knowledge of the pitfalls of a pentest order and their avoidance: from the ability of the pentester to correctly estimate the scope and ensuring smooth operation, to the correct handling of the results of the test.

This FAQ deals with the many questions that arise around the pentesting of web applications - and provides practical answers. Plus a multitude of practical experiences and some deeper insights and views from a long pentester life. It is intended for all those who wish to have their web application checked.

Such a document, which is focused on a specific topic, creates the impression that the pentest is the sole or best means of creating security in web applications. At least the former would clearly be wrong and the latter is also not correct in many cases (see questions 35 and 36). If you want to award a superlative, it is that the pentest is the least dispensable instrument.

Basic

1 Which terms should I know?

From now on we simply call the pentester **tester** and the one that threatens security, the **attacker**. We call the end user of the application, who is often the victim of an attack, **User**. A security problem is uniformly called a **vulnerability**, a discovered vulnerability is called a **finding**. We call a system in which a security incident has occurred **compromised**, and the incident itself **compromise**. We use the term **danger potential** to express the danger of a weak point - synonyms: criticality, threat or risk. In this document, the concept of **risk** is reserved for its technical definition, namely the product of the probability that the undesirable will occur (**probability of occurrence**) and the amount of the resulting (maximum) damage (**amount of damage**). An important concept in terms of risk is that of **risk management**. This term expresses something quite essential, namely that the goal of all security efforts is usually not to achieve absolute security, but rather to strike the right balance between the negative effects of security measures (costs, delays, restrictions, etc.) and the negative effects of insufficient security.

2 What is this, anyway, a vulnerability?

A vulnerability is any characteristic of an application that can be assumed to be unintentional and that it can be abused by a third party - i.e. exploited in a way that causes harm to the application operator. This also includes indirect damage, such as when the direct victims are the users.

3 Why are custom web applications a serious security threat?

Web applications have a number of features that make them a serious security risk.

- Custom web applications are unique. This means that they have not undergone an industrial maturing process such as "off-the-shelf" applications, in which quality enhancing and error eliminating effects usually occur solely through the large number of sold units. The probability of security-relevant errors is therefore very high per se.
- Developing an application with a high quality requirement is significantly more expensive than programming the same application without this requirement. Nevertheless, at first glance, you will not notice any difference between the inexpensive and the expensive variant after completion. With regard to classical software quality metrics, this only becomes visible over time, e.g. in high costs and high susceptibility to errors during further development. With regard to security, the difference is expressed in a high risk of damage through the presence of vulnerabilities. The budget provided for a software project often does not meet these requirements, which is reflected in quality and security deficiencies.
- The inherent complexity of an application is often underestimated. This is probably due to the fact that the inner structure is invisible, the many inner dependencies to the outside do not appear, and there is no "material consumption" in the software production process that would indicate the size of the task. However, complexity always means a high probability of errors.
- The level of knowledge on security issues among many software engineers is still quite low.
- Many application frameworks do not address the topic of security extensively enough. If security measures that could be programmed into the respective framework were also implemented by the framework developers there, the programmer would have far fewer possibilities to implement something insecure at all.

- Very often web applications have a connection or direct access to sensitive data. Even if these are usually well protected on the network level - on another system - they are still accessible via the web application. In the case of security issues in the application, the protective measures at network and system level are generally ineffective.

Measures to ensure application security are therefore particularly important for customer-specific applications.

4 Why should security defects not be treated the same as quality defects?

Security defects, as well as quality defects, are errors in the application. In both cases, development-related measures must be taken to ensure that they do not occur in the first place. And through ongoing and final testing, the aim in both cases is to further minimize the error rate.

Despite this relationship, one makes a - dangerous - mistake if one applies the findings and procedures from quality assurance unchanged to security. This is justified in the following capacity: In the case of quality defects, there is usually - very roughly - a proportional relationship between the size of the defect and the size of the damage. Small error - small damage, big error - big damage. In the case of security problems, on the other hand, the ratio is more in the form of a jumping function: even a small or difficult to find fault can cause very serious damage.

The example of a Shopping application illustrates this: Assume that the shop operator allows changes to be made to the check-out function. Such a change carries the risk, among other things, that the check-out will no longer function due to incorrect programming and that a total loss of turnover will occur, at least for a short time. Nevertheless, the risk of damage associated with the change is very small, as this worst-case scenario can easily be avoided by carrying out appropriate final tests. If more complex error constellations - such as the appearance of a non-Latin character in the customer's name that causes the check-out to crash - are not detected immediately, this can be tolerated, the case is rare and the damage correspondingly small.

Security's different. Let us assume that the change in question consists in the introduction of coupon codes and the corresponding input field in the check-out. Let us proceed from the - quite realistic - assumption of a high deadline pressure during implementation. Then it can easily happen that the responsible developer does not solidly implement the access to the voucher database required to check the validity of the voucher by extending the data model accordingly, but "just quickly" by direct database call. (For insiders: Instead of secure persistence via OR mapper, the extremely insecure method of dynamic SQL call).

A so-called SQL injection vulnerability introduced in this way is extremely dangerous because in the worst case an attacker can gain access to the database or the entire system. A pure quality test does not detect this weak point and it is not noticeable in any other way. The moment it is discovered by an attacker, the maximum damage occurs immediately.

