Die große Application Security Penetration Test FAQ für Auftraggeber

DE

Alles, was man vor, während und nach Beauftragung eines Application Security Penetrationstests wissen sollte

Autoren: Das Pentestteam der 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 München
Tel.: +49 89 358680-880
office@mgm-sp.com
http://www.mgm-sp.com

Der **Penetrationstest** - oder kurz **Pentest** - ist die nach wie vor beliebteste Art, Sicherheitsmängel in Webanwendungen aufzudecken. Die Verbreitung dieses Mittels rührt vor allem daher, dass es sich ohne großen Aufwand aufsetzen und von einem entsprechenden Experten bzw. einem darauf spezialisierten Unternehmen durchführen lässt.

Diese Einfachheit täuscht leicht darüber hinweg, dass der Erfolg eines Pentests von einer Reihe von Faktoren abhängt, die es zu berücksichtigen gilt. Ein Pentest ist dann erfolgreich, wenn er bei Minimierung des Aufwands zu einer maximalen Abdeckung führt. Frage 12 geht darauf ein, warum sich die Aufgabenstellung nicht als das Auffinden *aller* Schwachstellen definieren lässt.

Sind optimale Rahmenbedingungen nicht gewährleistet, droht die Gefahr des sich in falscher Sicherheit Wiegens (Frage 5): Man nimmt fälschlicherweise an, die Anwendung sei - bis auf die möglicherweise gefundenen Schwachstellen - sicher und wird in dem Bemühen um umfassende Sicherheit nachlässiger.

Ein wesentlicher Beitrag, dem entgegenzuwirken, besteht in der Kenntnis der Fallstricke einer Pentestbeauftragung und ihrer Vermeidung: Von der Befähigung des Pentesters, den Umfang korrekt abschätzen zu können, über das Sicherstellen des reibungsfreien Ablaufs bis hin zum richtigen Umgang mit den Ergebnissen des Tests.

Diese FAQ befasst sich mit den vielen Fragen, die sich rund um das Pentesten von Webanwendungen stellen - und gibt praxisnahe Antworten. Plus eine Vielzahl von Erfahrungen aus der Praxis und manche tiefergehende Ein- und Ansicht aus einem langen Pentesterleben. Sie wendet sich an alle, die ihre Webanwendung einer solchen Überprüfung unterziehen möchten.

Ein solches, auf ein Thema ausgerichtetes Dokument bringt es mit sich, dass der Eindruck entsteht, der Pentest sei das alleinige oder beste Mittel zur Herstellung von Sicherheit in Webanwendungen. Zumindest Ersteres wäre eindeutig falsch und Letzteres ist in vielen Fällen ebenfalls nicht richtig (siehe Frage 35 und 36). Wenn man einen Superlativ vergeben möchte, dann den, dass der Pentest das am wenigsten verzichtbare Mittel ist.

Grundlegendes

1 Welche Begriffe sollte ich kennen?
2 Was ist das überhaupt, eine Schwachstelle?
3 Warum sind kundenspezifisch entwickelte Webanwendungen eine ernstzunehmende Bedrohung für die Sicherheit?
4 Warum darf man Sicherheitsmängel nicht mit Qualitätsmängeln gleichsetzen?
5 Warum sollte ich das Phänomen des False-Sense-of-Security kennen?
6 Berührt die im Mai 2018 eingeführte europäische Datenschutz-Grundverordnung (DSGVO/GDPR) das Pentesten?

Penetrationstest Basics

7 Was ist ein Penetrationstest?
8 Und was ist ein Application Security Penetrationstest nicht?
9 Was hat ein Penetrationstest mit der Suche nach der Stecknadel im Heuhaufen zu tun?
10 Was lässt sich gegen die Überlegenheit der Angreifer tun?
11 Was ist ein Nachtest?
12 Findet man mit einem Pentest alle Schwachstellen?
13 Findet ein Pentest immer alle Stellen, an denen eine bestimmte Schwachstelle auftritt?
14 Was ist von günstigen Pentests (Quicktest, Ersttest, Low-Budget-Test, "Low-hanging-Fruits"-Test, rein tool-basierter Test) zu halten?

Eigenschaften von Pentests

15 Blackbox, Graybox, Whitebox?
16 Wie läuft ein Pentest ab?
17 Was wird bei einem Application Pentest überhaupt alles getestet?
18 Wie werden gefundene Schwachstellen bewertet?
19 Was ist der CVSS-Score und warum sollte ich mich damit befassen?
20 Wie verlässlich ist die Bewertung und wie gehe ich mit ihr um?
21 Kann man Pentests automatisiert durchführen?
22 Soll der Pentest auf dem Test- oder auf dem Produktivsystem stattfinden?

Umfang und Preis eines Pentests

Umgang mit dem Ergebnisbericht

Zu guter Letzt

Haben Sie Fragen oder Feedback?

Grundlegendes

1 Welche Begriffe sollte ich kennen?

Den Pentester nennen wir ab jetzt einfach **Tester** und den, der die Sicherheit bedroht, den **Angreifer**. Den Nutzer der Anwendung, der häufig das Opfer eines Angriffs ist, nennen wir **User**. Eine Sicherheitslücke, ein Sicherheitsproblem oder eine Angreifbarkeit nennen wir einheitlich **Schwachstelle** (englisch: **Vulnerability**), eine entdeckte Schwachstelle ein **Finding**. Ein System, bei dem ein Sicherheitsvorfall eingetreten ist, bezeichnen wir als **kompromittiert**, den Vorfall selbst als **Kompromittierung**. Den Begriff **Gefahrenpotenzial** verwenden wir, um die Gefährlichkeit einer Schwachstelle - Synonyme: Kritikalität, Bedrohung oder Risiko - auszudrücken. Der Begriff des **Risikos** ist im Rahmen dieses Dokuments seiner technisch-fachlichen Definition vorbehalten, nämlich dem Produkt aus der Wahrscheinlichkeit dafür, dass das Unerwünschte eintritt (**Eintrittswahrscheinlichkeit**) und der Höhe des daraus (maximal) entstehenden Schadens (**Schadenshöhe**). Ein wichtiges Konzept in Sachen Risiko ist das des **Risikomanagements**. Dieser Begriff bringt etwas ganz Wesentliches zum Ausdruck, nämlich dass das Ziel aller Sicherheitsbemühungen in der Regel nicht das Erreichen absoluter Sicherheit ist, sondern vielmehr das Herbeiführen der richtigen Balance aus den negativen Auswirkungen von Sicherheitsmaßnahmen (Kosten, Zeitverzug, Einschränkungen etc.) und den negativen Auswirkungen von zu wenig Sicherheit.

2 Was ist das überhaupt, eine Schwachstelle?

Eine Schwachstelle ist jede Eigenschaft einer Anwendung, bei der man davon ausgehen kann, dass sie nicht beabsichtigt ist und dass sie von einem Dritten missbraucht werden kann - also in einer Form ausnutzbar ist, die bei dem Betreiber der Anwendung zu einem

Schaden führt. Dies umfasst auch indirekte Schäden, etwa wenn die direkten Leidtragenden die User sind.

3 Warum sind kundenspezifisch entwickelte Webanwendungen eine ernstzunehmende Bedrohung für die Sicherheit?

Webanwendungen besitzen eine ganze Reihe von Eigenschaften, die sie zu einem ernstzunehmenden Sicherheitsrisiko machen.

- Kundenspezifisch entwickelte Webanwendungen sind Unikate. Damit haben sie keinen industriellen Reifeprozess durchlaufen wie Anwendungen "von der Stange", bei denen i.d.R. alleine über die große Stückzahl qualitätssteigernde und fehlerbeseitigende Effekte eintreten. Die Wahrscheinlichkeit für sicherheitsrelevante Fehler ist damit per se sehr hoch.
- Eine Anwendung mit einem hohen Qualitätsanspruch zu entwickeln, ist signifikant teurer, als dieselbe Anwendung ohne diesen Anspruch zu programmieren. Dennoch wird man nach der Fertigstellung auf den ersten Blick zwischen der preisgünstigen und der teuren Variante keinen Unterschied feststellen. Dieser wird hinsichtlich der klassischen Software-Qualitätsmetriken erst mit der Zeit, u.a. in hohen Kosten und hoher Fehleranfälligkeit bei der Weiterentwicklung, sichtbar. Hinsichtlich der Sicherheit drückt sich der Unterschied in einem hohen Schadensrisiko durch das Vorhandensein von Schwachstellen aus. Das für ein Softwareprojekt bereitgestellte Budget wird diesen Anforderungen häufig nicht gerecht, was sich in Qualitäts- und Sicherheitsmängeln niederschlägt.
- Die einer Anwendung innewohnende Komplexität wird oft unterschätzt. Was seinen Grund wohl darin hat, dass die innere Struktur unsichtbar ist, die vielen inneren Abhängigkeiten nach außen nicht in Erscheinung treten und beim Softwareherstellungsprozess auch kein "Materialverbrauch" auftritt, an dem sich die Größe der Aufgabe ablesen ließe. Komplexität bedeutet jedoch immer auch eine hohe Wahrscheinlichkeit für Fehler.

- Der Wissensstand zu Sicherheitsthemen ist bei vielen Softwareingenieuren immer noch auf einem recht niedrigen Niveau.
- Viele Application Frameworks nehmen sich des Themas Sicherheit nicht weitreichend genug an. Würden Sicherheitsmaßnahmen, die sich im jeweiligen Framework fest einprogrammieren ließen, von den Framework-Entwicklern dort auch umgesetzt, so hätte der Programmierer viel weniger Möglichkeiten, überhaupt noch etwas unsicher zu programmieren.
- Sehr häufig besitzen Webanwendungen eine Verbindung zu sensiblen Daten oder haben direkten Zugriff darauf. Auch wenn diese - auf der Netzwerkebene zumeist gut geschützt - auf einem anderen System liegen, sind sie doch über die Webanwendung erreichbar. Bei Sicherheitslücken in der Anwendung sind die Schutzmaßnahmen auf Netzwerk- und Systemebene in aller Regel wirkungslos.

