Eine Sonderveröffentlichung der Heise Zeitschriften Verlag GmbH & Co. KG

iX extra

Security

Sicherheit für Webanwendungen

Veranstaltungen

25. – 26. September, Konstanz
D.A.C.H. Security
www.syssec.at/dachsecurity2012

16. – 18. Oktober, Nürnberg
it-sa
www.it-sa.de

23. – 24. Oktober, Brüssel
ISSE
www.isse.eu.com

6. – 7. November, Berlin/
12. – 13. Dezember, München
iX-Workshop: iPhone & iPad Security
www.ix-konferenz.de

iX extra
Security zum Nachschlagen:
www.heise.de/ix/extra/security.shtml

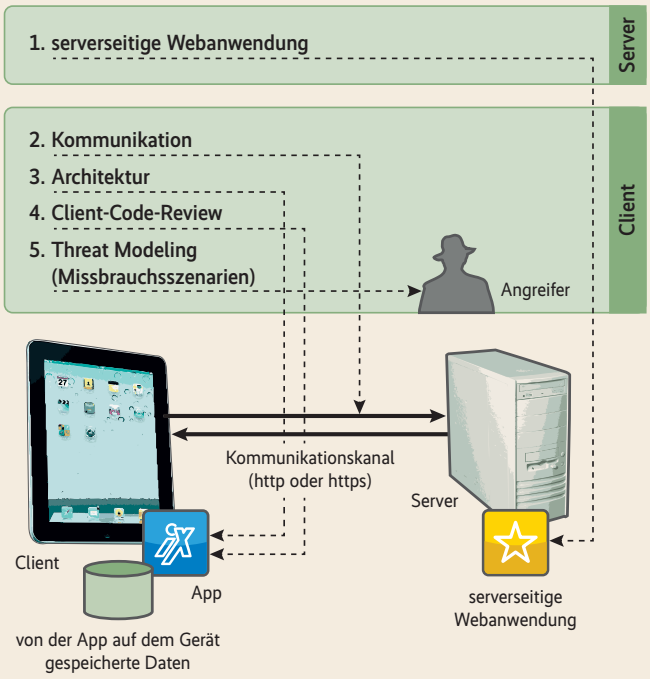
Von Anfang an

Sicherheit bei Webapplikationen beginnt mit der Entwicklung

Webanwendungen, etwa Onlineshops, sind ein begehrtes Ziel für Cyberangriffe. Sicherheit für diese Applikationen muss ein Bestandteil der Softwarequalität sein, fordern die Experten. Neben dem Ergreifen technischer Maßnahmen wie Web Application Firewalls lässt sich die Qualität nur über Schulungen, Leitfäden und Risikoanalysen erreichen.

In den frühen Zeiten des Web waren es lückenhaft konfigurierte Unternehmens-Firewalls oder unerkannte Fehler in diesen Systemen, über die Angreifer an interne Systeme und ihre Daten gelangten. Heute sind es die Webanwendungen, die das Haupteinfallstor bilden. Zumeist sind sie individuell entwickelt und damit bei Weitem nicht in dem Maße auf Fehler getestet wie Standardsoftware. Und sie besitzen entweder eine direkte Verbindung zu internen Datenschätzen oder aber beherbergen diese selbst.

Damit beispielsweise ein Kunde seine eigenen Bestellungen über das webbasierte Bestellsystem einsehen kann, muss die Anwendung Zugriff auf die Datenbank mit allen Bestelldaten haben. Der Kanal, über den Daten nach außen abfließen können, ist damit geschaffen. Es ist wichtig zu erkennen, dass die Abschottung dieser Daten durch eine elegante, grundlegende Maßnahme prinzipiell nicht möglich ist. Hat die Webanwendung einen entsprechenden Fehler (eine Sicherheitslücke), sind die Daten



Die Ansatzpunkte für Penetrationstests bei mobilen Apps ergeben sich aus den verschiedenen Ebenen der Angreifbarkeit.

ungeschützt. Da helfen weder Firewall noch SSL-Verschlüsselung.