The logic that can be justified in the area of quality assurance "We have tested the most critical error cases intensively and everything is in order. If we have overlooked hidden quality deficiencies for reasons of time or cost, this is not pleasant, but also not so bad" must not be transferred to security. The "bad cases" cannot be specifically addressed and excluded here.

The solution to this problem is not to always perform a complete penetration test, even with small changes. The costs incurred and the time delay do not allow this in practice. Rather, this problem is a further reason for anchoring application security deeply in the development process and for understanding the pentest as just one of several measures. See also questions 35 and 36.

5 Why should I know the phenomenon of False Sense of Security?

Knowing that a risk is not sufficiently covered by security measures (i.e. that there is a security issue) means that appropriate caution should be exercised in everything connected with this security issue, i.e. a higher standard should be set with regard to security. The information officer in a company where communication in the intranet is done via unencrypted HTTP will work towards greater rigour in defining security rules for external parties than the colleague in the company where all applications are accessible exclusively via https. The other way round, the introduction of https in the intranet will be carried out as a measure if it turns out that the required rules cannot be implemented.

Or, related to web applications: The realization that one is badly positioned with regard to application security leads to the decision not to make sensitive applications available over the web for the time being. If, on the other hand, you are on the right track here, for example because you are systematically pentesting, the decision is more in the direction of approving a sensitive application.

A problem arises when there is a gap between what you think you have and what is reality in terms of the level of security available. In the first example: The security rules are loosely laid out because the communication in the intranet is now encrypted - but one overlooks the fact that this is only done for some of the applications. In the second example, the decision to make sensitive data available to the outside world is a positive one, because you have security under control with pentests - and overlook the fact that the pentest s are far too weak for such applications.

In a nutshell: If you find that a security measure cannot be adequately implemented, then you must not rely on it either! Sometimes in such a case it is even better not to implement the security measure at all. This prevents the resulting false sense of security from being used elsewhere to make decisions that endanger security far more than the omitted security measure.

See also question 14.

6 Does the European General Data Protection Regulation (DSGVO/GDPR) introduced in May 2018 affect pentesting?

Yeah, definitely. In addition to the threats that already exist, another threat has been added, namely high fines if personal data is passed on to third parties or to the public via a vulnerability. The application of security measures in general and pentesting in particular have therefore become even more important since the entry into force of the European General Data Protection Regulation.

Penetration Test Basics

7 What is a penetration test?

In a penetration test, the testers assume the role of the attacker. They use all means at his disposal to access the application from *the outside* - in contrast to analysis techniques that start "inside", such as code analysis - in order to uncover vulnerabilities. They also use their extensive knowledge of vulnerabilities and all kinds of tricks to bypass security mechanisms. The result is a report that describes the vulnerabilities in a comprehensible way, assesses them with regard to their potential danger and shows countermeasures.

Because they act as a benign "hacker", the pentester is also called a **white-hat hacker** (as opposed to a malicious **black-hat hacker**) or an **ethical hacker.**

8 And what is an Application Security Penetration Test not?

A penetration test of a web or mobile application is *not* about simulating the attacker scenario as realistically as possible in order to conclude whether an attacker could penetrate the application or not. And as a result define it in the first case as insecure, in the second as secure. Rather, penetration testing is to be understood as a quality assurance measure. As far as possible, all anomalies affecting the security of an application must be identified, evaluated, also regarding the context, and, if they represent an actual risk, remedied. The tester can perform his role most effectively if he receives the best possible support - more on this in question 10.

9 What does a penetration test have to do with finding the needle in the haystack?

A lot! To be exact, the underlying principle is pretty much the same. Only the pentest is about a lot of "needles." Both have this unpleasant characteristic in common: The probability of finding something increases with the duration of the search and with the number of searchers, without ever reaching 100%. Consequently, it is in the nature of things that the crowd-sourced Internet, i.e. the sum of all attackers trying to hack the web application, is superior to any penetration tester with a limited time budget. Both number and search duration are potentially unlimited "on the other side".

10 What can be done about the superiority of the attackers?

Because the superiority of the opposite side in the method of "stochastic" searching is so overwhelming, it is all the more important to oppose something to this. The effective antidote is to provide a maximum of information. The more the tester knows about the internals of the application, the less he has to poke aimlessly in the fog, the more targeted and effective he can use the available time budget.

The information shall include, among other things : Programming technology, frameworks used, descriptions of the software architecture, security concepts, API descriptions and, depending on the type of application, further information to be individually coordinated.

See also question 15

11 What's a retest?

If the pentest has revealed vulnerabilities worth fixing, they should be re-tested after they have been corrected to the best of the availabe knowledge. It will be checked whether they have actually been completely remedied by the action taken. Strictly speaking, a complete pentest would have to be carried out again, because a change at one point can lead to security problems at a completely different point. For reasons of cost and time, however, a retest is usually carried out in the following manner: It checks whether the vulnerability found no longer occurs at the *previously reported point of occurrence.* Thus, it does not provide the statement that the vulnerability has been fundamentally remedied, i.e. that it is *not* present *in any other place.*

This situation becomes even more important in connection with the following paragraph.

12 Can you find all the vulnerabilties with a pentest?

No, as a rule the conclusion cannot be drawn that after a that the application does not contain any other weaknesses apart from those detected. This is mainly due to the needle-in-the-haystack nature of pentests (question 9). The following rule can be derived from this:

Rule #1: Do not rely on the pentest alone, but take additional measures to ensure the security of your application.

See also question 36.