Maßnahmen zur Sicherstellung von Application Security sind also besonders bei kundenspezifisch entwickelten Anwendungen von großer Bedeutung.

4 Warum darf man Sicherheitsmängel nicht mit Qualitätsmängeln gleichsetzen?

Bei Sicherheitsmängeln handelt es sich, genauso wie bei Qualitätsmängeln, um Fehler in der Anwendung. In beiden Fällen ist durch entwicklungsbegleitende Maßnahmen dafür zu sorgen, dass sie möglichst gar nicht erst auftreten. Und durch laufendes und abschließendes Testen gilt es in beiden Fällen, die Fehlerquote weiter zu minimieren.

Trotz dieser Verwandschaft macht man einen - gefährlichen - Fehler, wenn man die Erkenntnisse und Verfahren aus der Qualitätssicherung unverändert auf die Security anwendet. In folgender Eigenschaft ist dies begründet: Bei Qualitätsmängeln besteht zumeist - sehr grob betracht - ein proportionales Verhältnis zwischen Größe des Fehlers und Größe des Schadens. Kleiner Fehler - kleiner Schaden, großer Fehler - großer Schaden. Bei Sicherheitsproblemen hat das Verhältnis

hingegen eher die Form einer Sprungfunktion: selbst ein kleiner oder schwer zu findender Fehler kann sehr großen Schaden verursachen.

Am Beispiel einer Shoppinganwendung sei dies illustriert: Angenommen, der Betreiber des Shops lässt Änderungen an der Check-out-Funktion durchführen. Eine solche Änderung birgt u. a. die Gefahr, dass das Check-out durch Fehlprogrammierung nicht mehr funktioniert und, zumindest kurzzeitig, ein totaler Umsatzverlust eintritt. Dennoch ist das die Änderung begleitende Schadensrisiko sehr klein, denn dieser schlimmstmögliche Schadensfall kann durch entsprechend ausgerichtete Abschlusstests leicht vermieden werden. Wenn komplexere Fehlerkonstellationen - etwa: das Auftreten eines nicht-lateinischen Zeichens im Kundennamen bringt das Check-out zum Absturz - dabei nicht sofort entdeckt werden, ist das verschmerzbar, der Fall ist selten und der Schaden entsprechend klein.

Bei der Security verhält es sich anders. Nehmen wir an, besagte Änderung besteht in der Einführung von Gutscheincodes und dem zugehörigen Eingabefeld im Check-out. Gehen wir weiter von der - durchaus realistischen - Annahme eines hohen Termindruck bei der Umsetzung aus. Dann kann es leicht passieren, dass der zuständige Entwickler den zur Prüfung der Gültigkeit des Gutscheins benötigten Zugriff auf die Gutscheindatenbank nicht solide durch entsprechende Erweiterung des Datenmodells umsetzt, sondern "mal eben schnell" per direktem Datenbankaufruf. (Für Insider: Statt sicherer Persistierung per OR-Mapper die äußerst unsichere Methode des dynamischen SQL-Aufrufs).

Eine auf diese Weise eingebrachte sog. SQL-Injection-Schwachstelle ist äußerst gefährlich, weil darüber im schlimmsten Fall in die Datenbank oder das gesamte System eingedrungen werden kann. Ein reiner Qualitätstest entdeckt diese Schwachstelle nicht und sie macht sich auch sonst in keiner Weise bemerkbar. In dem Moment, wo sie von einem Angreifer entdeckt worden ist, tritt dann aber umgehend der Maximalschaden ein.

Die im Bereich der Qualitätssicherung vertretbare Logik "Wir haben die kritischsten Fehlerfälle intensiv getestet und alles ist in Ordnung. Wenn wir aus Zeit- oder Kostengründen versteckte Qualitätmängel übersehen haben, ist das zwar nicht erfreulich, aber auch nicht so

schlimm." darf also nicht auf die Security übertragen werden. Die "schlimmen Fälle" lassen sich hier nicht gezielt adressieren und ausschließen.

Die Lösung für diese Problematik besteht dabei nicht darin, auch bei kleinen Änderungen immer einen kompletten Penetrationstest durchzuführen. Die auftretenden Kosten und der Zeitverzug lassen das in der Praxis nicht zu. Vielmehr ist diese Problematik ein weiterer Grund dafür, Application Security tief im Prozess zu verankern und den Pentest als nur eine von mehreren Maßnahmen zu begreifen. Siehe auch Frage 35 und 36.

5 Warum sollte ich das Phänomen des False-Sense-of-Security kennen?

Das Wissen darum, dass ein Risiko nicht ausreichend durch sichernde Maßnahmen abgedeckt ist (also eine Sicherheitslücke besteht), bewirkt, dass man bei allem, was mit dieser Sicherheitslücke im Zusammenhang steht, entsprechende Vorsicht walten lässt, dh. hinsichtlich der Sicherheit eine höhere Messlatte anlegt. Der Information Officer in einem Unternehmen, wo im Intranet unverschlüsselt via HTTP kommuniziert wird, wird bei der Festlegung von Sichheitsregeln für Externe auf eine größere Strenge hinwirken als der Kollege in dem Unternehmen, wo alle Anwendungen ausschließlich über https erreichbar sind. Umgekehrt wird man als Maßnahme die https-Umstellung des Intranets vornehmen, wenn sich herausstellt, dass die geforderte Strenge nicht umsetzbar ist.

Oder, bezogen auf Webanwendungen: Die Erkenntnis, dass man hinsichtlich der Application Security schlecht aufgestellt ist, führt zur Entscheidung, sensible Anwendungen erst einmal nicht über das Web bereitzustellen. Ist man hier hingegen gut unterwegs, etwa, weil man das Pentesten systematisch betreibt, fällt die Entscheidung eher in die Richtung der Zulassung einer sensiblen Anwendung aus.

Ein Problem ergibt sich dann, wenn hinsichtlich des Maßes der vorhandenen Sicherheit eine Lücke klafft zwischen dem, was man denkt, dass man hat und dem, was wirklich ist. Im ersten Beispiel: Die Sicherheitsregeln sind locker ausgelegt, weil das Intranet jetzt ja verschlüsselt ist - man aber übersieht, dass dies nur lückenhaft erfolgt

ist. Im zweiten Beispiel: Die Entscheidung, sensible Daten nach außen verfügbar zu machen, fällt positiv aus, weil man die Sicherheit mittels Pentests ja im Griff hat - und übersieht, dass die Pentestvorgaben viel zu schwach sind für derartige Anwendungen.

Zusammengefasst: Wenn man feststellt, dass sich eine Sicherheitsmaßnahme nur unzureichend umsetzen lässt, dann darf man sich auf sie auch nicht abstützen! Manchmal ist es in einem solchen Fall sogar besser, die Sicherheitsmaßnahme gar nicht umzusetzen. So vermeidet man, dass aus dem daraus entstehenden False-Sense-of-Security an anderer Stelle Entscheidungen getroffen werden, die die Sicherheit weit mehr gefährden als die ausgelassene Sicherheitsmaßnahme.

Siehe dazu auch Frage 14.

6 Berührt die im Mai 2018 eingeführte europäische Datenschutz-Grundverordnung (DSGVO/GDPR) das Pentesten?

Ja, eindeutig. Denn zu den bereits bestehenden Bedrohungen ist eine weitere hinzugekommen, nämlich die von hohen Strafzahlungen, wenn über eine Schwachstelle personenbezogene Daten in fremde Hände oder in die Öffentlichkeit gelangen. Die Anwendung von Sicherheitsmaßnahmen im Allgemeinen und das Pentesten im Besonderen sind also seit Inkrafttreten der Europäischen Datenschutzgrundverordnung von noch größerer Bedeutung.

Penetrationstest Basics

7 Was ist ein Penetrationstest?

Bei einem Penetrationstest nimmt der Tester die Rolle des Angreifers ein. Er greift mit allen ihm zur Verfügung stehenden Mitteln *von außen* - im Unterschied zu Analysetechniken, die "innen" ansetzen, wie etwa die Codeanalyse - auf die Anwendung zu, um Schwachstellen aufzudecken. Dabei setzt er sein umfangreiches Wissen über Schwachstellen ein und wendet allerlei Tricks zum Aushebeln von Sicherheitsmechanismen an. Das Ergebnis ist ein Bericht, der die Schwachstellen in nachvollziehbarer Weise beschreibt, sie hinsichtlich ihres Gefahrenpotenzials bewertet und Gegenmaßnahmen nennt.

Weil er als gutartiger "Hacker" agiert, wird der Pentester auch gerne **White-Hat Hacker** (im Unterschied zum bösartigen **Black-Hat Hacker**) oder **Ethical Hacker** genannt.

8 Und was ist ein Application Security Penetrationstest nicht?

Bei einem Penetrationstest einer Web- oder mobilen Anwendung geht es *nicht* darum, das Angreiferszenario möglichst realistisch nachzustellen, um daraus schließen zu können, ob ein Angreifer in die Anwendung eindringen könnte oder nicht. Und diese im ersten Fall als unsicher, im zweiten als sicher zu bewerten. Vielmehr ist der Penetrationstest als Qualitätssicherungsmaßnahme zu verstehen. Es gilt, möglichst alle, die Sicherheit einer Anwendung berührenden Auffälligkeiten zu erkennen, sie, auch im Zusammenhang, zu bewerten und, wenn sie ein tatsächliches Risiko darstellen, einer Behebung zuzuführen. Der Tester kann seine Rolle dann am wirkungsvollsten ausführen, wenn er dabei die bestmögliche Unterstützung erhält - mehr dazu in Frage 10.