OWASP gibt Schützenhilfe

Die Webanwendung muss ein unbemerktes Abfließen der Daten verhindern. Und genau hier liegt das Problem: Die Zahl der Möglichkeiten für Programmierfehler, die genau dies untergraben, ist gewaltig; das Risiko, dass die Anwendung ihrer Aufgabe hinsichtlich der Datensicherheit nicht korrekt nachkommt, damit entsprechend hoch.

Berühmt geworden ist die Top-10-Liste der OWASP (Open Web Application Security Project), jener offenen Community, die sich der Verbesserung der Sicherheit von Webanwendungen verschrieben hat. Darin sind die am meisten verbreiteten Schwachstellenkategorien aufgeführt. Penetrationstests, also im Wesentlichen die Analyse des Anwendungsverhaltens mittels systematisch durchgeführter Eingaben in die Formularfelder, zeigen dabei schon seit Jahren fast unverändert dieses Bild: Rund 80 Prozent aller Webanwendungen sind bezüglich der OWASP Top 10 nicht schwachstellenfrei und damit dem Risiko der Kompromittierung ausgesetzt.

Doch die Angriffsmöglichkeiten sind mit den OWASP Top 10 bei Weitem nicht ausgeschöpft. Immer stärker zeigt sich, dass Schaden auch entstehen kann, wenn die Verantwortlichen bei der Umsetzung eines Geschäftsprozesses in die Dialogabläufe der Webanwendung Missbrauchsszenarien nicht ausreichend betrachtet haben. Ganz typisch: Eine Anwendung erlaubt das Recherchieren in den Datenbeständen des Unternehmens, etwa die Suche nach dem passenden Teil in einem Teilekatalog. Das Szenario erfordert, dass das Suchen mit Platzhaltern erlaubt ist. Damit nicht ein böswilliger Nutzer den gesamten Datenbestand auslesen kann, was für das Unternehmen einen erheblichen Informationsverlust bedeuten würde, begrenzt die Anwendung die Zahl der Antwortdatensätze beispielsweise auf zehn.

Diese Maßnahme erweist sich jedoch als unzureichend. Ein Angreifer würde einfach sehr viele Aufrufe der Suchfunktion ausführen und dabei die Parameter systematisch variieren, etwa im Feld Teilebezeichnung nacheinander die Muster „Aa*“, „Ab*“, „Ac*“ und so weiter eingeben. Die Tools, mit denen auch der Laie eine solche Aktion bequem ausführen kann, sind im Internet frei verfügbar. Die Anwendung muss also zusätzlich sicherstellen, dass von ein- und demselben Client innerhalb einer bestimmten Zeitspanne nur eine äußerst begrenzte Anzahl an Aufrufen möglich ist – im Übrigen viel leichter gesagt als getan.

VERWUNDBARKEIT VON WEBANWENDUNGEN

Die folgende Klassifizierungsübersicht zeigt die fünf Ebenen, auf denen Webanwendungen verwundbar sein können.

	Ebene	Inhalt (Beispiele)
5	Semantik	**Schutz vor Täuschung und Betrug** Die Abläufe dürfen nicht aus dem Kontext gerissen und missbraucht werden können. Gilt insbesondere bei Einbindung von E-Mail.
4	Logik	**Absicherung von Prozessen und Workflows als Ganzes** - fehlende Anti-Automatisierung: beliebig häufiges Durchführen von Funktionen/systematisches Ermitteln von Inhalten - Umgehung von Schutzmaßnahmen
3	Implementierung	**Vermeiden von Programmierfehlern, die zu Schwachstellen führen** - Cross-Site Scripting - SQL-Injection
2	Technologie	**richtige Wahl und sicherer Einsatz von Technologie** - unverschlüsselte Übertragung vertraulicher Daten - Authentifizierungsverfahren, die nicht dem Schutzbedarf angemessen sind
1	System	**Absicherung der auf der Systemplattform eingesetzten Software** - Fehler in der Konfiguration von Webserver, CMS et cetera - „Known Vulnerabilities“ in den eingesetzten Softwareprodukten