13 Does a pentest always find all places where a certain vulnerability occurs?

Let's take a normal web application and the **SQL injection vulnerability** as an example. The application has a number of forms and each form contains a number of fields. SQL injection occurs preferably when using the data entered in the fields. Among other things, the pentester successively checks the forms and fields for SQL injection. If they encounter this vulnerability, they describe it in the report. They then check other fields if necessary, but now only randomly. So not *every* location where SQL injection is present will appear in the report, as the time spent on it can be much better used to search for other vulnerabilities.

If a vulnerability scanner is used in the search for certain types of vulnerabilities (SQL injection is one of the vulnerability categories where scan tools often perform well), the list of listed vulnerabilities is probably longer, but here too completeness is not guaranteed. Often a scanner does not detect all forms and fields or encounters other problems that stand in the way of automated testing.

The result is that the pentest report does not always contain all the points of occurrence of a particular vulnerability, but may contain only one or a few selected ones. This in turn has a significant impact on the task of remedying the vulnerability. See question 41.

Note: A simple retest (question 11) does not reveal such errors anymore!

14 What about low-cost pentests (quick test, initial test, low-budget test, low-hanging fruits test, purely tool-based test)?

As a general rule, any type of test is better than no test at all!

But only on one condition: You are not subject to False Sense of Security (question 5)!

A quick test for applications with normal or high protection requirements (questions 29 and 32) is generally not a sufficient security measure, but it does serve well

- to achieve a limited risk reduction that is difficult to specify in scope.
- to give the client an initial indication of the level of securit.
- to assess the need for and nature of further action.

Properties of pentests

15 Black box, grey box, white box?

Depending on the amount of information with which the pentester starts their work, a pentest is called a black box, gray box or white box pentest. In a **black box pentest,** the tester does not receive any additional information, i.e. they find themself in the same situation as a third party. As we learned in question 10 this variant is generally not suitable for testing web applications.

The **white box pentest is** the opposite to be aimed for - full information. This goes all the way to the inspection of the source code of the application. Since this is usually associated with a much higher effort, the compromise of the **grey box pentest is** used much more frequently - maximum information, but no provision of the source code.

The white box test with code insight can be distinguished from the **selective code review** or the comprehensive **automatic static code analysis, which is a** useful additional measure to the pentest.

Rule #2: The pentester shall be equipped with a maximum of information; a grey box or white box penetration test should be performed.

16 How does a pentest work?

1. Offer Preparation

The foundation stone for the successful execution of a pentest is already laid during the preparation of the offer. The better the service provider is able to estimate the extent and overall scope, the smoother it will run. See question 32 on the determination of the need for protection, question 33 on the complexity of an application and question 34 on the determination of the test effort.

2. Test Preparation

The same applies to the quality of the preparation. All preparatory measures agreed with the service provider should be completed by the agreed start date. This is not without effort for the client and must be planned accordingly. The experiences taken up in question 38 show that many things can go wrong.

3. Test Execution

The penetration tester assumes the role of an external attacker. They use all their experience and the information provided to identify vulnerabilities in the application to be checked, supported by professional tools. Report generation usually goes hand in hand with testing.

Again, the assistance of the contracting authority is necessary to avoid the problems mentioned in question 38

4. Report Delivery

Due to the problem raised in question 41 is recommended not to complete the pentest upon reception of the report, but to consult with the service provider after thorough consideration of the reported vulnerabilities. See question 40.

5. Retest

Once the vulnerabilities have been remedied, it is recommended that a retest be carried out, which, in the simplest, most common case, will re-examine those parts of the application where vulnerabilities have previously been found. What has to be considered is discussed in question 11

17 What is tested at all in an Application Pentest?

In short, most of what can go wrong with the implementation of the application and has a negative impact on security. This includes above all implementation errors, but also faulty configuration and logical problems. The following categorization simplifies the overview:

1 **System level**: The software environment in which the web application is embedded is examined here. That menas the secure configuration of web server, CMS and other system software that is

needed for the application to run. In certain cases this area is excluded, see question 24.

2 **Technology**: Are basic security properties of the application guaranteed by the (correct) application of safeguarding technologies? This includes the use of secure encryption methods, secure password storage, implementation of a security architecture, etc.

3 **Implementation:** This is the large area in which it is checked whether the developer has made security-relevant implementation errors: SQL injection, XSS, CSRF, etc. - the list of possible vulnerabilities is long.

4 **Logic**: If a function has been implemented technically correct, but the internal processes have overlooked possible misuse, this is referred to as logical vulnerabilities. For example: password cracking is not prevented, the registration function can be misused to send emails on a massive scale, or the password reset procedure can result in cases where a third party is able to invade the account.

5 **Semantics**: For example, when an attacker plans a phishing attack, he chooses an application that is expected to be particularly vulnerable to its users. This is the case, for example, when users are used to receiving regular emails from a company containing links to the application or are even prompted to click on embedded links. Such "preconditioned" users are far more likely to click on a fraudulent link in a fake mail than users who are used to their company sending mails without links. Properties that favour these or similar attacks are examined here.

18 How are vulnerabilities found evaluated?

There is no standardised evaluation system for the danger potential of vulnerabilities. Nevertheless, most vendors assess vulnerabilities very similarly, according to a scheme in which the risk potential is divided into the categories **low**, **medium**, **high** and **critical.**

The categories can be roughly interpreted as follows:

Critical: This vulnerability represents an unacceptable risk. The application must not be used productively; if it has already been used, it must be deactivated immediately.