9 Was hat ein Penetrationstest mit der Suche nach der Stecknadel im Heuhaufen zu tun?

Sehr viel! Um genau zu sein, das zugrunde liegende Prinzip ist so ziemlich dasselbe. Nur dass es beim Pentest um viele "Nadeln" geht. Beiden gemeinsam ist diese unangenehme Eigenschaft: Die Wahrscheinlichkeit etwas zu finden steigt mit der Dauer der Suche und mit der Anzahl der Suchenden, ohne dabei jemals 100% zu erreichen. Folglich liegt es in der Natur der Sache, dass das "crowd-gesourcete" Internet, also die Summe aller Angreifer, die versuchen, die Webanwendung zu hacken, jedem mit einem begrenzten Zeitbudget ausgestatteten Penetrationstester überlegen ist. Denn sowohl Anzahl als auch Suchdauer sind "auf der anderen Seite" potenziell unbegrenzt.

10 Was lässt sich gegen die Überlegenheit der Angreifer tun?

Weil die Überlegenheit der Gegenseite bei der Methode des "stochastischen" Suchens so erdrückend ist, ist es umso wichtiger, dem etwas entgegenzusetzen. Das wirksame Gegenmittel ist die Bereitstellung eines Maximums an Informationen. Je mehr der Tester über die Interna der Anwendung weiß, desto weniger muss er sinnlos im Nebel stochern, desto gezielter und effektiver kann er das zur Verfügung stehende Zeitbudget einsetzen.

Zu den Informationen gehören u. a.: Programmiertechnologie, eingesetzte Frameworks, Beschreibungen der Softwarearchitektur, Sicherheitskonzepte, API-Beschreibungen und je nach Art der Anwendung weitere, individuell abzustimmende Informationen.

Siehe auch Frage 15

11 Was ist ein Nachtest?

Wenn der Pentest behebenswerte Schwachstellen aufgedeckt hat, sollten diese, nachdem sie nach bestem Wissen behoben worden sind, nachgetestet werden. Dabei wird geprüft, ob sie durch die durchgeführte Maßnahme auch tatsächlich vollständig behoben sind. Streng genommen müsste man den kompletten Pentest erneut

durchführen, denn eine Änderung an einer Stelle kann zu Sicherheitsproblemen an ganz anderer Stelle führen. Aus Kosten- und auch Zeitgründen wird ein Nachtest aber in aller Regel in folgender Weise durchgeführt: Er prüft, ob die gefundene Schwachstelle *an der zuvor berichteten Stelle des Auftretens* nun nicht mehr auftritt. Er liefert also nicht die Aussage, dass die Schwachstelle grundlegend behoben ist, also auch an *keiner anderen Stelle* vorhanden ist.

Weitere Bedeutung erlangt dieser Sachverhalt im Zusammenhang mit dem folgenden Absatz.

12 Findet man mit einem Pentest alle Schwachstellen?

Nein, in aller Regel lässt sich nach einem Pentest nicht die Schlussfolgerung ziehen, dass die Anwendung außer den erkannten keine weiteren Schwachstellen enthält. Das liegt im Wesentlichen an der Stecknadel-im-Heuhaufen-Natur von Pentests (Frage 9). Folgende Regel lässt sich hieraus ableiten:

Regel #1: Verlass dich nicht alleine auf den Pentest, sondern ergreife zusätzliche Maßnahmen zur Herstellung der Sicherheit deiner Anwendung.

Siehe dazu auch Frage 36.

13 Findet ein Pentest immer alle Stellen, an denen eine bestimmte Schwachstelle auftritt?

Nehmen wir eine ganz normale Webanwendung und die **SQL-Injection**-Schwachstelle als Beispiel. Die Anwendung besitzt eine Reihe von Formularen und jedes Formular enthält eine Reihe von Feldern. SQL-Injection tritt vorzugsweise bei der Verwendung der in die Felder eingegebenen Daten auf. Der Pentester prüft u. a. also sukzessive die Formulare und Felder auf SQL-Injection. Trifft er auf diese Schwachstelle, so beschreibt er sie im Bericht. Dann prüft er ggf. noch weitere Felder, aber jetzt nur noch stichprobenhaft. Es wird also nicht *jede* Stelle, an der SQL-Injection vorhanden ist, in dem Bericht erscheinen, da die dafür aufzuwendende Zeit viel besser für die Suche nach anderen Schwachstellen genutzt werden kann.

Kommt ein Vulnerabilityscanner bei der Suche nach bestimmten Schwachstellenarten zum Einsatz (SQL-Injection ist eine der Schwachstellenkategorien, bei der Scantools oft gute Dienste leisten), so ist die Liste der aufgeführten Stellen höchstwahrschenlich länger, aber auch hier gilt, dass Vollständigkeit nicht garantiert ist. Oft entdeckt ein Scanner nämlich nicht alle Formulare und Felder oder stößt auf anderweitige Probleme, die einem automatischen Testen im Wege stehen.

Es ergibt sich: Der Pentestbericht enthält nicht immer alle Stellen des Auftretens einer bestimmten Schwachstelle, sondern unter Umständen lediglich eine oder wenige ausgewählte. Das wiederum hat erhebliche Auswirkungen auf die Aufgabe des Behebens der Schwachstelle. Siehe dazu Frage 41.

Man beachte: Ein einfacher Nachtest (Frage 11) deckt derartige Fehler nicht mehr auf!

14 Was ist von günstigen Pentests (Quicktest, Ersttest, Low-Budget-Test, "Low-hanging-Fruits"-Test, rein tool-basierter Test) zu halten?

Generell gilt: Jede Art von Test ist besser als gar kein Test!

Aber nur unter einer Bedingung: Man unterliegt nicht dem False-Sense-of-Security (Frage 5)!

Ein Quicktest ist für Anwendungen mit normalem oder hohem Schutzbedarf (Fragen 29 und 32) in aller Regel keine ausreichende Sicherheitsmaßnahme, aber er leistet gute Dienste dabei,

- eine begrenzte, aber in ihrem Umfang schwer zu präzisierende Risikominderung zu erreichen.
- dem Auftraggeber eine erste Indikation über das Sicherheitsniveau zu geben.
- die Notwendigkeit und Art weitergehender Maßnahmen abzuschätzen.

Eigenschaften von Pentests

15 Blackbox, Greybox, Whitebox?

Abhängig davon, wie informationsbehaftet der Pentester seine Arbeit antritt, wird ein Pentest als Blackbox-, Greybox- oder Whitebox-Pentest bezeichnet. Bei einem **Blackbox-Pentest** erhält der Tester keine zusätzlichen Informationen, findet sich also in derselben Situation wie ein fremder Dritter wieder. Wie wir in Frage 10 gelernt haben, ist diese Variante in aller Regel für das Testen von Webanwendungen nicht geeignet.

Der **Whitebox-Pentest** stellt das anzustrebende Gegenteil dar - volle Information. Das geht bis hin zur Einsichtnahme in den Quellcode der Anwendung. Da diese zumeist mit einem deutlich höheren Aufwand verbunden ist, wird weitaus häufiger der Kompromiss des **Greybox-Pentests** angewendet - maximale Information, aber keine Zurverfügungstellung des Quellcodes.

Vom Whiteboxtest mit Codeeinsicht zu unterscheiden ist übrigens die Maßnahme des **selektiven Code-Reviews** bzw. der umfassenden **automatischen statischen Codeanalyse**, bei der es sich um eine sinnvolle zusätzliche Maßnahmen zum Pentest handelt.

Regel #2: Der Pentester ist mit einem Maximum an Informationen auszustatten; es ist ein Greybox- oder Whitebox-Penetrationstest durchzuführen.

16 Wie läuft ein Pentest ab?

1. Angebotserstellung

Der Grundstein für die erfolgreiche Durchführung eines Pentests wird bereits bei der Angebotserstellung gelegt. Je besser der Dienstleister

in die Lage versetzt wird, Umfang und Gesamtscope abzuschätzen, desto reibungsloser wird dieser verlaufen. Siehe Frage 32 zur Ermittlung des Schutzbedarfs, Frage 33 zur Komplexität einer Anwendung und Frage 34 zur Bestimmung des Testaufwands.

2. Testvorbereitung

Gleiches gilt für die Güte der Vorbereitung. Zum vereinbarten Starttermin sollten alle mit dem Dienstleister abgestimmten Vorbereitungsmaßnahmen getroffen sein. Das ist nicht ohne Aufwand für den Auftraggeber und entsprechend einzuplanen. Dass dabei vieles schief gehen kann, zeigen die in Frage 38 aufgegriffenen Erfahrungen.

3. Testdurchführung

Der Penetrationstester nimmt die Rolle eines externen Angreifers ein. Er setzt all seine Erfahrung und die zur Verfügung gestellten Informationen ein, um - unterstützt durch professionelle Tools - Schwachstellen in der zu überprüfenden Anwendung auszumachen. Die Reporterstellung geht meist Hand in Hand mit dem Testen.

Auch hier ist Unterstützung des Auftraggebers erforderlich, um die in Frage 38 genannten Probleme zu vermeiden.

4. Berichtsübergabe

Wegen der in Frage 41 angesprochenen Problematik ist es empfehlenswert, den Pentest mit dem Erhalt des Berichts nicht abzuschließen, sondern mit dem Dienstleister nach eingehender Beschäftigung mit den berichteten Schwachstellen Rücksprache zu halten. Siehe Frage 40.

5. Nachtest

Nach Behebung der Schwachstellen ist die Durchführung eines Nachtests zu empfehlen, bei dem - im meist praktizierten, einfachsten Fall - die Stellen in der Anwendung, an denen zuvor Schwachstellen gefunden worden sind, erneut überprüft werden. Was es dabei zu beachten gilt, ist in Frage 11 besprochen.