Sicherheit ist nicht gestiegen

Leider kann man nicht davon ausgehen, dass die Möglichkeiten, Fehler zu machen, mit der über die Zeit zunehmenden Reife der Webtechnologie signifikant abgenommen hätten:
– HTTP in der Version 1.1 hat sich seit seinem Erscheinen vor über 10 Jahren nicht substanziell weiterentwickelt und beinhaltet weiterhin viele Defizite, die die Webanwendungen ausräumen müssen. Webtechniker kennen beispielsweise die Unzulänglichkeiten des Cookie-Konzepts oder die Klimmzüge, die das Verhindern von Session-Riding-Angriffen (alias Cross-Site Request Forgery) erfordert.
– Programmier-Frameworks nehmen Sicherheit bringende Konzepte nur schleppend auf. So ist es beispielsweise bei .NET noch immer mit einigen Anstrengungen verbunden, die Schwachstelle der Session Fixation, über die ein Eindringen in fremde Benutzerkonten möglich ist, auszuräumen.
– Den meisten Entwicklern mangelt es nach wie vor an Verständnis für die Sicherheit von Webanwendungen.
– Die Erkenntnis, dass die geforderte Sicherheit bei Webanwendungen mit ausreichend Budget umgesetzt und schließlich geprüft werden muss, ist zu vielen Verantwortlichen noch nicht durchgedrungen.

Hinzu kommt, dass sich die Angriffsfelder durch neue Technologien ausgeweitet haben, etwa durch mobile Webanwendungen, die weitere Schwachstellen mitbringen (siehe Kasten „Verwundbarkeit von Webanwendungen“), oder durch den kommenden Standard HTML5, der eine deutliche Erweiterung des Funktionsumfangs im Browser mit sich bringt.

Maßnahmen bereits im Budget einplanen

Sicherheit für Webanwendungen, Web Services und Mobile Apps kommt nicht von alleine, sondern muss handfest angepackt werden. Dazu muss sich als Allererstes die Erkenntnis durchsetzen, dass Sicherheit zu den Softwarequalitätsmerkmalen gehört und ihr ein hoher Stellenwert zuzuordnen ist. Sie gehört als eigener Punkt insbesondere auch in die Budgetplanung hinein.

Vor allem folgende Maßnahmen haben sich als wirksam und sinnvoll für die ersten Schritte erwiesen:

Bei Management, Anwendungsverantwortlichen und Umsetzern muss das Bewusstsein für die Gefährdung der Daten durch unzureichend abgesicherte Webanwendungen hergestellt werden. Eine Awareness-Veranstaltung, bei der man sich an „lebenden“ Beispielen, möglicherweise an einer eigenen Webanwendung, von einem Experten zeigen lässt, wie leicht das Eindringen ist, wirkt hier Wunder. Dabei sollte der Vorführende auch das Spektrum der Gegenmaßnahmen und individuelle Lösungsansätze zeigen.

Sicherheit muss in die Budgets für die Umsetzung von Geschäftsprozessen in Webanwendungen als integraler Bestandteil aufgenommen werden. Bei der Umsetzung durch externe Dienstleister muss das Unternehmen sie entsprechend verpflichten.

Entwickler schulen und sensibilisieren

Die Erfahrung zeigt, dass ein Softwareentwickler, der schon

einmal erlebt hat, was man in Sachen Sicherheit alles falsch machen kann und mit welchen Konzepten man diese Fehler vermeidet, hochmotiviert ist, seine Anwendung sicher zu entwickeln. Die Maßnahme „Schulung der Entwickler im sicheren Programmieren“ sollte daher bei jedem Unternehmen, das seine Webanwendungen selbst entwickelt, ganz vorne stehen.

Ist das Wissen in der Softwareentwicklung angekommen, sollten die Verantwortlichen die Erstellung von Secure Coding Guidelines in Angriff nehmen. Dabei können Entwickler sich der generellen Guidelines der OWASP bedienen und diese an die eigene Umgebung und die eigenen Erfordernisse anpassen. Eine Unterstützung durch entsprechende Dienstleister ist hier besonders sinnvoll, da erfahrungsgemäß dem Web-Application-Security-Neuling die nötige Sicherheit bei der Beurteilung der Wirksamkeit noch fehlt.