High: This vulnerability must be addressed immediately, possibly by an emergency patch. Further measures for risk avoidance up to the remediation are to be considered.

medium: This vulnerability must be addressed, even if it involves (moderate) additional costs or other (moderate) disadvantages.

Low: The correction can be included in regular release planning.

Sometimes there is an additional parameter (**probability of occurrence** or **complexity**) that tells us how difficult it is to find and exploit the vulnerability - in other words, how likely it is that an exploitation will actually occur via the vulnerability in question.

The CVSS methodology (Common Vulnerability Scoring System) presented in question 19 whose ratings are given on a scale from 0 to 10, differentiates even further. Although its practicability in the area of web applications is rather limited, it is being used more and more.

The pentester mainly evaluates a type of vulnerability on the basis of general "danger benchmarks "; application-specific threats and abuse scenarios can only be taken into account to a very limited extent. It is therefore highly advisable to add a self-assessment to the report once it has been completed: Here It is examined whether other ratings might result from the inclusion of factors which the pentester is not aware of.

19 What is the CVSS score and why should I deal with it?

"The Common Vulnerability Scoring System (CVSS) is an industry standard for assessing the severity of potential or actual vulnerabilities in computer systems" (Source: Wikipedia). The design had "off the shelf" applications in mind, but not the specific requirements of custom web applications, which are unique in nature. Nevertheless, the CVSS has become more and more popular in recent years for the evaluation of vulnerabilities in customer-specific web applications due to a lack of better alternatives.

In the CVSS, the total score of a vulnerability results from the combination of a series of isolated evaluations of individual aspects (metrics), taking into account the weightings stored in the calculation formula.

Strengths:

- Splitting into metrics to be evaluated individually simplifies the evaluation for the pentester
- Very good comparability both at the level of the total score and at the level of the metrics
- Easy traceability of the overall score by means of "drill-down" to the individual evaluations

Weaknesses related to web applications:

- Metrics are included that are unimportant or less important for web applications (e.g. attack vector metrics, temporal score, remediation level metrics are defined in a way that is not meaningful for custom web applications).
- Small differences in the input parameters can lead to large differences in the overall score under unfavorable circumstances.

It must be said, for example, that the increasing popularity in the area of application security is less due to the particularly good suitability than to the lack of alternatives.

The following experiences should be taken into account when using CVSS:

- If the comparability of the evaluation across many applications and/or different pentesters is important - e.g. for large companies with many applications or for integration into DevOps processes - the CVSS should be considered as an evaluation scheme.
- Companies that occasionally carry out pentests or where comparability is not important for other reasons should rather use the criterion of simplicity. The CVSS is then not the first choice.

- The CVSS has a counterproductive effect when the rules for the consequences to be drawn from a vulnerability are too rigidly laid down. So if, as part of the deployment process, a score that exceeds a specified threshold automatically results in the stop of a go-live - possibly imminent and very important for the business. The process should therefore include a feedback loop to ensure that vulnerability assessments with serious consequences can be subject to a detailed assessment and that the CVSS assessment can be corrected manually.

20 How reliable is the rating and how do I deal with it?

It is hardly possible to evaluate a vulnerability in any form and with regard to all requirements in an objective and resilient way. The technical connections are too complex and the scope for interpretation too wide. The assessment is based on the information available to the pentester, which is usually very limited. In case of doubt, the pentester will lean towards more security and rate the risk higher.

In particular, where the removal of a vulnerability involves high direct costs (the removal itself) or high indirect costs (e.g. loss of revenue due to delay or interruption), a more differentiated risk assessment should be carried out.

This uncertainty in the rating system is also the reason why it is not necessarily a good idea to include in the release process irrevocable rules that prevent an application from not being released as long as it still contains category x vulnerabilities. Instead, such a case should result in a detailed risk assessment on the basis of which the decision 'Go' or 'No-Go' is taken.

21 Can pentests be performed automatically?

So-called DAST tools (Dynamic Application Security Testing) or application vulnerability scanners promise to be able to scan a web application comprehensively and automatically for vulnerabilities. However, the great variety of programming technologies, the high complexity of the application logic and the lack of standardization in the dialog processes between client and server make such automation an unsolved problem to this day. It has to be said, however, that DAST tools are justified in special cases (such as when testing specific vulnerabilities or in build environments with high test repetition rates), but are still unsuitable as the sole testing tool. A serious pentester uses such tools only to support manual testing and as a supplement to the many small tools in his toolbox.

Pentests that are completely or primarily performed by a tool are generally not suitable as a replacement for a manually performed pentest.

22 Should the pentest take place on the test system or on the production system?

There is no clear answer to this question. Even if the risk of damage or side effects caused by the (manually controlled) pentest is very low, this scenario should be avoided if possible. In an ideal world, the pentest should take place in a staging or pre-production environment that is as similar as possible to the live environment.

Frequently, however, further limiting conditions stand in the way of this requirement and the pentest must be carried out on a less ideal system, e.g. also in productive operation.

The existence of personal data would also argue against carrying out the test on the productive system. Legally, pentesting represents **data processing on behalf of a third party** and is therefore subject to the rules of the GDPR. See also question 14.

Rule #3: The pentest should be performed on the system which, taking into account all the relevant conditions, proves to be the most appropriate. Note the possible side effects when it is the productive system, especially as far as data protection is concerned.

