Aus dem Alltag eines Beraters

Nachhaltige Sicherheit von der Softwareentwicklung bis zur Unternehmenskultur

Bastian Braun

Erfolgreiche Angriffe auf Anwendungen und Unternehmen gehören in der öffentlichen Berichterstattung längst zum Alltag. Jeder erfolgreiche Angriff ist letztendlich auf einen menschlichen Fehler zurückzuführen. Wer allerdings glaubt, solche Fehler restlos ausräumen oder vermeiden zu können, der irrt. Stattdessen ist ein ganzheitlicher Umgang zur Minimierung von Ursachen, Identifikation von eingetretenen Fehlern und Reaktion auf Ausnutzungsversuche angemessen.

In den Medien wird regelmäßig über Verwundbarkeiten, erfolgreiche Angriffe oder gar Cyberwar debattiert. Auch die Politik fühlt sich berufen, eine Strategie zur Abwehr von Cyber-Angriffen auf die kritischen Infrastrukturen des Landes zu entwerfen. Doch was bedeuten die (Un-)Sicherheit von Software und entsprechende Angriffsoptionen im Detail?

Genau genommen ist die Sicherheit einer Anwendung keine Ja/nein-Entscheidung. Sicherheit ist eine Querschnittsfunktion ähnlich der Performanz. Es ist daher genauso wenig möglich zu sagen, dass eine Anwendung sicher ist, wie man behaupten kann, dass sie performant ist. Stattdessen sind beide Eigenschaften nahezu beliebig skalierbar.

Die Bedrohungsanalyse

Zu Beginn der Projektphase erstellt ein Sicherheitsberater oder ein anderes Teammitglied mit erweiterten Kenntnissen im Bereich Sicherheit eine Bedrohungsanalyse (engl. Threat Modelling [GitHub]). Diese Analyse berücksichtigt die angestrebte Funktionalität und die verarbeiteten Daten und stellt Bedrohungen für diese Daten beispielsweise nach dem STRIDE-Modell dar [Wiki].

Dr. Bastian Braun arbeitet als Senior Consultant IT Security bei mgm security partners. Er unterstützt Entwicklerteams bei der Integration von IT-Sicherheit, gibt Seminare für Entwickler, Projektleiter, Entscheidungsträger und Penetrationstester und führt Produkt- und Sicherheitsanalysen durch.
E-Mail: bastian.braun@mgm-sp.com

Ein angemessenes Sicherheitsniveau wird durch die Festlegung von geeigneten Maßnahmen zur Minderung der identifizierten Bedrohungen definiert. Maßnahmen sind geeignet, wenn sie die Bedrohungsminderung mit den Kosten für die Umsetzung und Pflege sowie die Auswirkungen auf die Benutzbarkeit der Anwendung miteinander in Einklang bringen. Somit darf das Ziel nicht grundsätzlich lauten „So viel Sicherheit wie möglich", sondern „So viel Sicherheit wie nötig". Es ist wirtschaftlich nicht zu vertreten, stets die mächtigste Maßnahme zu ergreifen. Stattdessen dürfen die Kosten einer zu ergreifenden Maßnahme nicht höher sein als der Schaden, den sie verhindert.

Die Auswahl der bevorzugten Maßnahme geschieht durch Abstimmung aller wichtigen Interessenvertreter aus Fachbereich, IT und gegebenenfalls Marketing und Vertrieb. Es ist wichtig, die Analyse während des Projektverlaufs regelmäßig anzupassen, um neu hinzugekommenen Bedrohungen sinnvoll begegnen zu können. Dies gilt insbesondere für agile Projekte.

Das Training

Ein Team kann eine Bedrohungsanalyse und die vereinbarten Maßnahmen nur mit einem Grundverständnis zum Thema Sicherheit umsetzen. Aus diesem Grund sollte das gesamte Entwicklungsteam zu einem möglichst frühen Zeitpunkt ein Grundlagen-Training erhalten.

