Side-Channel Leaks in Web-Applications

Clara Lange
Betreuer: Ralph Holz
Hauptseminar Innovative Internettechnologien und Mobilkommunikation SS2011
Lehrstuhl Netzarchitekturen und Netzdienste, Lehrstuhl Betriebssysteme und Systemarchitektur
Fakultät für Informatik, Technische Universität München
Email: langecl@in.tum.de

KURZFASSUNG

Mit der Entwicklung des World Wide Web von den Anfängen bis zum Web 2.0 stehen dem Benutzer immer mehr interaktive Dienste und Anwendungen im Internet zur Verfügung. Es können Sicherheitsprobleme entstehen, wie zum Beispiel Seitenkanalangriffe gegen Web-Anwendungen, die in dieser Arbeit behandelt werden. Eine Web-Anwendung ist aufgeteilt in eine Browser-Komponente, die die Schnittstelle für den Client bzw. den Endnutzer zur Nutzung der Anwendung darstellt, und in eine Server-Komponente. Diese Aufteilung hat zur Folge, dass ein Anteil der internen Informationen während der Kommunikation beider Komponenten im Netzwerk sichtbar wird. Daraus können Sicherheitslücken resultieren. Schwerpunktmäßig wird in dieser Arbeit das aktuelle Problem der Seitenkanalangriffe (*side-channel attacks*) gegen Web-Anwendungen behandelt, wobei der Angreifer trotz Verschlüsselung die Möglichkeit hat, Seitenkanal-Information zu belauschen, um personenbezogene Daten ausspähen zu können. Zusätzlich werden technische Herausforderungen, die sich während der Suche nach Lösungsansätzen ergeben, und Lösungsvorschläge, um die Gefahr zu beseitigen, in dieser Arbeit diskutiert. Abschließend wird der Blick auf zukünftige Entwicklungen gerichtet.

Schlüsselworte

Seitenkanalangriffe, *side-channel attacks*, Web 2.0, Web-Anwendungen, Benutzereingabe, Sicherheitslücken, Regelwerk

1. EINLEITUNG

Das World Wide Web unterlag einer drastischen Entwicklung bis hin zum Web 2.0 [6]. Früher existierten Webseiten nur als Plattformen für Informationen, das heißt es waren statische Seiten, die der Benutzer lediglich zur Informationsbeschaffung nutzte. Neben Performanzsteigerung und Reduktion der Kosten gelten Webseiten in der heutigen Zeit als Plattformen zum Mitmachen und als Plattformen, die Software-Anwendungen über das Internet bereitstellen. Immer mehr Daten, Dienste und Anwendungen werden im World Wide Web bereitgestellt. Im Web 2.0 steht der Benutzer im Fokus, es wird Wert auf Benutzerfreundlichkeit und Interaktivität gelegt. Dem Benutzer wird es ermöglicht, mehr mit dem Internet zu agieren und interaktiv an zum Beispiel der Gestaltung mitzuwirken. Daneben wird auch die Kommunikation zwischen Nutzern erleichtert, zum Beispiel durch soziale Netzwerke und Online-Communities. Durch die wachsenden Kommunikationsmöglichkeiten hat der Benutzer den Vorteil, dass Daten leichter zu jeder Zeit und an jedem Ort verfügbar sind - Webseiten spielen vor allem für den Informationsaustausch eine große Rolle.

Zusätzlich können Desktop-Anwendungen in absehbarer Zeit durch Web-Anwendungen ersetzt werden. Seit Web 2.0 wird die Funktionalität von Desktop-Anwendungen in das Internet transferiert, wobei der Browser die Schnittstelle zwischen Benutzer und Anwendung darstellt. Man spricht auch von *Software-as-a-Service* [1]. Dabei sollte Schutz vor Nutzerdaten im Vordergrund stehen, die Sicherheit für personenbezogene Daten muss gewährleistet sein.

Seitenkanalangriffe gegenüber Web-Anwendungen gewinnen immer mehr an Bedeutung. Da Web-Anwendungen unterteilt in eine Browser- und eine Server-Einheit sind, ist Kommunikation zwischen diesen beiden Einheiten notwendig. Der Seitenkanal ist hierbei der Kommunikationsweg. Trotz dass die Kommunikation zwischen Browser und Server verschlüsselt ist, kann der Angreifer Attribute dieser verschlüsselten Datenübertragung ausspähen, sog. Seitenkanal-Informationen, und dadurch Rückschlüsse auf die Eingabe des Benutzers ziehen.