23 Do you want to test with or without Web Application Firewall (WAF) and Intrusion Detection System (IDS/IPS)?

This question arises for companies that use perimeter-based protection that filters at the HTTP protocol level, i.e. protects against improper input and access to web applications. The clear answer is:

Rule #4: A perimeter protection (Web Application Firewall, IDS/IPS) must be deactivated during the pentest.

Otherwise it will obscure the vulnerabilities in the application. We remember question 8: An Application Security Pentest is a quality assurance measure. More importantly, if a filter rule is subsequently changed - for example, because it causes problems in another application - the protection of all applications that depend on that rule is suddenly at stake.

Exception: If the task is not the pentest of the web application, but if it is necessary to check the security of the entire system or the effectiveness of the set filter rules.

In addition to the filter function, some web application firewalls offer further protection functions, such as the prevention of password cracking attacks or the systematic tapping of information through mass access. If the application uses these protective functions, the Pentest should be divided into two parts. After the application has been extensively tested without perimeter protection, the effectiveness of the other protective functions should be tested with activated filters. Careful coordination in advance is essential to ensure that there are no gaps here.

24 Should the host also be tested?

In large enterprises with many web applications, these are usually embedded in a professionally operated hosting environment. The security of servers and infrastructure is guaranteed centrally, so that a penetration test can concentrate exclusively on the web application.

If this is not the case, it is absolutely necessary to clarify with the service provider whether the pentest should be extended to the host. A host test examines the following components and answers the following questions:

- Is the server operating system hardened?
- Are there no redundant services and open ports?
- Is the web server securely configured?
- Are the operating system and all services visible to the outside world up to the latest version for which no vulnerabilities are publicly known?

- Are used standard software systems like a CMS (e.g. Wordpress, Drupal, Joomla) or shopping software (e.g. Magento, Shopify) used securely and are the security settings set correctly?

In contrast to the application level, automated tools such as *nessus* or *OpenVAS* perform well in this area (the so-called "system level" according to the categories in question 17).

If hosting takes place in the cloud, the same rules apply: If the cloud provider is responsible for the underlying infrastructure, the review may be limited to the web application. If the servers are managed by the customer, a test of the host system is advisable. The specifications of the cloud providers for pentests must be observed.

25 What to consider when pentesting web applications in the cloud or hosting providers?

Some cloud or hosting providers prohibit (unannounced) pentesting. The regulations at your own provider must therefore be clarified in good time in advance. During preparation, the pentest service provider should be able to support the client or even completely take care of these formalities.

26 At what stage of development should the pentest be carried out?

A pentest, which has the task of the final security assurance of a release, must be performed on the final state of this release before the go-live. Subsequent changes should (actually: must!) no longer take place, this would reduce the confidence in the results of the pentest. Changes in regard to the remediation of found vulnerabilities should be verified by a retest (see question 11).

In addition, a (comprehensive or partial) pentest can take place at any time, provided that a runnable state of development exists. Such preliminary pentests do not replace the final pentest.

27 How often should a pentest take place?

In theory, after every change, because with every change security-relevant errors can creep in, even those with massive effects.

In many cases, however, the practice is different: If an application has already been subjected to a pentest, "felt" small changes are usually released without renewed security checks. In the case of more extensive changes, there is often a desire to test only the changed or newly added functional areas. Only in the case of major changes and when the pentest was a long time ago does it become clear that a new complete pentest must be carried out.

If the quality criterion of security has been integrated into the development process from the beginning, i.e. the recommendations addressed in question 36 have been followed, this pragmatic approach may be sufficient. Without this prerequisite, however, this frequently observed procedure must be classified as risky.

In any case, the following applies: At the latest when major changes are made to the architecture and functionality, the original penetration test loses its validity and a new comprehensive test is necessary!

28 How do you test agile development and continuous integration?

Modern agile development generally requires a different approach to pentesting. The motto here is to test individual, already completed components of the application early in the development cycle in order to identify possible vulnerabilities as early as possible and to avoid similar problems during further development. In subsequent releases, individual completed features or a sum of changes can be tested for security (feature or delta test). Although the vulnerability coverage of automated security tests is still very limited, as much automation as possible should be built into the process chain to detect vulnerabilities.

With the agile development process, it is particularly important to free oneself from dependence on the pentest, not least because of the poor automatability of pentests and the short release cycles. See also question 36.

Scope and price of a pentest

29 What is the protection requirement?

It should come as no surprise to learn that a bank should spend more time testing its online banking system than a provider of an app for searching for a parking space. But what criteria do you use to orientate yourself? This is where the need for protection comes in. The higher the damage arising in the event of abuse and the more likely an abuse is, the greater the need for protection. An application with a high need for protection requires a greater depth and scope of testing - in short: more intensive searching - than an application with a low need for protection. More frequent testing and the use of other analytical techniques, in particular code analysis (see question 36), are also appropriate means for applications with high protection requirements.

The determination of the need for protection is often not trivial. In addition to the effects on the business processes, it is primarily relevant how valuable the processed data is for protection. It should also not be underestimated that even if the application and data are considered to be worthy of limited protection, the loss of trust (damage to image) caused by a security incident can still be considerable and in the worst case could endanger the business model.

30 ... and why does the need for protection matter?

Because of the cost. Since applications cannot be extensively tested by a machine, test effort and price are closely related. It is therefore necessary to determine the degree of effort required to achieve an acceptable risk reduction at minimum cost. See the concept of risk management in question 1.