Mit der Festschreibung von Secure Coding Guidelines sollte das Einrichten von Security Libraries einhergehen. Auch hier können Entwickler, soweit es geht, auf bestehende Bibliotheken wie die der OWASP Enterprise Security API zurückgreifen und die einzelnen Sicherheitsfunktionen den Coding Guidelines zuordnen. Coding Guidelines und Security Libraries sind Mittel, die die Fachleute über die Zeit weiterentwickeln und verfeinern sollten, insofern ist mit der Einführung ein entsprechender Verbesserungsprozess anzustoßen.

Idealerweise sollte es spezialisierte Ansprechpartner als „Interessenvertreter“ der Web Application Security und zentrale Anlaufstelle in einer Organisation geben, die an der Gestaltung und Umsetzung von Prozessen maßgeblich beteiligt sind.

Von Vorteil ist es, eine zusätzliche Sicherungslinie einzurichten. Web Application Firewalls (WAFs) sind Geräte, die vor dem Webserver positioniert werden und auf HTTP-Ebene schädliche Daten filtern.

ANBIETER VON SCHWACHSTELLENSCANNERN UND TEST-TOOLS

Hersteller	Produkt	Website
Acunetix	AcuSensor	www.acunetix.com
Cenzic	Hailstorm	www.cenzic.com
DBAPPSecurity	MatriXay 3.0	www.dbappsecurity.com/webscan.html
eEye Digital Security	Retina Web	www.eeye.com
GamaSec	GamaScan	www.gamasec.com/Gamascan.aspx
Greenbone	Greenbone Security Manager	www.greenbone.net
Hewlett-Packard	Webinspect	www.hpenterprisesecurity.com/products/hp-fortify-software-security-center/hp-webinspect-real-time
IBM	Rational AppScan	www-01.ibm.com/software/awdtools/appscan
Mavituna Security	Netsparker	www.mavitunasecurity.com
nCircle	WebApp360	www.ncircle.com
N-Stalker	N-Stalker	www.nstalker.com
NT OBJECTives	NTO Spider	www.ntobjectives.com
Outpost24	Web Application Scanning	www.outpost24.com/web-application-scanning
PortSwigger	Burp Suite Professional	www.portswigger.net/burp/
Qualys	QualysGuard	www.qualys.de
Rapid7	nexpose	www.rapid7.com
Sec4app	WebCruiser	sec4app.com
Syhunt	Sandcat	www.syhunt.com
Tenable	Nessus	www.nessus.org
WhiteHat Security	Sentinel PL	www.whitehatsec.com
Open Source		
Andres Riancho	w3af	w3af.sourceforge.net
Andreas Schmidt	WATABO	sourceforge.net/apps/mediawiki/watobo/index.php?title=Main_Pagesourceforge.net/apps
BrotherSoft	VulnDetector	www.brothersoft.com/vulndetector-344655.html
Byrne/Duprey	Grendel Scan	www.securitytube.net/video/4422
Casaba	Watcher	websecuritytool.codeplex.com
CIRT	Nikto	www.cirt.net
Ethical Hacking	XSSploit	www.ehacking.net/2011/02/xss-vulnerability-scanner.html
GitHub	arachni	arachni.segfault.gr
Google	Skipfish	code.google.com/p/skipfish/
Nicolas Surribas	Wapiti	wapiti.sourceforge.net/
Oedipus	Oedipus	webscripts.softpedia.com/script/Security-Systems/Oedipus-Web-App-Vulnerability-Scanner-18443.html
OWASP	OWASP ZAP	https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
Paros	Paros Proxy	www.parosproxy.org
PortSwigger	Burp Suite	portswigger.net
Rfp.labs	libwhisker	sourceforge.net/projects/whisker
SensePost	Wikto	www.sensepost.com/labs/tools/pentest/wikto
Websecurify	Websecurify	www.websecurify.com

Die Übersicht erhebt keinen Anspruch auf Vollständigkeit.

Hier gibt es etwa die frei verfügbare WAF ModSecurity (als Plug-in für den Apache Webserver); sie hat Industriequalität und beinhaltet einen leistungsfähigen Core Rule Set (generischer Filtersatz) für einen durchaus sinnvollen Grundschutz. WAFs dürfen aber nicht als Ersatz für die an der Ursache ansetzenden Maßnahmen dienen – sie sind und bleiben eine zusätzliche Sicherungslinie.