Ein solches Training vermittelt nicht nur technisches Wissen, das hilft, sicheren Code zu schreiben, sondern auch ein Bewusstsein, das dazu führt, kritische Funktionen und Codestellen zu erkennen und gegebenenfalls einen Experten zurate zu ziehen. Ohne dieses Sicherheitsbewusstsein und die Unterstützung des Teams kann ein Berater beziehungsweise Sicherheitsbeauftragter die Schutzziele der Anwendung nur schwer erreichen. Wir haben darüber hinaus erlebt, wie das Sicherheitsbewusstsein Vorbehalte des Entwicklerteams gegen die Integration eines Sicherheitsberaters abbauen kann.

Ohne das Training und das Bewusstsein für die Materie fühlen sich viele insbesondere erfahrene Entwickler kontrolliert oder sogar verfolgt. Es entsteht die Meinung, dass man auch in der Vergangenheit ohne solche Unterstützung ausgekommen sei und der Berater nur dafür geholt wurde, einzelne Entwickler als Ursache von Verwundbarkeiten zu identifizieren und zu brandmarken. Stattdessen ist ein Berater stets Teil des Teams und an der Erreichung der gemeinsamen Ziele interessiert.

Auch für das Management und die Entscheidungsträger kann ein sogenanntes Awareness-Training hilfreich sein, um die Gegenmaßnahmen im Rahmen der Bedrohungsanalyse einordnen zu können und Aufwände für die Sicherheit der Anwendung argumentieren zu können. Je weiter das Sicherheitsbewusstsein in einem Unternehmen verbreitet ist, desto besser werden Maßnahmen bei Entwicklung und Betrieb akzeptiert. Wir erleben, dass auch nicht-technisches Personal enorm von solchen Awareness-Schulungen profitiert. Die Konfrontation härtet die Betroffenen ein Stück weit gegen Social-Engineering-Angriffe über Telefon und E-Mail.

Die Erstellung der Bedrohungsanalyse und das Training des Teams sind wichtige Grundlagen für den Einstieg in die Entwicklungsphase.

Die Entwicklung

Aus Sicherheitsperspektive sind die Bedrohungsanalyse und die daraus abgeleiteten Maßnahmen die Grundlage für die Aktivitäten während der Entwicklungsphase. Grundsätzlich sind während dieser Phase zwei wesentliche Paradigmen zu beachten.

Zum einen müssen die Sicherheitsanforderungen jeder Komponente definiert werden. Das bedeutet, dass auf Ticketebene neben den fachlichen auch die Sicherheitsanforderungen beschrieben sind. Stellen Sie sich die Entwicklung einer Verkaufswebseite vor und ein Ticket, das die Integration von Gutscheincodes vorsieht. Die Sicherheitsanforderungen beschreiben dann nicht nur die Prüfung der eingegebenen Codes, sondern auch eine Maßnahme, um automatisiertes Raten gültiger Codes zu verhindern (Anti-Automatisierung). In der Summe müssen für die meisten Tickets keine spezifischen Anforderungen definiert werden. Es ist trotzdem