17 Was wird bei einem Application Pentest überhaupt alles getestet?

Kurz gesagt, das meiste von dem, was bei der Umsetzung der Anwendung schiefgehen kann und sich negativ auf die Sicherheit auswirkt. Dazu gehören vor allem Programmierfehler, aber auch fehlerhafte Konfiguration und logische Probleme.Folgende Kategorisierung erleichtert die Übersicht:

1. **Systemebene**: Hier wird die Softwareumgebung, in die die Webanwendung eingebettet ist, untersucht. Also die sichere Konfiguration von Webserver, CMS und sonstiger Systemsoftware, die benötigt wird, damit die Anwendung laufen kann. In bestimmten Fällen ist dieser Bereich ausgeklammert, siehe Frage 24.
2. **Technologie**: Sind grundlegende Sicherheitseigenschaften der Anwendung durch die (korrekte) Anwendung sicherheitsgebender Technologien gewährleistet? In diesen Bereich gehört der Einsatz von sicheren Verschlüsselungsverfahren, die sichere Passwortablage, die Implementierung einer Sicherheitsarchitektur etc.
3. **Implementierung**: Das ist der große Bereich, in dem geprüft wird, ob dem Entwickler sicherheitsrelevante Programmierfehler unterlaufen sind: SQL-Injection, XSS, CSRF usw. - die Liste möglicher Schwachstellen ist lang.
4. **Logik**: Wenn eine Funktion technisch korrekt programmiert ist, bei den inneren Abläufen aber Missbrauchsmöglichkeiten übersehen worden sind, spricht man von logischen Schwachstellen. Als Beispiel: Passwort-Cracking wird nicht verhindert, die Registrierungsfunktion kann zum massenhaften Versenden von Emails missbraucht werden oder beim Verfahren zum Zurücksetzen des Passworts können Fälle auftreten, die es einem Dritten ermöglichen, in das Konto einzudringen.
5. **Semantik**: Wenn ein Angreifer beispielsweise eine Phishing-Attacke plant, dann sucht er sich eine Anwendung aus, bei der zu erwarten ist, dass deren Nutzer besonders anfällig sind. Das ist etwa dann der Fall, wenn Nutzer es gewohnt sind, dass sie von einem Unternehmen regelmäßig Emails bekommen, in denen Links auf

die Anwendung enthalten sind oder sie sogar aufgefordert werden, auf eingebettete Links zu klicken. So "vorkonditionierte" Nutzer werden weit häufiger auch auf einen betrügerischen Link in einer gefälschten Mail klicken als Nutzer, die von ihrem Unternehmen an Mails ohne Links gewöhnt sind. Eigenschaften, die diese oder ähnliche Angriffe begünstigen, werden hier geprüft.

18 Wie werden gefundene Schwachstellen bewertet?

Ein standardisiertes Bewertungssystem für das Gefahrenpotenzial von Schwachstellen gibt es nicht. Dennoch bewerten die meisten Anbieter Schwachstellen sehr ähnlich, nach einem Schema, bei dem das Gefahrenpotenzial in die Kategorien **niedrig**, **mittel**, **hoch** und **kritisch** eingeteilt ist.

Die Kategorien lassen sich grob folgendermaßen interpretieren:

kritisch: Diese Schwachstelle stellt ein inakzeptables Risiko dar. Die Anwendung darf nicht live geschaltet werden; falls bereits geschehen, ist sie umgehend zu deaktivieren.

hoch: Diese Schwachstelle ist umgehend, ggf. im Rahmen eines Notfall-Patches, zu beheben. Weitergehende Maßnahmen zur Risikovermeidung bis zur Behebung sind in Betracht zu ziehen.

mittel: Diese Schwachstelle ist zu beheben, auch dann, wenn dies mit (moderaten) Zusatzkosten oder anderen (moderaten) Nachteilen verbunden ist.

niedrig: Die Behebung kann regulär in die Releaseplanung aufgenommen werden.

Manchmal kommt ein weiterer Parameter (**Eintrittswahrscheinlichkeit** oder **Komplexität**) hinzu, der Auskunft darüber gibt, wie schwierig es ist, die Schwachstelle zu finden und auszunutzen - also wie wahrscheinlich es ist, dass der Schadensfall über die jeweilige Schwachstelle auch tatsächlich eintritt.

Noch weitergehend differenziert die in Frage 19 vorgestellte CVSS-Methodik (Common Vulnerability Scoring System), deren Bewertungen auf einer Skala von 0 bis 10 angegeben werden. Obwohl ihre Praktikabilität im Bereich von Webanwendungen eher eingeschränkt ist, findet sie eine immer breitere Verwendung.

Der Pentester nimmt seine Bewertung vorwiegend anhand allgemeiner "Gefährlichkeitsmaßstäbe" einer Schwachstellenart vor, anwendungsspezifische Bedrohungen und Missbrauchsszenarien kann er dabei nur sehr begrenzt mit in die Waagschale werfen. Es ist daher durchaus anzuraten, nach Fertigstellung des Berichts diesen um eine Eigenbewertung zu ergänzen: Dabei wird geprüft, ob sich durch Einbeziehung von Faktoren, die der Pentester nicht kennt, andere Bewertungen ergeben.

19 Was ist der CVSS-Score und warum sollte ich mich damit befassen?

"Das Common Vulnerability Scoring System (CVSS) ist ein Industriestandard zur Bewertung des Schweregrades von möglichen oder tatsächlichen Sicherheitslücken in Computer-Systemen" (Quelle: Wikipedia). Beim Entwurf hatte man hinsichtlich der Application Security Anwendungen "von der Stange" im Sinn, nicht jedoch die speziellen Anforderungen kundenspezifischer Webanwendungen, die von ihrer Natur Unikate sind. Dennoch hat der CVSS in den letzten Jahren mangels besserer Alternativen auch für die Bewertung von Schwachstellen in kundenspezifischen Webanwendungen immer mehr Verbreitung gefunden.

Im CVSS ergibt sich der Gesamtscore einer Schwachstelle aus der Zusammenführung einer Reihe von isoliert zu ermittelnden Bewertungen einzelner Aspekte (Metriken) unter Berücksichtigung von in der Berechnungsformel hinterlegten Gewichtungen.

Stärken:

- Zerlegung in einzeln zu bewertende Metriken erleichtert dem Pentester die Bewertung
- Sehr gute Vergleichbarkeit sowohl auf Ebene des Gesamtscores als auch auf Ebende der Metriken
- Leichte Nachvollziehbarkeit des Gesamtscores anhand "Drill-down" zu den Einzelbewertungen

Schwächen, bezogen auf Webanwendungen:

- Es sind Metriken einbezogen, die bei Webanwendungen unwichtig oder weniger wichtig sind (Beispiel: Attack-Vector-Metrik, Temporal Score, Remediation-Level-Metrik sind in einer Weise definiert, die bei kundenspezifischen Webanwendungen wenig Aussagekraft besitzt)
- Kleine Unterschiede in den Eingangsparametern können unter ungünstigen Umständen zu großen Unterschieden im Gesamtscore führen.

So muss man sagen, dass die zunehmende Verbreitung im Bereich der Application Security weniger der besonders guten Eignung als vielmehr dem Mangel an Alternativen zuzuschreiben ist.

Folgende Erfahrungen sollten bei der Verwendung des CVSS berücksichtigt werden:

- Wenn die Vergleichbarkeit der Bewertung über viele Anwendungen und/oder verschiedene Pentester wichtig ist - beispielsweise ist das der Fall bei großen Unternehmen mit vielen Anwendungen oder bei der Einbindung in DevOps-Prozesse - sollte der CVSS als Bewertungsschema in Betracht gezogen werden.
- Unternehmen, die hin und wieder Pentests durchführen oder wo die Vergleichbarkeit aus anderen Gründen keine Rolle spielt, sollten eher das Kriterium der Einfachheit zugrundelegen. Der CVSS ist dann nicht die erste Wahl.
- Kontraproduktiv wirkt der CVSS dann, wenn die Regeln für die aus einer Schwachstelle zu ziehenden Konsequenzen zu starr festgeschrieben werden. Wenn also im Rahmen des Deplymentprozesses ein Score, der einen festgelegten Schwellwert überschreitet, automatisch dazu führt, dass der - möglicherweise kurz bevorstehende und für das Geschäft sehr wichtige - Go-Live platzt. Der Prozess sollte daher unbedingt eine Feedbackschleife enthalten, die sicherstellt, dass Schwachstellenbewertungen mit schwerwiegenden Konsequenzen einer Detailbegutachtung unterzogen werden dürfen und die CVSS-Bewertung manuell korrigiert werden kann.

20 Wie verlässlich ist die Bewertung und wie gehe ich mit ihr um?

Eine Schwachstelle in jeder Erscheinungsform und hinsichtlich aller Anforderungen immer objektiv und belastbar zu bewerten, ist kaum möglich. Dazu sind die technischen Zusammenhänge zu komplex und die Interpretationsspielräume zu groß. Die Bewertung ist eine auf den dem Pentester zur Verfügung stehenden - meist sehr eingeschränkten - Informationen beruhende Einschätzung. Im Zweifel wird der Pentester in Richtung mehr Sicherheit gehen und das Risiko höher einstufen.

Insbesondere dann, wenn die Beseitigung einer Schwachstelle mit hohen direkten Kosten (die Behebung selbst) oder hohen indirekten Kosten (etwa wegen Umsatzverlust durch Verzögerung oder Unterbrechung) verbunden ist, empfiehlt es sich, eine differenziertere Risikobewertung durchzuführen.

Diese im Bewertungssystem steckende Unschärfe ist auch der Grund dafür, dass es nicht unbedingt eine gute Idee ist, in den Freigabeprozess unumstößliche Regeln einzubauen, die verhindern, dass eine Anwendung nicht freigegeben wird, solange sie noch Schwachstellen der Kategorie x enthält. Stattdessen sollte ein solcher Fall eine detaillierte Risikobewertung nach sich ziehen, auf deren Grundlage dann die Entscheidung "Go" oder "No-Go" gefällt wird.