Heutzutage existieren neben Seitenkanalangriffen weitere Angriffe wie zum Beispiel Phishing, Sniffen und Spoofen. Phishing-Angriffe erfolgen, indem der Angreifer eine vertraute Webseite nachbildet, um den Benutzer zu verleiten, seine persönlichen Authentifikationsdaten einzugeben. Sniffen meint das Belauschen des Netzwerkverkehrs und Spoofen das Vortäuschen einer falschen Identität während eine Kommunikation - sprich Identitätsdiebstahl [10].

Insbesondere bei extrem sicherheitsbedürftigen Web-Anwendungen werden Lösungen zum Schließen von Sicherheitslücken benötigt. Zum einem müssen Sicherheitslücken identifiziert werden; zum anderem sind Regelwerke zur Schadensbegrenzung erforderlich. Hierbei ist anwendungsspezifisches Wissen von großem Nutzen.

Überblick - Aufbau der Arbeit. In dem folgenden Abschnitt 2 werden Grundlagen zu den Themen Seitenkanalangriffe, Wandlung von Desktop- zu Web-Anwendungen und spezifische Eigenschaften von Web-Anwendungen erklärt. Abschnitt 3 zeigt explizite, aktuelle Beispiele von Seitenkanalangriffe gegen Web-Anwendungen. Die darauffolgenden Abschnitte 4 und 5 behandeln technische Herausforderungen, Lösungsansätze beziehungsweise Verteidigungsmechanismen. Abschnitt 6 enthält eine kurzes Fazit.

doi: 10.2313/NET-2011-07-2_19

2. GRUNDLAGEN

2.1 Seitenkanalangriffe

Ein Angriff kann definiert werden als ein nicht-autorisierter Zugriff auf ein schützenswertes Objekt [10].

Allgemein bezeichnen Seitenkanalangriffe Attacken, die Schwachstellen in der Implementierung bzw. im Sourcecode ausnutzen und nicht Attacken, die einen kryptographischen Algorithmus selbst angreifen [3]. Seitenkanalangriffe treten nicht nur gegen Web-Anwendungen auf.

2.1.1 Angriffsmethoden

Grundsätzlich stellen Seitenkanäle den Transportweg für die Kommunikation dar. Das Hauptproblem bei Seitenkanalangriffen ist, dass Seitenkanal-Informationen ausgenutzt werden, um auf die Inhalte rückschließen zu können. Solche Seitenkanal-Informationen können Attribute einer verschlüsselten Kommunikation, wie zum Beispiel Paketgröße oder Dauer des Paketes, sein. Meist sind es physikalische Eigenschaften, die vom Angreifer beobachtet werden.

Grundsätzlich können Seitenkanäle mit zwei verschiedenen Methoden angegriffen werden [3]:

- *Data Leakage* (auch Seitenkanalanalyse genannt): Der Angreifer belauscht passiv einen Seitenkanal, um an personenbezogene Daten zu kommen.
- *Fault Injection*: Der Angreifer manipuliert aktiv den Seitenkanal, indem er Fehler einfügt, die das Verhalten der Anwendung ändern.

2.1.2 Spezielle Seitenkanalangriffe

Spezielle, bekannte Seitenkanalangriffe sind zum Beispiel folgende [2]:

- *Timing Attack*: Angriff, bei dem der Angreifer die Zeiten beobachtet und analysiert, die gebraucht werden, um kryptographische Algorithmen auszuführen, da diese Algorithmen sehr rechenintensiv sind
- *Fault (Injection) Attack*: Angriff gegen kryptographische Hardwaregeräte (z. Bsp. Smartcards), indem der Angreifer beispielsweise fehlerhafte Eingabedaten sendet
- *Power Analysis Attack*: Angriff, bei dem der Stromverbrauch bei kryptographischen Elementen betrachtet wird
- *EM Attack*: Angriff, bei dem der Angreifer elektromagnetische Strahlung von elektronischen Geräten beobachtet
- *Acoustic Attack*: Angriff gegen einen Audio-Kanal

Auch eine Kombination mehrerer Angriffe ist möglich.