Viele Beteiligte, viele Perspektiven

Als argloser Mensch denkt man bei der Umsetzung einer Webanwendung zumeist nicht an Missbrauch – und stattet die Anwendung entsprechend schlecht mit Abwehrmechanismen aus. Deshalb sollte das Konzipieren einer Webanwendung standardmäßig von einer systematischen Risiko- und Bedrohungsanalyse begleitet werden. Hier konzentrieren sich möglichst alle Beteiligten darauf, herauszufinden, welche Interessenlagen zu welchen Missbrauchsgefahren führen können, um diesen dann im Anwendungsdesign zu begegnen.

Webanwendungen müssen abschließend vor der Freigabe auf ihre Sicherheit hin überprüft werden. Dazu gehören entsprechende Prozesse einschließlich Einplanen der nötigen Zeit für die Prüfung selbst und die Behebung gefundener Probleme, Festlegung von Freigabekriterien (ab wann ist eine Webanwendung hinreichend sicher?) und Umgang mit verbleibenden Restrisiken.

Viele sehen in der Testphase den Penetrationstest als Mittel der Wahl an. Er hat jedoch meist keine dauerhafte Wirkung auf die Sicherheit, denn die Ergebnisse des Tests legen nicht die Wurzel des Übels offen, sondern nur dessen Symptome. Deshalb empfiehlt es sich zu prüfen, ob nicht das – zumeist deutlich aufwendigere – Mittel der statischen Source-Code-Analyse wirtschaftlich angewendet werden kann. Dieses zeigt für den

Programmierer leicht nachvollziehbare Mängel sowie die Ansätze zur Behebung auf und legt den Grundstein für eine an den Ursachen ansetzende Behebung des jeweiligen Problems. Mit der Zeit lässt sich diese Art der automatisierten Unterstützung immer weiter hin zum Entwickler verschieben, sodass Sicherheit nicht erst über die abschließende Prüfung, sondern bereits bei der Programmierung berücksichtigt wird. Das Mittel des Penetrationstests ist zu diesem fundamentalen Schritt nicht in der Lage.

Dass Sicherheit von Software ein stetiger Verbesserungsprozess ist, müssen alle beteiligten Stellen erkennen und verankern. Die Entwicklung geeigneter, auf die Unternehmensorganisation zugeschnittener Metriken, die in ein entsprechend etabliertes Reporting einfließen, stellen in Verbindung mit einem wirksamen Anreizsystem hierbei die Kernelemente dar. Bei der Umsetzung helfen spezielle, auf die Belange der abgestimmte Softwaresicherheit zugeschnittene Vorgehens- und Reifegradmodelle, beispielsweise das frei verfügbare Software Assurance Maturity Model (OpenSAMM, www.opensamm.org).

Sicherheit für mobile Webanwendungen

Schließlich geht es noch um das Testen der Sicherheit mobiler Webanwendungen (siehe Abbildung). Um einem allgemeinen Missverständnis gleich zuvorzukommen: Der Hauptunsicherheitsfaktor bei den meisten Apps ist nicht die App selbst, sondern – ebenso wie bei den browserbasierten Webanwendungen – die Serverseite. So erstreckt sich die Sicherheitsprüfung in erster Linie auf die Analyse der Kommunikation zwischen App und Server (Schritt 2 im Bild) und die darüber zu ermittelnden Eintrittspunkte in die Webanwendung, die dann mit Mitteln des Web-Application-Security-Penetrationstests auf Schwachstellen geprüft wird (Schritt 1 im Bild). Der potenzielle Missbrauch besteht dabei nicht im Ausspähen der Kommunikation durch den Angreifer, sondern vielmehr in der Möglichkeit, dass der Angreifer die betreffende App auf seinem eigenen Handy installiert und mit entsprechenden Tools die App-Zugriffe sichtbar und manipulierbar macht.