21 Kann man Pentests automatisiert durchführen?

Sogenannte DAST-Tools (Dynamic Application Security Testing) oder Application-Vulnerability-Scanner versprechen, eine Webanwendung umfassend automatisch nach Schwachstellen scannen zu können. Jedoch machen die große Vielfalt an Programmiertechnologien, die hohe Komplexität in der Anwendungslogik und die fehlende Standardisierung in den Dialogabläufen zwischen Client und Server eine derartige Automatisierung bis heute zu einem nicht zufriedenstellend gelösten Problem. So muss man sagen, dass DAST-Tools in speziellen Fällen (etwa beim Testen ganz bestimmter Schwachstellen oder in Build-Umgebungen mit hoher Wiederholrate der Tests) zwar ihre Berechtigung haben, als alleiniges Mittel aber nach wie vor ungeeignet sind. Ein seriöser Pentester setzt solche Tools

höchstens unterstützend zum manuellen Testen und als Ergänzung der vielen kleinen Tools in seiner Werkzeugkiste ein.

Pentests, die vollständig oder schwerpunktmäßig von einem Tool durchgeführt werden, sind in aller Regel nicht als Ersatz für einen manuell durchgeführten Pentest geeignet.

22 Soll der Pentest auf dem Test- oder auf dem Produktivsystem stattfinden?

Auf diese Frage gibt es keine eindeutige Antwort. Auch wenn die Gefahr, dass durch den (manuell gesteuerten) Pentest Schäden oder Seiteneffekte auftreten, sehr gering ist, ist dieses Szenario wenn möglich zu vermeiden. In einer idealen Welt sollte der Pentest in einer möglichst exakt der Live-Umgebung entsprechenden Staging- oder Vor-Produktionsumgebung stattfinden.

Häufig stehen dieser Anforderung jedoch weitere Rahmenbedingungen im Weg und der Pentest muss auf einem weniger idealen System, z. B. auch im Produktivbetrieb, durchgeführt werden.

Gegen die Testdurchführung auf dem Produktivsystem würde ebenfalls das Vorhandensein personenbezogener Daten sprechen. Denn rechtlich stellt das Pentesten eine **Datenverarbeitung im Auftrag** dar und unterliegt somit den Regeln der DSGVO. Siehe auch Frage 14.

Regel #3: Der Pentest ist auf dem System durchzuführen, das sich unter Einbeziehung aller Rahmenbedingungen als das geeignetste erweist. Man beachte mögliche Seiteneffekte wenn es das Produktivsystem ist, insbesondere was den Datenschutz betrifft.

23 Soll mit oder ohne Web Application Firewall (WAF) und Intrusion Detection System (IDS/IPS) getestet werden?

Diese Frage stellt sich für Unternehmen, die einen am Perimeter ansetzenden Schutz im Einsatz haben, welcher auf Ebene des HTTP-Protokolls filtert, also vor missbräuchlichen Eingaben und Zugriffen auf Webanwendungen schützen soll. Die klare Antwort lautet:

Regel #4: Ein Perimeterschutz (Web Application Firewall, IDS/IPS) ist während des Pentests zu deaktivieren.

Denn sonst verdeckt dieser ja die Schwachstellen in der Anwendung. Wir erinnern uns, Frage 8: Ein Application Security Pentest ist eine Qualitätssicherungsmaßnahme. Und, noch wichtiger: Wenn eine Filterregel nachträglich verändert wird - etwa, weil sie bei einer anderen Anwendung zu Problemen führt - steht plötzlich der Schutz aller von dieser Regel abhängigen Anwendungen auf dem Spiel.

Ausnahme: Wenn die Aufgabe nicht der Pentest der Webanwendung ist, sondern wenn es gilt, die Sicherheit des Gesamtsystems oder die Wirksamkeit der eingestellten Filterregeln zu prüfen.

Neben der Filterfunktion bieten einige Web Application Firewalls weiterere Schutzfunktionen, etwa die Verhinderung von Passwort-Cracking-Angriffen oder das systematische Abgreifen von Informationen durch massenhafte Zugriffe. Nutzt die Anwendung diese Schutzfunktionen, so sollte der Pentest zweigeteilt werden. Nachdem die Anwendung zunächst ohne Perimeterschutz umfassend getestet worden ist, sollte man die Wirksamkeit der weiteren Schutzfunktionen bei aktivierten Filtern prüfen. Damit hier keine Lücken entstehen, ist eine sorgfältige Abstimmung im Vorfeld unerlässlich.

24 Sollte auch der Host getestet werden?

In großen Unternehmen mit vielen Webanwendungen sind diese zumeist in eine professionell betriebene Hosting-Umgebung eingebettet. Die Sicherheit von Servern und Infrastruktur wird hier zentral gewährleistet, so dass ein Penetrationstest sich ausschließlich auf die Webanwendung konzentrieren kann.

Ist dies nicht der Fall, ist mit dem Dienstleister unbedingt zu klären, ob der Pentest auf den Host auszudehnen ist. Bei einem Host-Test werden u. a. folgende Komponenten untersucht bzw. folgende Fragen beantwortet:

- Ist das Server-Betriebssystem gehärtet?
- Sind keine überflüssigen Dienste und offenen Ports vorhanden?
- Ist der Webserver sicher konfiguriert?

- Befinden sich das Betriebssystem sowie sämtliche nach außen sichtbare Dienste auf einem aktuellen Versionsstand, zu dem keine Schwachstellen öffentlich bekannt sind?
- Werden verwendete Sandardsoftwaresysteme wie ein CMS (z. B. Wordpress, Drupal, Joomla) oder Shopping-Software (z. B. Magento, Shopify) usw. sicher verwendet und sind die Sicherheitseinstellungen korrekt gesetzt?

Im Unterschied zur Anwendungsebene leisten in diesem Bereich (der sog. "Systemebene" gemäß der Kategorien in Frage 17) automatisierte Tools wie *nessus* oder *OpenVAS* gute Dienste.

Erfolgt das Hosting in der Cloud, so gelten dieselben Regeln: Ist der Cloud Provider für die zugrunde liegende Infrastruktur verantwortlich, kann sich die Überprüfung auf die Webanwendung beschränken. Werden die Server vom Kunden verwaltet, ist ein Test des Hostsystems ratsam. Die Vorgaben der Cloudprovider für Pentests sind dabei zu beachten.

25 Was ist beim Pentesten von Webanwendungen in der Cloud oder Hostinganbietern zu beachten?

Manche Cloud- oder Hostinganbieter verbieten das (unangemeldete) Pentesten. Die Regelungen beim eigenen Provider sind also rechtzeitig vorab zu klären. Bei der Vorbereitung sollte der Pentest-Dienstleister unterstützen oder dem Auftraggeber diese Formalitäten sogar komplett abnehmen können.

26 In welchem Entwicklungsstadium soll der Pentest erfolgen?

Ein Pentest, dem die Aufgabe der abschließenden Sicherheitsabnahme eines Releases zufällt, ist auf dem finalen Zustand dieses Releases vor der Liveschaltung durchzuführen. Nachträgliche Änderungen sollten (eigentlich: dürfen!) dann keine mehr stattfinden, das würde das Vertrauen in die Ergebnisse des Pentests schmälern. Änderungen, die auf das Fixen gefundener Schwachstellen zurückzuführen sind, sollten mittels eines Nachtests (siehe Frage 11) verifiziert werden.

Darüber hinaus kann ein (umfassender oder partieller) Pentest zu jedem Zeitpunkt stattfinden, sofern ein lauffähiger Entwicklungsstand vorliegt. Solche Vorab-Pentests ersetzen aber nicht den abschließenden Pentest.

27 Wie häufig sollte ein Pentest stattfinden?

In der Theorie nach jeder Änderung, denn mit jeder Änderung können sich sicherheitsrelevante Fehler einschleichen, auch solche mit massiven Auswirkungen.

Die Praxis ist aber in vielen Fällen eine andere: Ist eine Anwendung bereits einem Pentest unterzogen worden, werden "gefühlt" kleine Änderungen zumeist ohne erneute Sicherheitsprüfungen freigeschaltet. Bei etwas umfangreicheren Änderungen besteht häufig der Wunsch, nur die geänderten oder neu hinzugekommenen Funktionsbereiche zu testen. Erst bei großen Änderungen und wenn der Pentest lange zurück gelegen hat, greift die Einsicht, dass ein erneuter kompletter Pentest durchzuführen ist.

Wenn das Qualitätskriterium Sicherheit von Anfang an in den Entwicklungsprozess integriert worden ist, also den in Frage 36 angesprochenen Empfehlungen gefolgt wurde, mag dieser pragmatische Ansatz ausreichend sein. Ohne diese Voraussetzung muss dieses häufig beobachtete Vorgehen jedoch als riskant eingestuft werden.

Auf jeden Fall gilt: Spätestens wenn große Änderung an Architektur und Funktionalität durchgeführt werden, verliert der ursprüngliche Penetrationstest seine Aussagekraft und eine erneuter umfassender Test ist notwendig!

28 Wie testet man bei agiler Entwicklung und Continuous Integration?

Moderne agile Entwicklung erfordert im Allgemeinen ein anderes Herangehen an Pentests. Hier lautet die Devise, einzelne, bereits fertiggestellte Komponenten der Anwendung früh im Entwicklungszyklus zu testen, um mögliche Schwächen frühestmöglich zu erkennen und bei der weiteren Entwicklung ähnliche Probleme zu

vermeiden. Bei darauffolgenden Releases können dann einzelne abgeschlossene Features oder eine Summe von Änderungen auf Sicherheit getestet werden (Feature- bzw. Deltatests). Auch wenn die Schwachstellenabdeckung von automatisierten Securitytests nach wie vor sehr eingeschränkt ist, sollte doch so viel Automatismus zur Aufdeckung von Schwachstellen wie möglich in die Prozesskette eingebaut werden.