Gegenmaßnahmen können *randomization*, *blinding* und *masking* sein. Bei der Randomisierung werden die Daten, die durch Seitenkanäle durchsickern können, zufällig verteilt. *Blinding* dient dazu, dass die Eingabe eines Algorithmus in einige unvorhersehbare Zustände geändert wird. Unter Maskierung versteht man das Verdecken der Nachricht und des Schlüssels, indem man am Anfang der Berechnung sowohl die Nachricht als auch den Schlüssel mit einer zufälligen Maske umhüllt [4].

2.1.3 Seitenkanalangriffe aus der Vergangenheit

Bei dem in Referenz [8] beschriebenen Angriff handelt es sich um Seitenkanalangriffe, die im Zusammenhang mit SSH möglich sind. SSH steht für *Secure Shell* und ist ein Netzwerkprotokoll. Der Angreifer kann ein verstecktes *Markov Diagramm* erstellen. Das bedeutet, dass aus den eingegebenen Daten Wahrscheinlichkeiten vom Angreifer berechnet werden können, so dass auf die Tastenschläge des Benutzers rückgeschlossen werden kann. Mit Hilfe dieses Modells kann der Angreifer zum Beispiel ein Passwort 50 Mal schneller als mit einer *Brute-Force* Attacke erraten.

Seitenkanalangriffe können außerdem bei *Voice over IP* auftreten [9]. *Voice over IP* bezeichnet im Allgemeinen digitalisierte Telefonie über Computernetzwerke. Hierbei kann ein Angreifer, trotz verschlüsselter Übertragung der Audioanrufe, die Längen der verschlüsselten VoIP-Pakete nutzen, um gesprochene Sätze der Benutzer zu identifizieren. Dem Angreifer wird es somit ermöglicht, das Telefongespräch zu belauschen und somit die Konversation teilweise zu rekonstruieren.

2.2 Wandlung von Desktop-Anwendungen zu Web-Anwendungen

Auch gegen Web-Anwendungen sind Seitenkanalangriffe möglich. Dies wird bei den Unterschieden zwischen Desktop- und Web-Anwendungen offensichtlich.

Mit der Entwicklung des Internets kann der Benutzer interaktiver im und mit dem Internet agieren. Außerdem werden vollwertige Anwendungen und Dienste im World Wide Web bereitgestellt. Die Ähnlichkeiten zwischen Web- und Desktop-Anwendungen sind sehr groß. Eingaben kommen sowohl für Desktop- als auch für Web-Anwendungen entweder vom Benutzer selbst oder vom Dateisystem. Bei beiden Anwendungen regelt der interne Informationsfluss, der aus Daten- und Kontrollfluss besteht, die Zustandsübergänge.

Jedoch gibt es auch Unterschiede. Ein Vorteil von Web-Anwendungen besteht darin, dass der Endnutzer keine spezielle Software und die dazugehörige Installation benötigt, um diese auszuführen. Unabhängig vom Betriebssystem kann der Endnutzer die Anwendung im Browser benutzen. Außerdem existieren automatische Updates, so dass Aktualisierungen alleine durch das System durchgeführt werden und die Anwendung immer auf dem neusten Stand ist - der Nutzer hat somit keinen zusätzlichen Arbeitsaufwand.

Ein weiterer großer Unterschied ist, dass Web-Anwendungen dem Benutzer online zur Verfügung stehen. Sie bieten den gleichen Funktionsumfang und ähnliche Benutzeroberflächen wie Desktop-Anwendungen.

Desktop-Anwendungen besitzen nur den Informationsfluss, der aus Daten- und Kontrollfluss besteht. Bei Webanwendungen hingegen existieren noch die Informationen, die durch

doi: 10.2313/NET-2011-07-2_19

das Netzwerk gehen, da Web-Anwendungen in eine Browser- und eine Serverkomponente aufgeteilt sind. Diese Informationen werden *Web-Flows* genannt und existieren neben dem Informationsfluss. Dieser Unterschied führt dazu, dass Seitenkanalangriffe überhaupt gegen Web-Anwendungen möglich sind, wie weiter unten beschrieben wird.

Ein Vorteil von Web-Anwendungen ist offensichtlich: Ein Server stellt Daten zur Verfügung, speichert und verwaltet diese. Der Endbenutzer kann über den Browser, der als Schnittstelle zwischen dem Benutzer und dem Server fungiert, auf diese Daten in der Anwendung zugreifen.