Die App unter die Lupe nehmen

Die App selbst wird in den Schritten 3 und 4 betrachtet. Sind die Sicherheitsanforderungen an die App aufgrund ihres Funktionsumfangs, der Vertraulichkeit der Daten oder des Schutzbedarfs höher, so analysieren die Verantwortlichen zunächst die plattformspezifische Architektur (Schritt 3 im Bild). Dabei prüfen sie anhand der technischen Feinspezifikation (oder ähnlichen Beschreibungen), ob die Webanwendung die Security-Guidelines des Herstellers beachtet, das heißt der Anbieter die sicherheitsgebenden Plattformeigenschaften dem Schutzbedarf angemessen auf Daten und Funktionen anwendet.

Beispielsweise geht es darum, ob vertrauliche Daten in der Key Chain (iOS) ablegt werden. Da die Plattformen den Apps einen großen Teil der Client-Sicherheit abnehmen, ist diese Prüfung oftmals ausreichend – das Vorhandensein entsprechender Dokumente vorausgesetzt. Besonders heikle Apps oder solche, die ihre eigenen Sicherheitsverfahren umsetzen, erfordern eine Codeanalyse (Schritt 4 im Bild). Diese deckt sicherheitskritische Implementierungsmängel auf, etwa solche, die zu Möglichkeiten des Knackens eines Passworts führen, weil das Hashing, das das Kennwort unkenntlich macht, für heutige Cracking-Methoden unzureichend ist.

Apps, und damit die von ihnen verwalteten Daten und genutzten Zugänge zum Server, sind aufgrund der hohen Portabilität des Geräts, auf dem sie sich befinden, einer deutlich höheren Gefahr ausgesetzt, in fremde Hände zu gelangen, als browserbasierte Anwendungen. Ein Angreifer, der weiß, dass eine App beispielsweise die Anmeldung des Benutzers dauerhaft speichert, damit dieser sich nicht bei jeder Benutzung erneut einloggen muss, könnte gezielt Smartphones stehlen mit dem Ziel, sich diesen Zugang zum Benutzerkonto zu verschaffen. Dem muss der Entwickler im Gestalten der App durch Betrachtung einschlägiger Missbrauchsszenarien Rechnung tragen in Schritt 5 der Sicherheitsüberprüfung.

(ur/sf)

Thomas Schreiber ist Geschäftsführer des auf Webanwendungssicherheit und Softwareentwicklung spezialisierten Beratungsunternehmens SecureNet GmbH.

ANBIETER VON WEB APPLICATION FIREWALLS

Hersteller	Produkt	Website
Applicure	dotDefender Web Application Firewall	www.applicure.com
Armorlogic	profense WAF	www.armorlogic.com
Barracuda Networks	Barracuda Web Application Firewall	https://www.barracudanetworks.com/ns/products/web-site-firewall-overview.php
Bee Ware	iSuite WAF	www.bee-ware.net/en/products/web-application-firewall
Cisco	ACE Web Application Firewall	www.cisco.com
Citrix	NetScaler Web Application Firewall	www.citrix.de/produkte/netscaler/
Dell	SecureWorks	www.secureworks.com/it_security_services/web_app_firewall/
DenyAll	rWeb	www.denyall.com/decision-maker/products/rweb_en
Ergon	Airlock	www.ergon.ch/de/airlock/
F5	BIG-IP Application Security Manager	www.f5.com/products/big-ip/application-security-manager.html
Fortinet	Fortiweb	www.fortinet.com
Imperva	Web Application Firewall	www.imperva.com
Qualys	QualysGuard	www.qualys.com/forms/web-application-firewall/
Radware	AppWall	www.radware.com
Riverbed	Stingray	www.riverbed.com/us/products/stingray/stingray_af.php
secunet	Web Application Firewall	www.secunet.com/de/produkte-dienstleistungen/business-security/netzwerksicherheit/web-application-firewall/
SonicWall	Web Application Firewall	www.sonicwall.com
Sophos	Astaro WAF	www.astaro.com/node/13927
Trustwave	WebDefend	https://www.trustwave.com/web-application-firewall
United Security Providers	USP Secure Entry Server	https://www.united-security-providers.ch/de/loesungen/web-access-management/web-application-firewall
Open Source		
IronBee	Web Application Firewall	www.ironbee.com
ModSecurity	ModSecurity	www.modsecurity.org
WebCastellum	Open Source Web Application Firewall	www.webcastellum.org

Die Übersicht erhebt keinen Anspruch aufVollständigkeit.