31 In which unit is the need for protection measured?

The metric for the classification of protection needs is not clearly defined. It usually consists of three to five stages. For the purpose of pentesting, the following 3-step classification has proved practicable: **standard**, **high** and **very high**. Strictly speaking, there is a fourth level in the game, the one with no need for protection at all. Since a pentest is not required for such applications, however, we do not need it here.

The need for protection of an application is rated as **high** if a compromise leads to considerable and usually unjustifiable damage. If the damage even has existentially threatening dimensions, the need for protection would be **very high**. All other applications worthy of protection are assignt to the level **standard**.

32 ... and how do I determine the protection needs of my application?

The topic of determining the need for protection goes far beyond IT security. It covers all business risks and is a science in its own right that has produced sophisticated processes. Large companies are generally well positioned and have a comprehensive risk management system in which the need for protection of applications and data is embedded. The following is therefore intended for those who do not possess such a system and wish to determine the protection requirements of the application to be tested pragmatically and by simple means.

In order to arrive at the assessment of the need for protection, we must assume the maximum damage scenario. Imagine the following: An attacker has entered your application or website and gained full access. He is able to read any data or manipulate it in any way. It can modify the content of the site, store false information that is obvious or difficult to detect, or insert malicious code (Trojan horses, viruses, ransomware, etc.) with the aim of deceiving the users who trust your site. He can nest himself unrecognized in the server and from there wreak further havoc. Think of the maximum damage that can result from compromising the application.

Would the impact of such an incident on your business have serious consequences? Then the level **high** is appropriate. If the effects are even to be regarded as business endangering, the level must be **very high**. In all other cases, the need for protection would be considered **standard**.

In particular, the nature of the data shall be taken into account in the evaluation. If personal data such as name, email address or address data are involved, the level **standard** is no longer appropriate, but **high** is the right choice. This circumstance arises solely from the legal consequences of (in the worst case, negligent) handling of personal data. If, in addition, highly sensitive data such as credit card, payment, account information, health data, other confidential personal data, business secrets or sensitive financial data of the company are accessible for the application, the classification **very high** is appropriate.

One particular potential danger that needs to be considered is the impact on the company's reputation - **damage to its image**. This is often very high, regardless of the actual, possibly rather small damage. Simplifying or improper reporting in the press or the targeted exaggeration by competitors or groups that are not well-disposed towards the company can considerably increase the damage to the company's image.

Rule #5: If not already present, a simple protection requirement determination shall be carried out according to the specified procedure. If in doubt, it is better to play it safe and choose the next higher level.

33 What else has an impact on the price?

The second relevant factor in determining the test effort is the size of the application: The number of forms and fields, the size of dialog lines, the number of functions, number and type of roles and rights, etc. Since, in addition to these parameters, there are usually other parameters to consider - such as the complexity of dialog processes and use cases - the **complexity** of the application is usually used instead of size.

A third factor is often added, namely when the view of the externally visible and accessible part of the application is not sufficient to test the security of a system. For example, the case that files are received by the application via upload and processed internally without the results being returned to the web application. It must therefore be clarified whether such areas are to be tested as well, i.e. the testscope must be defined. This must be documented exactly in the final report, especially if aspects have been excluded, so that it becomes clear that the pentest result only provides a partial statement.

The estimated testing effort, and thus the price, is thus roughly derived from the product of the need for protection and the complexity of the application, taking into account the test scope.

There is, by the way, a natural limit for the applicability of the penetration test. At a certain degree of complexity, the penetration test can no longer be applied comprehensively. Then it can be considered at most as part of a testing concept, in addition to means such as static code analysis (question 36).

34 How is the final test effort determined?

Parameter 1, the protection requirement, is provided by the operator of the application and meaningfully questioned by the service provider. Parameter 2, complexity, is to be determined as an essential input for the preparation of the offer by the service provider. Various methods are available for this purpose, which are listed here with their advantages and disadvantages:

#	proceedings	advantages	drawbacks
1	**self-inspection** If the application already exists and all areas to be tested are accessible from the outside, the pentest service provider is provided with one or - in the case of different roles - several test users. The	• Low cost for the client • High accuracy	• Only possible if application is accessible to the tester (from outside).

	provider clicks through the application and gets an overview on his own.		
2	**walk-through** A web meeting is set up and the client guides the pentest service provider through the application.	• High accuracy • Enables queries and facilitates coordination	• Associated with (moderate) expenditure for the client
3	**On document basis** In particular, if the application is not yet ready, there is the possibility of assessment based on specifications, mockups, functional descriptions.	• Low effort	• Low accuracy
4	**Example** If the application is a "typical" representative of an application area (e.g. shop, employee portal, discussion forum, etc.), the assessment can be made on the basis of the application area name plus answers to questions.	• Minimal effort for both sides • Fast	• prone to errors due to the potential for misunderstandings in the communication between client and service provider

No matter which procedure is chosen, the most complete possible transmission of the information referred to in question 10 contributes to a smooth process and an accurate assessment.

Rule #6: Support the bidding process by providing comprehensive and reliable information.

35 Is the pentest sufficient as a means of establishing security?

The answer to this question can best be illustrated with this practical experience: For applications that were developed without considering the security quality feature - for example, because one assumes that one could subsequently set up the security on the basis of the pentest results - the pentest delivers, as expected, a considerable number of vulnerabilities. But the really worrying thing is this: In such applications, the retest after the vulnerabilities have been fixed shows disproportionately often that the reported vulnerabilities have not been properly remediated. And later pentests on subsequent releases of the same application usually show the recurrence of previously reported vulnerabilities, only perhaps in other places or in a slightly different garb.