Jedoch wird hier auch ein Nachteil sichtbar. Der Browser muss mit dem Server kommunizieren. Das Problem besteht darin, dass *Web-Flow* Vektoren, die durch den Seitenkanal gehen, für einen Angreifer sichtbar sind. Diese Vektoren kann der Angreifer nutzen, um Benutzereingaben zu ermitteln. In den folgenden Abschnitten wird genauer erklärt, wie Web-Anwendungen aufgebaut sind und wie daraus Seitenkanalangriffe gegen Web-Anwendungen resultieren.

2.3 Web-Anwendungen - Aufbau, Funktionalität und Schwachstellen

Web-Anwendungen sind unterteilt in eine Browser- und eine Server-Komponente. Der Webbrowser ist die darstellende Komponente bzw. das User-Inferface, die es dem Benutzer erlaubt, die Anwendung zu benutzen. Das resultierende Problem ist, dass ein Bruchteil des internen Informationsflusses im Netzwerk sichtbar wird - sprich die *Web-Flows*. Der Grund ist, dass eine Anwendung in verschiedene Zustände mit einer Zustandsübergangsfunktion übergehen kann. Diese Information kann mit Hilfe von *Web-Flow* Vektoren sichtbar werden. Der Angreifer kann Rückschlüsse auf personenbezogene Daten ziehen, sobald er *Web-Flows* belauschen kann. Es wird offensichtlich, dass *Web-Flows* den Angriffspunkt für Belauscher darstellen.

Grundsätzlich sind kryptographische Methoden notwendig, die den Netzwerkverkehr verschlüsseln, so dass der Angreifer keinen direkten Zugriff auf Klartextnachrichten hat. Abhilfe sollen HTTPS oder WPA/ WPA2 schaffen. Trotz dieser Kommunikationsprotokolle und Verschlüsselungsmethoden kann der Angreifer Seitenkanal-Informationen ausspähen, wie zum Beispiel:

- Größe des Pakets
- Anzahl der Pakete
- Dauer der Übermittlung eines Pakets

Mit Hilfe der Seitenkanal-Informationen, kann der Angreifer Rückschlüsse über interne, vertrauenswürdige Informationen und die Inhalte der Kommunikation herausfinden.

Somit sieht der allgemeine, grobe Ablauf eines Seitenkanalangriffs folgendermaßen aus: Eine Webseite hat eine individuelle Größe. Sobald Ressourcen bzw. Objekte mit verschiedenen Größen auf die Seite geladen werden und der Angreifer Seitenkanal-Informationen belauschen kann, kann der Angreifer eingegebene Informationen des Benutzers ermitteln. Die genaue Vorgehensweise des Angreifers bei einem Seitenkanalangriff wird in Kapitel 2.3.2 beschrieben.

Die Gefahr, die durch einen erfolgreichen Seitenkanalangriff entstehen kann, ist, dass ein Angreifer auf personenbezogene Daten und Online-Aktivitäten eines Benutzers zugreifen kann. Erstens resultiert daraus ein enormes Sicherheitsproblem, falls der Angreifer an extrem sicherheitsbedürftige Daten kommen kann. Zweitens stellt dies eine Verletzung der Privatsphäre des Benutzers dar.

2.3.1 Modellierung von Web-Anwendungen

Eine Webanwendung kann wie folgt als Dreiertupel beschrieben werden:

- S: S stellt die Menge aller Zustände der Web-Anwendung dar.
- Σ: Σ stellt die Menge der Eingaben dar, die akzeptiert werden.
- δ: S x $\Sigma \rightarrow$ S: δ ist die Zustandsübergangsfunktion.

Allerdings kann eine Web-Anwendung aus Sicht des Angreifers zu einem Fünfertupel erweitert werden, wie es in Referenz [1] definiert ist. Neben S, Σ und δ kommen folgende zwei Elemente dazu:

- V: V stellt die Menge aller *Web-Flow* Vektoren dar. Diese Menge beschreibt die beobachteten Eigenschaften des verschlüsselten Datenaustauschs.
- f: S x $\Sigma \rightarrow$ V: f ist die Funktion, die angibt, was der Angreifer beobachten kann. Zustandsübergänge sind immer mit *Web-Flow* Vektoren verbunden.