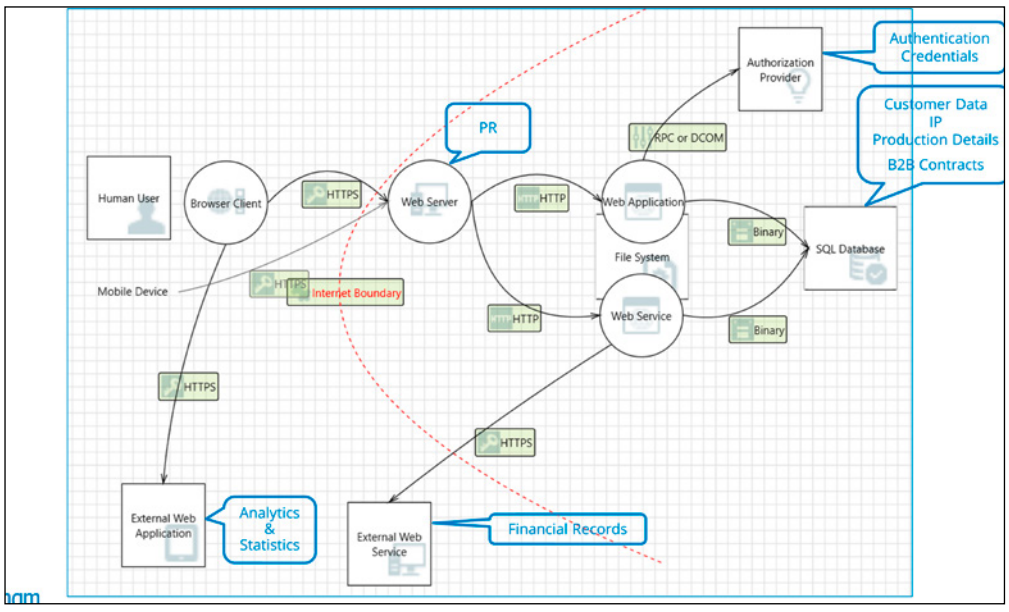
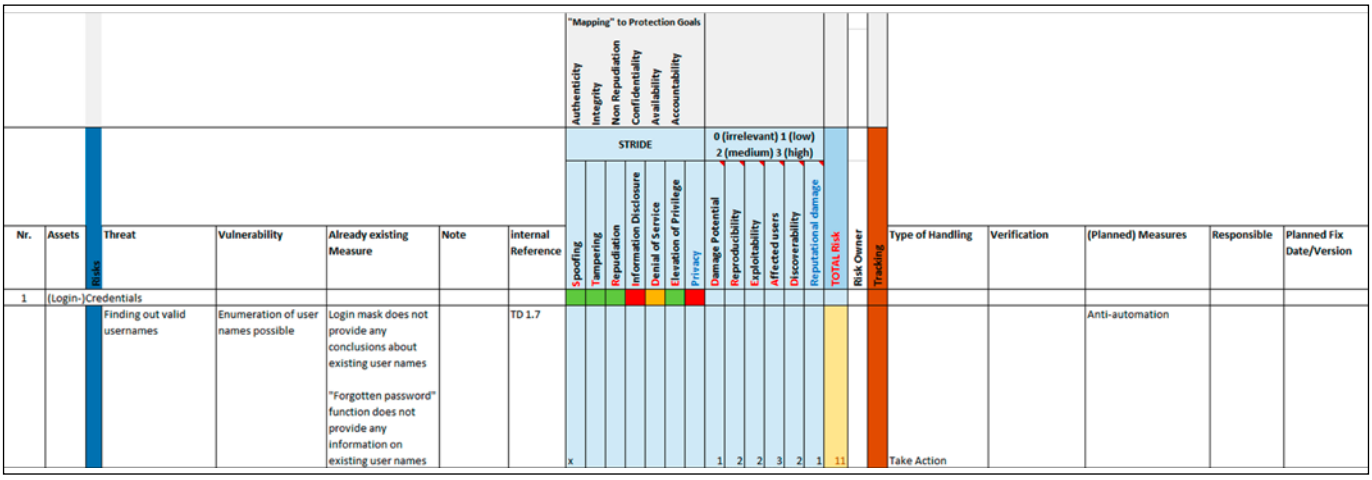


Abb. 1: Erster Schritt zur Bedrohungsanalyse durch grafische Darstellung der Komponenten und Assets (blau)



							"Mapping" to Protection Goals																			
							Authenticity	Integrity	Non Repudiation	Confidentiality	Availability	Accountability														
							STRIDE								0 (irrelevant) 1 (low) 2 (medium) 3 (high)											
Nr.	Assets	Risks	Threat	Vulnerability	Already existing Measure	Note	internal Reference	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege	Privacy	Damage Potential	Reproducibility	Exploitability	Affected users	Discoverability	Reputational damage	TOTAL Risk	Risk Owner	Tracking	Type of Handling	Verification	(Planned) Measures	Responsible	Planned Fix Date/Version
1	(Login-)Credentials																									
			Finding out valid usernames	Enumeration of user names possible	Login mask does not provide any conclusions about existing user names "Forgotten password" function does not provide any information on existing user names		TD 1.7	x							1	2	2	3	2	1	11			Take Action		Anti-automation		

Abb. 2: Zweiter Schritt zur Bedrohungsanalyse durch systematische Erfassung der Bedrohungen, zum Beispiel in einer Tabelle, alternativ eignet sich ein Ticketingsystem

wichtig, dass jedes Ticket durch eine Ja/nein-Auswahl als kritisch beziehungsweise unkritisch eingestuft wird, um für die kritischen Tickets entsprechende Anforderungen definieren zu können. Ein erfahrener Berater mit Kenntnissen der Anwendung kann mehrere Tickets pro Minute vorsortieren.