Bottom line: If an attempt is made to make an application secure - quasi in the aftermath of development - by means of a pentest and the subsequent elimination of the problems it identifies, it will be difficult to ever get its security under control properly. Unplanned efforts and associated additional costs and delays will be incurred throughout the life cycle of the application. Instead, security as a quality characteristic must be anchored in the development process (Question 36). It applies:

Rule #7: The role of the pentest is not to establish the security of an application. It is used to determine whether a sufficient level of security is provided.

36 So what else should be done for security?

This question clearly goes beyond a Pentest FAQ, so the most important measures are only briefly listed here. The website of the "Open Web Application Security Project" (OWASP, www.owasp.org) is a starting point for comprehensive information on application security.

If one is still at the beginning of a new software project, one is in the fortunate position to be able to carry out the following extremely effective measures:

- Developer training: Seminars on the introduction to Web Application Security and Secure Coding Training should be a natural basis for all team members.
- Security Architecture Workshop: The first considerations regarding the system and software architecture of the application must be compared with an application security specialist with regard to the consideration of security aspects.
- Building Security In: Include security as a quality feature in the development process right from the start.

- Make code inspections for security a measure accompanying development.
- Especially in agile process models and continuous integration environments, the build chain can be extended with tools for automated security testing as far as the state of the art permits.

For existing applications:

- Architecture analysis: Analysis of the embedding of the application in the surrounding system landscape, communication relationships, trust transitions, etc.
- Static code analysis in the form of manual code inspection of particularly security-relevant code areas or comprehensive code analysis using commercial SAST tools.

37 Is it enough to only test Internet applications, or should Intranet applications also be tested?

The answer from the mouth of the security representative would be that of course all intranet applications in need of protection should also be tested. The pragmatist, on the other hand, is of the opinion that this should be made dependent on the trust status of the respective environment and that of course the Internet has a lower trust status or no trust status at all. The classification is therefore subject to security principles of a higher level, a generally valid answer cannot be given.

But what needs to be borne in mind is this: Many applications that once started as pure intranet applications - and without special security requirements - eventually expand from their own employees to external partners and are then opened for access over VPN and finally, for further simplification, also for access over the Internet. If this is done without the introduction of rigid security and testing measures, high risks arise.

38 How can I contribute to maximizing the quality of the pentest?

The quality of a pentest does not depend solely on the quality of the pentester. Good organisation and communication as well as a smooth process also have a great influence. The client can make a significant contribution to maximising quality by avoiding these errors:

- Delayed or incomplete provision of the application on the agreed start date
- Test users who do not function or do not have the required rights
- Access problems due to obstructive firewall settings (check beforehand whether the access also works *from outside*!)
- Failures of the application or individual parts during test execution
- Tests or deployments of the application running in parallel that lead to data changes or changes in the scope of functions. (These pull the ground away from the tester, because his reference points are no longer correct.)
- Non-availability of a contact who is able to provide information.
- Missing or incomplete test or example data in the application.
- Missing or incomplete description of API calls

Rule #8: By creating the best possible framework conditions, the client can make a significant contribution to maximizing the percentage of vulnerabilities identified.

Dealing with the results report

39 How much time must be planned for the elimination of vulnerabilities?

A widespread error is that too little time is scheduled between the end of the pentest and the go-live date. If vulnerabilities are found, there is not enough time left for a thorough repair and the application can either be patched "with a hot needle" (see question 41) or made available while accepting vulnerabilities that have not yet been repaired. Critical vulnerabilities can then throw the entire planning overboard at the last minute.

Rule #9: Assume that a pentest will find vulnerabilities and therefore plan enough time to fix them.

40 How do I deal with the pentest report?

It is important that the vulnerabilities found are well understood both in terms of their impact and their cause. If this is not the case, the pentester should be contacted for support. (The pentest service provider should also offer this service).

Further tips for a pragmatic handling of the results:

- As a general rule, the evaluations should be repeated, as the context of a vulnerability in the actual environment is sometimes difficult to assess from the external pentester's point of view (see questions 18 and 19).
- Take critical vulnerabilities very seriously and act immediately, see question 18!

- Prioritise the vulnerabilities classified from high to low according to decreasing risk potential, i.e. use the effort - i.e. the budget available to remedy them and the time available - to a large extent for the serious vulnerabilities and with correspondingly decreasing priority for the remaining vulnerabilities.
- It should not be overlooked that a pentest sometimes only finds evidence of a vulnerability but cannot detect it. Often, an assessment with a lower danger potential is then made in the report, supplemented by a corresponding remark. If a finding has such comments, it should definitely be followed up to rule out that the risk has been underestimated.

The other side of the coin should also be considered: Security is not an end in itself! The advantage of higher security often brings with it the acceptance of disadvantages. Before one falls into actionism, a vulnerability, the remedying of which would have side effects, should be subjected to a sober cost-benefit analysis.

41 What do I need to consider when fixing a vulnerability?

A quick fix that causes the reported vulnerability to disappear is generally not sufficient. As shown in question 13 aim of a pentest is not to find all places where a vulnerability occurs, but only to prove the existence of the vulnerability and to prove it in the report by means of a concrete occurrence. It is therefore necessary to solve the problem fundamentally and permanently and to proceed according to this rule:

Rule #10: You must look for the root of the problem and fix the vulnerability there.