Beim agilen Entwicklungsvorgehen ist es nicht zuletzt aufgrund der schlechten Automatisierbarkeit von Pentests und der kurzgetakteten Releasezyklen besonders wichtig, sich aus der Abhängigkeit vom Pentest zu befreien. Siehe auch Frage 36.

Umfang und Preis eines Pentests

29 Was ist der Schutzbedarf?

Die Feststellung, dass eine Bank für das Testen ihres Online-Bankings mehr Aufwand spendieren sollte als der Anbieter einer App zur Parkplatzsuche, dürfte nicht verwundern. Aber an welchem Kriterium orientiert man sich dabei? Hier kommt der Schutzbedarf ins Spiel. Je höher der im Missbrauchsfall entstehende Schaden und je wahrscheinlicher ein Missbrauchsfall ist, desto höher ist der Schutzbedarf. Eine Anwendung mit hohem Schutzbedarf erfordert eine größere Testtiefe und einen größeren Testumfang - auf den Pentest bezogen kurz gesagt: intensiveres Suchen - als eine Anwendung mit niedrigem Schutzbedarf. Auch häufigeres Testen und der Einsatz weiterer Analysetechniken, wie insbesondere die Codeanalyse (siehe Frage 36), sind geeignete Mittel bei Anwendungen mit hohem Schutzbedarf.

Die Bestimmung des Schutzbedarfs ist dabei häufig nicht trivial. Neben den Auswirkungen auf die Geschäftsprozesse ist primär relevant, wie schützenswert die verarbeiteten Daten sind. Nicht zu unterschätzen ist auch, dass selbst dann, wenn Anwendung und Daten als eher begrenzt schützenswert gelten, der Vertrauensverlust (Imageschaden) durch einen Sicherheitsvorfall doch erheblich sein kann und im schlimmsten Fall das Geschäftsmodell gefährden könnte.

30 ... und warum spielt der Schutzbedarf eine Rolle?

Wegen der Kosten. Da Anwendungen nicht umfänglich von einer Maschine getestet werden können, hängen Testaufwand und Preis eng zusammen. Es gilt also, das Maß an Aufwand zu bestimmen, mit

dem bei minimalen Kosten eine akzeptable Risikominderung eintritt. Siehe den Begriff des Risikomanagements in Frage 1.

31 In welcher Einheit misst man den Schutzbedarf?

Die Metrik für die Einstufung des Schutzbedarfs ist nicht klar definiert. Sie besteht aus in der Regel drei bis fünf Stufen. Zum Zwecke des Pentestens hat sich folgende 3-Stufung als praktikabel erwiesen: **normal**, **hoch** und **sehr hoch**. Streng genommen ist noch eine vierte Stufe im Spiel, nämlich die mit gar keinem Schutzbedarf. Da bei solchen Anwendungen ein Pentest nicht erforderlich ist, benötigen wir sie hier auch nicht.

Als **hoch** bewertet man den Schutzbedarf einer Anwendung dann, wenn eine Kompromittierung zu erheblichem und in der Regel nicht vertretbarem Schaden führt. Hätte der Schaden gar existenzbedrohende Ausmaße, so wäre der Schutzbedarf als **sehr hoch** einzustufen. Alle anderen schützenswerten Anwendungen werden der Stufe **normal** zugeordnet.

32 … und wie bestimme ich den Schutzbedarf meiner Anwendung?

Die Thema Schutzbedarfsfeststellung geht weit über die IT-Security hinaus. Es umfasst alle Unternehmensrisiken und ist eine Wissenschaft für sich, die ausgeklügelte Verfahren hervorgebracht hat. Große Unternehmen sind hier in der Regel gut aufgestellt und besitzen ein umfassendes Risikomanagement, in das der Schutzbedarf von Anwendungen und Daten eingebettet ist. Das Folgende wendet sich daher an den Kreis derjenigen, die so etwas nicht besitzen und den Schutzbedarf der zu testenden Anwendung pragmatisch und mit einfachen Mitteln bestimmen möchten.

Um zur Bewertung des Schutzbedarfs zu gelangen, müssen wir vom maximalen Schadensszenario ausgehen. Stellen Sie sich dazu Folgendes vor: Ein Angreifer ist in Ihre Anwendung bzw. Website eingedrungen und hat vollen Zugriff erlangt. Er ist in der Lage, beliebige Daten auszulesen oder sie in beliebiger Weise zu manipulieren. Er kann die Seiteninhalte verändern, ganz

offensichtliche oder nur schwer zu erkennende Falschinformationen hinterlegen oder Schadcode (Trojaner, Viren, Ransomware, etc.) einbauen, mit dem Ziel, die Benutzer, die Ihrer Website vertrauen, zu hintergehen. Er kann sich unerkannt in den Server einnisten und von dort aus weiteres Unwesen treiben. Denken Sie an den maximalen Schaden, der Ihnen aus der Kompromittierung der Anwendung entstehen kann.

Würden die Auswirkungen eines solchen Vorfalls auf die Geschäftstätigkeit Ihres Unternehmens ernsthafte Konsequenzen nach sich ziehen? Dann ist die Stufe **hoch** angemessen. Sind die Auswirkungen gar als geschäftsgefährdend zu werten, ist die Stufe **sehr hoch** zu wählen. In allen anderen Fällen würde der Schutzbedarf als **normal** eingeordnet werden.

Bei der Bewertung zu berücksichtigen ist insbesondere auch die Art der Daten. Sofern personenbezogene Daten wie Name, Emailadresse oder Adressdaten im Spiel sind, ist die Stufe **normal** in der Regel nicht mehr angemessen, sondern **hoch** die richtige Wahl. Dieser Umstand ergibt sich allein aus den juristischen Konsequenzen im (schlimmstenfalls fahrlässigen) Umgang mit personenbezogenen Daten. Sind außerdem hochsensible Daten wie Kreditkarten-, Zahlungs-, Kontoinformationen, Gesundheitsdaten, sonstige vertrauliche persönliche Daten, Geschäftsgeheimnisse oder sensible Finanzdaten des Unternehmens für die Anwendung zugreifbar, ist die Einordnung **sehr hoch** angemessen.

Ein besonderes Gefahrenpotenzial, das es zu beachten gilt, stellt die Auswirkung auf das Außenbild des Unternehmens - der **Imageschaden** - dar. Dieser ist, ungeachtet des tatsächlichen, möglicherweise eher geringen Schadens, häufig sehr hoch. Vereinfachende oder unsachgemäße Berichterstattung in der Presse oder das gezielte Aufbauschen durch Wettbewerber oder dem Unternehmen nicht wohlgesonnene Gruppen können den Imageschaden noch erheblich vergrößern.

Regel #5: Falls nicht bereits vorhanden, ist eine einfache Schutzbedarfsfeststellung nach dem angegebenen Verfahren durchzuführen. Im Zweifel lieber auf Nummer sicher gehen und die nächst höhere Stufe wählen.

33 Was hat darüber hinaus Auswirkungen auf den Preis?

Der zweite maßgebliche Faktor bei der Bestimmung des Testaufwands ist die Größe der Anwendung: Die Anzahl Formulare und Felder, die Länge von Dialogstrecken, die Menge an Funktionen, Anzahl und Art von Rollen und Rechten, usw. Da neben diesen Parametern meist weitere zu berücksichtigen sind - wie die Komplexität von Dialogabläufen und Use-Cases - spricht man statt von Größe zumeist von der **Komplexität** der Anwendung.

Häufig kommt ein dritter Faktor hinzu, nämlich dann, wenn der Blick auf den von außen sichtbaren und erreichbaren Teil der Anwendung zur Prüfung der Sicherheit eines Systems nicht ausreicht. Etwa der Fall, dass Dateien per Upload von der Anwendung entgegengenommen und intern weiterverarbeitet werden, ohne dass die Ergebnisse zurück in die Webanwendung gelangen. Es ist also zu klären, ob derartige Bereiche mitzutesten sind, d.h. der Testscope ist zu definieren. Dieser ist besonders dann, wenn Aspekte ausgeschlossen worden sind, im abschließenden Bericht exakt zu dokumentieren, damit deutlich wird, dass das Pentestergebnis nur eine Teilaussage liefert.

Der zu veranschlagende Testaufwand, und damit der Preis, ergibt sich also grob aus dem Produkt von Schutzbedarf und Komplexität der Anwendung, unter Berücksichtigung des Testscopes.

Es gibt übrigens eine natürliche Grenze für die Anwendbarkeit des Mittels Penetrationstest. Ab einem bestimmten Komplexitätsgrad kann der Penetrationstest nicht mehr umfassend angewendet werden. Dann kommt er höchstens partiell, in Ergänzung zu Mitteln wie der statischen Codeanalyse (Frage 36), in Betracht.