Neben den Anforderungen auf Ticketebene können viele Richtlinien projektweit festgelegt werden. Hierzu gehört die Verwendung von Bibliotheken, unter anderem zur Kryptografie, Eingabevalidierung und Ausgabecodierung (engl. „escaping“ oder „encoding“). Die Menge dieser Richtlinien wird als Secure Coding Guidelines bezeichnet und berücksichtigt die im Projekt verwendeten Technologien. Sie dienen als Nachschlagwerk für Entwickler, als Orientierungspunkt während des Peer Reviews und als Einstiegspunkt für neue Projektmitglieder.

Das zweite wesentliche Paradigma ist die Verifikation der definierten Sicherheitsanforderungen. Hierfür bietet sich eine Kombination aus automatischen und manuellen Überprüfungen an. Statische und dynamische Analysewerkzeuge können zeit- oder ereignisgesteuerte Prüfungen der Anwendung durchführen. Die Ergebnisse der Überprüfungen sollten zeitnah begutachtet werden, um falschpositive Meldungen, das heißt, berichtete Schwachstellen, die de facto keine Schwachstellen sind, auszusortieren und echte Schwachstellen unverzüglich zu beheben. Solche automatischen Überprüfungen sind gut geeignet, um Schwachstellen auf Codeebene zu identifizieren. Dies sind beispielsweise unzureichende Eingabevalidierung und fehlende Ausgabecodierung. Im Verhältnis zu manuellen Codeanalysen oder Penetrationstests skalieren die automatischen Analysen bei regelmäßiger Durchführung erheblich besser.

Allerdings können Analysewerkzeuge bisher keine logischen Schwachstellen identifizieren. Zur Klasse dieser Schwachstellen gehört zum Beispiel die fehlende Anti-Automatisierung schützenswerter Funktionen, siehe oben, sowie die Antwort, dass bei der Anmeldung das falsche Passwort eingegeben wurde statt einer generischen Fehlermeldung, dass Benutzername oder Passwort falsch seien. Für die Identifikation von logischen Schwachstellen ist ein Verständnis der Funktionalität notwendig. Deshalb können sie nur von menschlichen Testern erkannt werden. Glücklicherweise ist die Menge möglicher logischer Schwachstellen deutlich geringer als die Schwachstellen auf der Codeebene, sodass eine manuelle Verifikation immer noch gut skaliert.

An dieser Stelle wird die Rolle von Sicherheitsüberprüfungen als Verifikation von im Vorfeld definierten Eigenschaften offenkundig. Dies widerspricht dem immer noch weit verbreiteten Irrglauben, man könne Anwendungssicherheit in der Endphase der Entwicklung durch einen Test erreichen. In diesen Fällen überprüft der Tester Eigenschaften, die nie festgehalten oder angestrebt wurden, was die meist verheerenden Ergebnisse unausweichlich macht.

Das Deployment

Neben den Sicherheitseigenschaften des Codes trägt auch die Ausführungsumgebung zur Sicherheit des Gesamtsystems bei. Die Kommunikationssicherheit muss durch starke Kryptografie gewährleistet werden. Die Plattform sollte mit den aktuellen Sicherheitsupdates versorgt sein, und Drittsoftware wie DBMS, JVM, und Anwendungsserver müssen aktuell und ohne bekannte Schwachstellen sein.

Im Bereich von Webanwendungen müssen zusätzlich HTTP-Header aktiviert werden, die zusammen mit der Interpretation durch den Browser für Sicherheit sorgen.

Das Deployment geht für produktive Anwendungen in die Monitoringphase über.

Monitoring & Patch-Prozesse

Sobald die Anwendung produktiv ist, müssen Ereignisse im Rahmen des Betriebs überwacht werden. Die Log-Dateien der Anwendung geben Auskunft über funktionale Fehler und Angriffsversuche. Der Betrieb sammelt die Log-Dateien von allen produktiven Anwendungen inklusive der (Web Application-) Firewall an zentraler Stelle in einem Security Information and Event Management System (SIEM) und wertet die Ereignisse im Gesamtbild aus. Dabei werden Regeln angewendet, um sequenzielle Ereignisse zu Angriffsmustern zu verknüpfen. Sobald ein ernst zu nehmender Angriff[1] erkannt ist, findet eine Analyse der Anwendung statt, inwiefern sie für diesen Angriff verwundbar sein könnte.