If only the symptom of the vulnerability is corrected at the site of occurrence, the same vulnerability may remain elsewhere. Or you yourself or your colleagues developing the next release rebuild the same vulnerability.

Last but not least

42 OWASP Top 10, OWASP Testing Guide, ASVS, CWE - which Pentest is right for me?

There are various collections of vulnerabilities that can be used as orientation for pentests of web applications. These in particular should be mentioned:

The **OWASP Top 10**[1] are the compilation of the "10 most common risks for web applications". They have been intended as a contribution to raising awareness in the field of web applications. The rapidly gaining popularity has prompted security vendors to advertise their tools and services - often erroneously - with the property "covers the OWASP Top 10". As a result, the OWASP Top 10 is now widely regarded as the standard for what a pentest or security analysis should check. They can't live up to that:

- Most of the risks included are quite general and exemplary and thus leave much room for interpretation with regard to concrete tests. This does not make them particularly useful as a contractual basis.
- Only the technical areas are covered, but not the business logic (related to the classification in question 17: Levels 4 and 5 are not considered).
- Important technical vulnerabilities (for example, Cross-Site Request Forgery (CSRF)) are not included.
- In general, one of the rules (A10, concerning logging) cannot be checked at all with the penetration test method.

In a nutshell: The customer's request "please provide a pentest according to OWASP Top 10" or the provider’s offer "we test according to OWASP Top 10" are both very spongy and generally inadequate,

[1] https://www.owasp.org/index.php/OWASP_Top_Ten_Project

even in the case of a wider design. It should be specified more precisely and supplemented by additional tests.

The **OWASP Testing Guide (OTG)**[2] is a comprehensive collection of vulnerabilities and instructions on how to locate them. The OTG also does not have the aim of being used as a catalogue of vulnerabilities. Rather it addresses itself with the description of penetest techniques primarily to (prospective) pentesters. Due to the much higher degree of completeness and detail, it is nevertheless much better suited as a basis for an assignment than the OWASP Top 10. Its biggest weakness is the fact that it is already very old in the current version 4 of 2014 and therefore does not contain newer vulnerabilities.

The **OWASP Application Security Verification Standard (ASVS)**[3] claims to be a comprehensive framework of **security** requirements and measures (**Security Controls**). It covers functional and non-functional requirements and covers not only testing, but also the design and development of applications. Since its first release in 2009, the framework has been significantly revised several times, taking into account user experience, and is now available in a mature version 4.

ASVS defines three levels of severity of the checks. Level 1 corresponds approximately to the protection requirement *standard*, level 2 to the protection requirement *high* and level 3 to the protection requirement *very high* according to the nomenclature dealt with in question 31

Only level 1 can be covered by pentests alone; the other two levels contain tests that cannot be performed by looking at them "from the outside" or even explicitly require further test procedures.

The specification of a "pentest according to ASVS 4 Level 1" is therefore not sufficient for many applications. And the specification "Pentest according to ASVS Level 2" is contradictory, because Level 2 cannot be covered by pentests alone. Also the specification regarding the latter, to only carry out the tests from level 2 that can be carried

[2] https://www.owasp.org/index.php/OWASP_Testing_Project

[3] https://www.owasp.org/index.php/OWASP_Application_Security_Verification_Standard_Project

out using the pentest, is not meaningful, since these tests are not clearly identifiable because they are tailored to test methods other than pentesting.

The **Common Weakness Enumeration (CWE)**[4] is a continuously maintained list of application security vulnerabilities. Their claim is to be the benchmark for the coverage of security tools and baseline for vulnerability assessment and security measures. The list is very detailed, currently over 800 entries. However, the vulnerability descriptions do not provide any information as to whether a vulnerability can be tested with a pentest. Therefore, it is also not well suited for sharply defining the scope of a pentest. Nevertheless, the requirement for the pentester to cover the CWE as far as possible is the most accurate of all the methods mentioned here.

Ultimately, it must therefore be stated that there is no (fully applicable) penetration test standard to which reference can be made in order to compare offers and commission penetration tests. Unless company guidelines explicitly stipulate otherwise, trust in the pentest service provider should be the deciding factor for the test procedure and scope after extensive consultation.

43 How do I find the right pentester for my application?

There is no standard suitable for all situations and requirements that makes it easy to formulate requirements for the penetration tester (see question 42). In order to find the right approach for your own needs, you should seek advice from the appropriate experts.

Tips for choosing a reliable supplier:

- Ask how long the vendor has been running Application Security Pentests / how many applications they have tested / how much experience the have with companies and applications of your size / for which (similar) companies they have already performed penetration tests.

[4] https://cwe.mitre.org

- Ask which test standards the provider uses as a basis and how they ensuresthat the scope of testing is continuously adapted to technical development.
- Demand that the test be carried out by the person or persons with the appropriate experience.
- Let them show you a detailed description of their procedure and a sample report.
- Use this information to compare multiple vendors.

Do you have any questions or feedback?

We are constantly updating this FAQ. The aim is to inform you as the client as comprehensively as possible about all aspects of pentesting that affect you.

Please send us your questions and improvement requests. We give a direct answer and include it in the FAQ: office@mgm-sp.com or https://www.mgm-sp.com/en/contact/