34 Wie wird der finale Testaufwand ermittelt?

Parameter 1, der Schutzbedarf, wird vom Betreiber der Anwendung beigesteuert und vom Dienstleister sinnvollerweise noch einmal hinterfragt. Parameter 2, die Komplexität, gilt es als wesentlichen Input für die Angebotserstellung durch den Dienstleister zu ermitteln. Dazu stehen verschiedene Verfahren zur Verfügung, die hier mit ihren Vor- und Nachteilen aufgeführt sind:

#	Verfahren	Vorteile	Nachteile
1	**Selbstinspektion** Wenn die Anwendung bereits existiert und alle zu testenden Bereiche von außen zugänglich sind, wird dem Pentestdienstleister ein oder - bei unterschiedlichen Rollen - mehrere Testuser zur Verfügung gestellt. Der Anbieter klickt sich durch die Anwendung und verschafft sich selbstständig einen Überblick.	• Geringer Aufwand für den Auftraggeber • Hohe Genauigkeit	• Nur möglich, wenn Anwendung für den Tester (von außen) zugreifbar
2	**Walk-Through** Es wird ein Web-Meeting aufgesetzt und der Auftraggeber führt den Pentestdienstleister durch die Anwendung.	• Hohe Genauigkeit • Ermöglicht Rückfragen und erleichtert die Abstimmung	• Mit (moderatem) Aufwand für den Auftraggeber verbunden
3	**Auf Dokumentenbasis** Insbesondere dann, wenn die Anwendung noch nicht fertig ist, besteht die Möglichkeit der Beurteilung auf Basis von Spezifikationen, Mockups, Funktionsbeschreibungen.	• Geringer Aufwand	• Geringe Genauigkeit
4	**Am Beispiel** Wenn es sich bei der Anwendung um einen "typischen" Vertreter eines Anwendungsbereichs (z.B. Shop, Mitarbeiterportal, Diskussionsforum, etc.) handelt, dann kann die Abschätzung auf Basis der Nennung des Anwendungsbereichs plus Beantwortung von Fragen erfolgen.	• Minimaler Aufwand für beide Seiten • Schnell	• Fehleranfällig wegen des Potential für Missverständnisse in der Kommunikation zwischen Auftraggeber und Auftragnehmer

Egal welches Verfahren gewählt wird, die möglichst vollständige Übermittlung der in Frage 10 genannten Informationen trägt zu einem reibungslosen Ablauf und einer treffsicheren Abschätzung bei.

Regel #6: Man unterstütze bereits die Angebotserstellung durch Bereitstellung umfassender und verlässlicher Informationen.

35 Ist der Pentest als Mittel zur Herstellung der Sicherheit ausreichend?

Die Antwort auf diese Frage lässt sich am besten mit dieser Praxiserfahrung illustrieren: Bei Anwendungen, die ohne Beachtung des Qualitätsmerkmals Sicherheit entwickelt wurden - etwa, weil man annimmt, man könnte die Sicherheit anhand der Pentest-Ergebnisse nachträglich aufsetzen - liefert der Pentest erwartungsgemäß eine erkleckliche Menge an Schwachstellen. Das eigentlich Bedenkliche ist aber Folgendes: Bei solchen Anwendungen zeigt der Nachtest nach dem Beheben der Schwachstellen unverhältnismäßig oft, dass die berichteten Schwachstellen nicht richtig beseitigt worden sind. Und spätere Pentests auf Folgereleases derselben Anwendung zeigen zumeist das erneute Auftreten bereits zuvor berichteter Schwachstellen, nur vielleicht an anderen Stellen oder in einem etwas anderen Gewand.

Unterm Strich: Wird versucht, eine Anwendung - quasi im Nachgang zur Entwicklung - über einen Pentest und die anschließende Behebung der darin ermittelten Probleme sicher zu machen, wird man es schwer haben, deren Sicherheit jemals richtig in den Griff zu bekommen. Man wird über den gesamten Lebenszyklus der Anwendung mit ungeplanten Aufwänden sowie damit einhergehenden Zusatzkosten und Verzögerungen zu kämpfen haben. Stattdessen ist Qualitätsmerkmal Sicherheit im Entwicklungszyklus zu verankern (Frage 36). Es gilt:

Regel #7: Die Rolle des Pentests ist es nicht, die Sicherheit einer Anwendung herzustellen. Er dient dazu festzustellen, ob ein ausreichendes Maß an Sicherheit gegeben ist.

36 Was sollte also sonst noch für die Sicherheit getan werden?

Diese Frage geht über eine Pentest FAQ deutlich hinaus, deshalb seien die wichtigsten Maßnahmen hier nur kurz aufgeführt. Eine Anlaufstelle für umfassende Informationen zur Application Security ist die Website des "Open Web Application Security Project" (OWASP, www.owasp.org).

Befindet man sich noch am Anfang eines neuen Softwareprojekts, ist man in der glücklichen Lage, folgende äußert wirksame Maßnahmen durchführen zu können:

- Entwickler schulen: Seminare zur Einführung in die Web Application Security und Secure Coding Trainings sollten für alle Teammitglieder selbstverständliche Grundlage sein.
- Security-Architektur-Workshop durchführen: Die ersten Überlegungen zur System- und Softwarearchitektur der Anwendung sind mit einem Application-Security-Spezialisten hinsichtlich der Berücksichtigung von Sicherheitsaspekten abzugleichen.
- Building-Security-In: In den Entwicklungsprozess von Beginn an das Qualitätsmerkmal Sicherheit mit aufnehmen.
- Codeinspektionen auf Sicherheit zu einer entwicklungsbegleitenden Maßnahme machen.
- Insbesondere beim Agilen Vorgehensmodell und in Continuous Integration-Umgebungen die Build Chain um Tools zum automatisierten Securitytesten erweitern, soweit der Stand der Technik das zulässt.

Bei bereits bestehenden Anwendungen:

- Architekturanalyse: Analyse der Einbettung der Anwendung in die umgebende Systemlandschaft, Kommunikationsbeziehungen, Vertrauensübergänge etc.
- Statische Codeanalyse in Form einer manuellen Codeinspektion besonders sicherheitsrelevanter Codebereiche oder als umfassende Codeanalyse mithilfe kommerzieller SAST-Tools.

37 Reicht es, nur Internet-Anwendungen zu testen, oder sollen auch Intranet-Anwendungen getestet werden?

Die Antwort aus dem Munde des Sicherheitsvertreters würde lauten, dass selbstverständlich auch alle schutzbedürftigen Intranet-Anwendungen zu testen sind. Der Pragmatiker hingegen wird den Standpunkt vertreten, dass dies vom Vertrauensstatus der jeweiligen Umgebung abhängig gemacht werden sollte und dass selbstverständlich das Internet einen niedrigeren oder gar keinen Vertrauensstatus besitzt. Die Einordnung unterliegt also Sicherheitsprinzipien einer übergeordneten Ebene, eine allgemeingültige Antwort kann nicht gegeben werden.

Was es aber zu bedenken gilt, ist Folgendes: Viele Anwendungen, die einst als reine Intranet-Anwendungen - und ohne besondere Sicherheitsanforderungen - begonnen haben, erfahren irgendwann eine Ausweitung der Benutzergruppe, von den eigenen Mitarbeitern hin zu externen Partnern und werden dann für den Zugriff über das VPN und schließlich zur weiteren Vereinfachung auch zum Zugriff über das Internet geöffnet. Geschieht dies ohne Einführung rigider Security- und Testmaßnahmen, ergeben sich hohe Risiken.

38 Wie kann ich zur Qualitätsmaximierung des Pentests beitragen?

Die Qualität eines Pentests ist nicht allein von der Qualität des Pentesters abhängig. Gute Organisation und Kommunikation sowie ein reibungsloser Ablauf haben ebenfalls großen Einfluss. Der Auftraggeber kann erheblich zur Qualitätsmaximierung beitragen, indem er diese Fehler vermeidet:

- Verspätete oder unvollständige Bereitstellung der Anwendung zum vereinbarten Starttermin
- Nicht funktionierende oder nicht mit den erforderlichen Rechten ausgestattete Test-User
- Zugriffsprobleme durch behindernde Firewall-Einstellungen (So ist vorher zu prüfen, ob der Zugriff auch *von außen* funktioniert!)
- Ausfälle der Anwendung oder einzelner Teile während der Testdurchführung

- Parallel laufende Tests oder Deployments der Anwendung, die zu Datenänderungen oder Änderungen am Funktionsumfang führen. (Diese ziehen dem Tester gewissermaßen den Boden unter den Füßen weg, da seine Referenzpunkte nun nicht mehr stimmen.)
- Nicht-Verfügbarkeit eines auskunftsfähigen Ansprechpartners.
- Fehlende oder unvollständige Test- bzw. Beispieldaten in der Anwendung.
- Fehlende oder unvollständige Beschreibung von API-Aufrufen

Regel #8: Durch Schaffung möglichst idealer Rahmenbedingungen kann der Auftraggeber einen erheblichen Beitrag zur Maximierung der Quote aufgedeckter Schwachstellen leisten.

Umgang mit dem Ergebnisbericht

39 Wieviel Zeit muss für das Beheben von Schwachstellen eingeplant werden?

Ein weit verbreiteter Fehler ist, dass zwischen Ende des Pentests und Termin der Produktivschaltung zu wenig Zeit eingeplant wird. Werden Schwachstellen gefunden, ist für eine gründliche Behebung nicht mehr ausreichend Zeit und die Anwendung geht entweder "mit heißer Nadel" gepatcht (siehe Frage 41) oder unter Inkaufnahme noch nicht behobener Schwachstellen an den Start. Kritische Schwachstellen können dann in letzter Minute die gesamte Planung über den Haufen werfen.

Regel #9: Man gehe davon aus, dass ein Pentest Schwachstellen findet und plane daher ausreichend Zeit für die Behebung ein.

40 Wie gehe ich mit dem Pentestbericht um?

Wichtig ist, dass die gefundenen Schwachstellen sowohl hinsichtlich ihrer Auswirkungen als auch ihrer Ursache gut verstanden werden. Ist dies nicht der Fall, so sollte der Pentester zwecks Unterstützung kontaktiert werden. (Der Pentestdienstleister sollte diesen Service mit anbieten).

Weitere Tipps für einen pragmatischen Umgang mit den Ergebnissen:

- Generell sollten die Bewertungen noch einmal nachvollzogen werden, da aus der externen Pentestersicht der Kontext von Schwachstellen in der tatsächlichen Umgebung manchmal schwer einzuschätzen ist (siehe Fragen 18 und 19).
- Als kritisch eingestufte Schwachstellen sehr ernst nehmen und sofort handeln, siehe Frage 18!