Viele Angriffe dieser Art zielen auf Verwundbarkeiten in verwendeter Drittsoftware wie dem Anwendungsserver, Bibliotheken oder Frameworks. Nach der Veröffentlichung eines Sicherheitsupdates

1 Wir beobachten quantitativ primär stumpfe automatisierte Angriffe beispielsweise auf klassische Verwundbarkeiten in PHP-Anwendungen, die jedoch auch auf Java-Anwendungen durchgeführt werden. Solche Angriffe ohne jede Aussicht auf Erfolg sind aus unserer Perspektive nicht ernst zu nehmen.

durch den Hersteller beginnen professionelle Angreifer, diese Updates zu analysieren und die behobene Schwachstelle im Detail zu identifizieren. Anschließend generieren sie sogenannten Exploits, das heißt präparierte Anfragen, die die identifizierte Schwachstelle ausnutzen. Diese Exploits werden dann mit einer üblichen Verzögerung von wenigen Stunden [PSA2014-003] gegen Ziele im Internet angewendet. Es ist entscheidend, funktionierende Patch-Prozesse zu etablieren, um die Risiken für solche Angriffe zu minimieren. Für jede Maschine und jede Anwendung existiert eine verantwortliche Stelle, die zeitnah von Updates für die eingesetzten Komponenten erfährt. Nach der Veröffentlichung eines Updates wird dieses auf einem Testsystem geprüft und – sofern dort keine Fehlfunktionen auftreten – in Produktion gebracht. Sofern der unverzüglichen Anwendung des Patches wichtige Gründe entgegenstehen, ist sogenanntes Virtual Patching [OWASP] empfohlen.

An dieser Stelle sei auch vor der Einstellung „Wer soll denn schon unser System angreifen, uns kennt doch keiner" insbesondere im privaten Bereich gewarnt. Da diese Angriffe auf Schwachstellen in Standardsoftware leicht automatisiert werden können, finden sie im Allgemeinen großflächig ohne Ansehen des Einzelfalls statt. Hierdurch gerät fast jedes System früher oder später in den Fokus entsprechender Angriffsversuche unabhängig von dessen Bekanntheitsgrad oder vorgehaltener Daten. Ein erfolgreicher Angriff, der das System unter die Kontrolle des Angreifers stellt, integriert die Maschine meist in ein Botnetz, um sie wirtschaftlich nutzbar zu machen.

Wirtschaftliche Betrachtungen

Obwohl die in diesem Artikel beschriebenen Maßnahmen zweifellos Geld kosten, stellen sie bei konsequenter Anwendung die kostengünstigste und kalkulierbarste Option für IT-Sicherheit dar [Brau18].

Erstens sind Maßnahmen zur Behebung von Schwachstellen umso teurer, je später sie durchgeführt werden, was bereits durch Capers Jones im Buch „Applied Software Measurement" [Jon08] beschrieben wurde. Dieser Umstand erklärt sich leicht, wenn die Kosten für einen Berater oder einen Sicherheitsbeauftragten mit den Kosten für das ganze Team verglichen werden, das im Rahmen einer ungeplanten Projektverlängerung nach der Durchführung eines Penetrationstests die identifizierten Schwachstellen verstehen und beheben muss.

Zweitens sind die Kosten für die proaktiven und entwicklungsbegleitenden Maßnahmen kalkulierbar. Ein Berater kostet in etwa so viel wie ein Architekt wird aber üblicherweise nicht Vollzeit für ein einziges Projekt gebraucht. Verglichen mit den Kosten für das ganze Team, das „nachsitzen" muss, sind die Kosten für den Berater also verschwindend gering und zu jeder Phase kalkulierbar. Im Gegensatz dazu sind sowohl die Kosten für die Behebung der Schwachstellen nach einem Penetrationstest als auch die Kosten für einen erfolgreichen Angriff auf das eigene System kaum zu kalkulieren.