- Die von hoch bis niedrig eingestuften Schwachstellen nach absteigendem Gefahrenpotenzial priorisiert behandeln, d.h. den Aufwand - also das zur Behebung zur Verfügung stehende Budget und die verfügbare Zeit - zum großen Teil für die schwerwiegenden Schwachstellen und mit entsprechend abnehmender Priorität für die restlichen Schwachstellen verwenden.
- Es sollte nicht außer Acht gelassen werden, dass ein Pentest manchmal nur Anzeichen auf eine Schwachstelle findet, diese jedoch nicht nachweisen kann. Oft erfolgt dann im Bericht eine Bewertung mit einem niedrigeren Gefahrenpotenzial, ergänzt um eine entsprechende Bemerkung. Weist ein Finding solche Anmerkungen auf, sollte ihm definitiv nachgegangen werden, um auszuschließen, dass das Risiko unterbewertet worden ist.

Zu Bedenken ist auch die andere Seite der Medallie: Sicherheit ist kein Selbstzweck! Der Vorteil höherer Sicherheit bringt oft die Inkaufnahme von Nachteilen mit sich. Bevor man in Aktionismus verfällt sollte eine Schwachstelle, deren Behebung Seiteneffekte nach sich ziehen würde, einer nüchternen Kosten-Nutzen-Abwägung unterzogen werden.

41 Was ist beim Beheben einer gefundenen Schwachstelle zu beachten?

Ein schneller Fix, der die berichtete Schwachstelle zum Verschwinden bringt, ist in aller Regel nicht ausreichend. Wie in Frage 13 dargestellt, ist das Ziel eines Pentests nicht, alle Stellen zu finden, an denen eine Schwachstelle auftritt, sondern lediglich das Vorhandensein der Schwachstelle nachzuweisen und im Bericht beispielhaft an einem konkreten Auftreten zu belegen. Es gilt daher, das Problem grundlegend und dauerhaft zu lösen und nach dieser Regel zu verfahren:

Regel #10: Es ist nach der Wurzel des Problems zu suchen und die Schwachstelle dort zu beheben.

Wird lediglich das Symptom der Schwachstelle an der Stelle des Auftretens behoben, so wird dieselbe Schwachstelle an anderer Stelle womöglich weiter vorhanden sein. Oder man selbst oder die Kollegen, die am nächsten Release entwickeln, bauen dieselbe Schwachstelle erneut ein.

Zu guter Letzt

42 OWASP Top 10, OWASP Testing Guide, ASVS, CWE - welcher Pentest ist der richtige für mich?

Es existieren verschiedene Schwachstellensammlungen, an denen sich Pentests von Webanwendungen orientieren. Zu nennen wären insbesondere diese:

Die **OWASP Top 10**[1] sind die Zusammenstellung der "10 verbreitetsten Risiken für Webanwendungen". Sie sind und als Beitrag zur Bewusstseinbildung im Bereich der Webanwendungen gedacht gewesen. Die schnell erlangte Popularität hat Securityanbieter veranlasst, ihre Tools und Dienstleistungen - oft fälschlicherweise - mit der Eigenschaft "deckt die OWASP Top 10 ab" zu bewerben. Was dazu geführt hat, dass die OWASP Top 10 heute vielfach als Standard dafür verstanden werden, was ein Pentest oder eine Sicherheitsanalyse alles abprüfen sollte. Dem können sie nicht gerecht werden:

- Die meisten der aufgenommenen Risiken sind recht allgemein und beispielhaft benannt und lassen somit hinsichtlich konkreter Tests viel Interpretationsspielraum. Das macht sie als Vertragsgrundlage nicht besonders brauchbar.
- Es werden nur die technischen Bereiche erfasst, jedoch nicht die Geschäftslogik (bezogen auf die Systematik in Frage 17: Die Ebenen 4 und 5 werden nicht berücksichtigt)
- Wichtige technische Schwachstellen (Beispiel: Cross-Site Request Forgery (CSRF)) sind nicht enthalten
- Eine der Regeln (A10, das Logging betreffend) lässt sich mit dem Mittel des Penetrationstests im Allgemeinfall gar nicht abprüfen.

Zusammengefasst: Die auftraggeberseitige Anforderung "bitte einen Pentest nach OWASP Top 10" oder das auftragnehmerseitige Angebot

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

"wir testen gemäß OWASP Top 10" sind beide sehr schwammig und in der Regel, auch bei weiter Auslegung, unzureichend. Es sollte genauer ausspezifiziert und um zusätzliche Tests ergänzt werden.

Der **OWASP Testing Guide (OTG)**[2] ist eine umfangreiche Sammlung von Schwachstellen und eine Anleitung, wie diese aufzufinden sind. Auch der OTG hat nicht das Ziel der Verwendung als Schwachstellenkatalog. Vielmehr richtet er sich mit der Beschreibung von Penetesttechniken in erster Linie an (angehende) Pentester. Aufgrund des sehr viel höheren Vollständigkeits- und Detaillierungsgrades eignet er sich dennoch weit besser als Grundlage für eine Beauftragung als die OWASP Top 10. Seine größte Schwäche ist die Tatsache, dass er in der aktuellen Version 4 von 2014 schon sehr betagt ist und damit neuere Angreifbarkeiten nicht enthält.

Der **OWASP Application Security Verification Standard (ASVS)**[3] hat den Anspruch eines umfassenden Frameworks von Sicherheitsanforderungen und -maßnahmen (**Security Controls**). Er deckt funktionale und nicht-funktionale Anforderungen ab und erstreckt sich nicht nur auf das Testen, sondern auch den Entwurf und die Entwicklung von Anwendungen. Seit dem ersten Erscheinen im Jahr 2009 ist das Framework unter Einbeziehung von Erfahrungen aus Nutzerkreisen mehrfach signifikant überarbeitet worden und liegt mittlerweile in einer entsprechend ausgereiften Version 4 vor.

ASVS legt drei Level der Strenge der Prüfungen fest. Wobei Level 1 in etwa dem Schutzbedarf *normal*, Level 2 dem Schutzbedarf *hoch* und Level 3 dem Schutzbedarf *sehr hoch* gemäß der in Frage 31 behandelten Nomenklatur entspricht.

Nur der Level 1 kann alleine mit Pentests abgedeckt werden, die beiden anderen Level enthalten Prüfungen, die durch das Betrachten "von außen" nicht geleistet werden können bzw. verlangen sogar ganz explizit weitere Prüfverfahren.

---

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

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

Die Vorgabe eines "Pentests nach ASVS 4 Level 1" ist also für viele Anwendungen nicht ausreichend. Und die Vorgabe "Pentest nach ASVS Level 2" ist in sich widersprüchlich, da Level 2 nicht alleine mit Pentests abgedeckt werden kann. Auch die Präzisierung hinsichtlich Letzterem, nur die mittels Pentest durchführbaren Prüfungen aus Level 2 vorzunehmen, ist nicht aussagekräftig, da diese Prüfungen nicht eindeutig identifizierbar sind, weil sie auf andere Prüfmethoden als das Pentesten zugeschnitten sind.

Bei der **Common Weakness Enumeration (CWE)**[4] handelt es sich um eine laufend gepflegten Liste von Application Security Schwachstellen. Ihr Anspruch ist, Messlatte für die Abdeckung von Securitytools und Baseline für Schwachstellenprüfung und Sicherheitsmaßnahmen zu sein. Die Liste ist sehr detailliert, gegenwärtig über 800 Einträge. Die Schwachstellenbeschreibungen geben aber keine Informationen darüber, ob eine Schwachstelle mit einem Pentest geprüft werden kann. Daher ist sie zur scharfen Definition des Umfangs eines Pentests ebenfalls nicht gut geeignet. Dennoch ist die Vorgabe an den Pentester, den CWE weitest möglich abzudecken, von allen hier genannten Methoden die genaueste.

Letztlich muss also festgestellt werden, dass es einen (umfassend anwendbaren) Penetrationsteststandard, auf den sich Bezug nehmen lässt, um Angebote zu vergleichen und Penetrationstests zu beauftragen, nicht gibt. Sofern nicht Unternehmensrichtlinien diesbezüglich explizite Vorgaben machen, sollte das Vertrauen in den Pentestdienstleister nach ausführlicher Rücksprache den Ausschlag für Testvorgehen und Testumfang geben.

43 Wie finde ich den richtigen Pentester für meine Anwendung?

Einen für alle Situationen und Anforderungen passenden Standard, anhand dessen die Formulierung von Anforderungen an den Penetrationstester leicht fällt, gibt es nicht (siehe Frage 42). Um den geeigneten Ansatz für die eigenen Belange zu finden, sollte man sich von entsprechenden Experten beraten lassen.

---

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

Tipps zur Auswahl eines zuverlässigen Anbieters:

- Fragen Sie danach, wie lange der Anbieter schon Application Security Pentests durchführt / wie viele Anwendungen er getestet hat / wieviel Erfahrungen er mit Unternehmen und Anwendungen Ihrer Größe hat / für welche (ähnliche) Unternehmen er schon Penetrationstests durchgeführt hat.
- Fragen Sie danach, welche Teststandards der Anbieter zugrunde legt und wie er sicherstellt, dass der Testumfang der technischen Entwicklung laufend angepasst wird.
- Verlangen Sie, dass der Test auch von der oder den Personen durchgeführt wird, die die entsprechende Erfahrung besitzen.
- Lassen Sie sich eine ausführliche Beschreibung seines Vorgehens und einen Beispielbericht zeigen.
- Vergleichen Sie mithilfe dieser Informationen mehrere Anbieter.

Haben Sie Fragen oder Feedback?

Wir passen diese FAQ laufend an. Ziel ist es, Sie als Auftraggeber möglichst umfassend über alle Sie betreffenden Aspekte des Pentestens zu informieren.

Bitte schicken Sie uns Ihre Fragen und Verbesserungswünsche. Wir geben direkte Antwort und lassen diese in die FAQ einfließen: office@mgm-sp.com oder https://www.mgm-sp.com/kontakt/