Drittens verhindert der beschriebene Ansatz das sonst häufige Dilemma des Projektverantwortlichen, sich zwischen einem pünktlichen Livegang mit Schwachstellen und einer Verschiebung des unter Umständen bereits angekündigten Livegangs entscheiden zu müssen. Beide Optionen bergen unterschiedliche wirtschaftliche Risiken, denen man am besten durch die Integration von Sicherheitsmaßnahmen in den Entwicklungsprozess begegnet.

Schließlich kann nur eine koordinierte Planung und Umsetzung von Maßnahmen sinnvolle Kompromisse erzielen. Die oben beschriebenen Maßnahmen nach der Bedrohungsanalyse sind ausgewogen und aufeinander abgestimmt, sodass an einer Stelle ggf. der Benutzbarkeit der Anwendung der Vorzug gegeben werden kann, wenn an einer anderen Stelle eine sehr mächtige Maßnahme umgesetzt wurde. Die nachträgliche Behebung von Schwachstellen erlaubt solche Abwägungen im Normalfall nicht mehr.

Die Unternehmenskultur

Die in diesem Artikel beschriebenen Maßnahmen beruhen auf Erfahrungen. In unserer Vergangenheit haben sie sehr gut funktioniert und belastbare Ergebnisse erzielt. Eine wesentliche Voraussetzung für die Einführung proaktiver Maßnahmen in einem Unternehmen ist jedoch stets eine ehrliche Fehlerkultur. Der eingangs beschriebene Umstand, dass jede Schwachstelle auf einem menschlichen Fehler basiert, bedeutet in der Konsequenz, dass man erst beginnen kann, solche Fehler zu reduzieren, wenn ein Unternehmen eine Kultur lebt, in der Fehler akzeptiert werden.

Erst die Fähigkeit eines Unternehmens, menschliche Fehler anzuerkennen, erlaubt, die oben beschriebenen Maßnahmen als Handlungsoption wahrzunehmen und gegebenenfalls umzusetzen. Den Mitarbeitern muss zugestanden werden, trotz bestem Wissen und Gewissen Fehler zu machen. Erst dann kann ein Verbesserungsprozess beginnen, in dessen Rahmen die Mitarbeiter ein Training bekommen und Analysewerkzeuge, die ihnen helfen, frühzeitig Rückmeldung hinsichtlich eingebrachter Schwachstellen zu erhalten. Als Berater erleben wir Erleichterung seitens der Entwickler, einen Ansprechpartner für komplexe Fragestellungen zu haben, Verantwortung entsprechend der Fachkenntnis zu verteilen, anstatt der stetigen Erwartungshaltung ausgesetzt zu sein, den Bereich Sicherheit selbst zu beherrschen.

Umgekehrt ist die Ignoranz menschlicher Fehler der sichere Weg zu wiederkehrenden Schwachstellen, teuren und fehlerbehafteten Anwendungen und frustrierten, weil permanent überforderten Mitarbeitern. In diesen Unternehmen wird Verantwortung nicht durch das Führungspersonal getragen, sondern von oben nach unten delegiert. Sobald die Verantwortung für die Sicherheit einer Anwendung dann auf einer Hierarchieebene angekommen ist, auf der kein Handlungsspielraum für den Einkauf von externem Wissen mehr besteht, ist der Weg zu nachhaltiger Sicherheit versperrt und die Frustration vorprogrammiert.

Literatur und Links

[Brau18] B. Braun, Mit Sicherheit agil: Entwicklungsbegleitende Security löst das Kostendilemma, mgm, 13.2.2018, https://live.mgm-tp.com/de/mit-sicherheit-agil-entwicklungsbegleitende-security-loest-das-kostendilemma/

[GitHub] OWASP/CheatSheetSeries, https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Threat_Modeling_Cheat_Sheet.md

[Jon08] C. Jones, Applied Software Measurement: Global Analysis of Productivity and Quality, 3. Auflage, McGraw-Hill, 2008

[OWASP] Virtual_Patching_Best_Practices, https://www.owasp.org/index.php/Virtual_Patching_Best_Practices

[PSA2014-003] Drupal Core – Highly Critical, Public Service Announcement 2014-003, https://www.drupal.org/forum/newsletters/security-public-service-announcements/2014-10-29/drupal-core-highly-critical

[Wiki] https://en.wikipedia.org/wiki/STRIDE_(security)