Die beobachteten Attribute können vom Angreifer genutzt werden, um Originalzustände und die dazugehörigen Eingaben zu rekonstruieren.

2.3.2 Seitenkanalangriffe gegen Web-Anwendungen - Reduktion des Ambiguity Sets

Das *Ambiguity Set* kann man definieren als eine Menge von Mehrdeutigkeiten. Der Angreifer versucht, die verschiedenen Bedeutungen der Benutzereingabe zu analysieren und bestimmte auszuschließen. Somit kann er eine eindeutige Lösung herleiten, wie die Benutzereingabe aussah.

Der Angreifer versucht aus den für ihn sichtbaren Daten, Rückschlüsse über die Eingabe des Benutzers zu ziehen. Er beobachtet eine Sequenz von Vektoren (v_t, ..., v_{t+n-1}) und verringert per Ausschlusskriterium das *Ambiguity Set* so weit, bis nur noch eine bestimmte eindeutige Kombination aus der Menge der möglichen Informationen für einen Vektor v übrig bleibt. Grundsätzlich muss dem Angreifer eine gewisse Grundmenge von Informationen zur Verfügung stehen, um Vektoren beobachten, identifizieren und ausschließen zu können.

Abbildung 1 veranschaulicht die Reduktion eines *Ambiguity Sets*: Zu einem bestimmten Zeitpunkt t befindet sich eine

doi: 10.2313/NET-2011-07-2_19

Anwendung im Zustand s_t, der eine Eingabe entweder vom Benutzer oder vom Dateisystem akzeptiert - mit einer Eingabe kommt die Anwendung vom Zustand s_t zum Zustand s_{t+1}. Der Eingaberaum ist in k verschiedene Mengen aufgeteilt, die disjunkt sind. Jede Eingabemenge bewirkt eine Zustandsänderung in einen Zustand, der von s_t aus erreicht werden kann (alle Zustände, die von s_t aus erreicht werden können, liegen in der Menge S_{t+1}, wobei diese Menge eine Teilmenge von S darstellt: $S_{t+1} \subset S$).

Der Angreifer betrachtet eine Sequenz von Vektoren (v_t, ..., v_{t+n-1}). Diese Vektoren veranlassen die Web-Anwendung, eine Zustandsänderung von s_t zu s_{t+1} durchzuführen. Die Sequenz entsteht demnach durch n Zustandsübergänge vom Ausgangszustand s_t. Bis jetzt hat der Angreifer lediglich die Information über die Vektoren; er hat noch keine Informationen über die Eingabe des Benutzers.

Da es k verschiedene Möglichkeiten für die Eingabe gibt, hat das *Ambiguity Set* die Größe k. Nun geht der Angreifer wie folgt vor: mit Hilfe des ersten Vektors v_t weiß er, dass nur Eingaben, die zu einer Untermenge von S_{t+1} führen, den Vektor v_t produzieren können. Das bedeutet, dass nur Übergänge zu einer bestimmte Untermenge von S_{t+1} durch diesen Vektor ermöglicht werden. Diese Untermenge wird mit D_{t+1} gekennzeichnet. Der Angreifer hat nun die Möglichkeit, eine bestimmte Menge aus der Menge der möglichen Informationen zur Eingabe auszuschließen - die tatsächliche Eingabe des Benutzers kann nur von k/α der Menge aller Eingabemöglichkeiten stammen. α ist der erste Reduktionsfaktor dieses Zustandsüberganges.

Der Angreifer wiederholt diese Prozedur und kann erneut das *Ambiguity Set* von D_{t+1} reduzieren. Hierbei betrachtet er die folgenden Vektoren (v_{t+1}, ..., v_{t+n-1}). Bei dieser Reduktion kann der Angreifer erneut eine bestimmte Menge aus der Menge der möglichen Informationen mit Hilfe des folgenden Reduktionsfaktors β ausschließen: k / (α*β). Somit wiederholt der Angreifer die Methode, die Menge an Mehrdeutigkeiten zu reduzieren, so lange, bis er eine eindeutige Lösung hat, wie die Benutzereingabe aussah. Dazu verwendet er die weiteren *Web-Flow* Vektoren ab v_{t+2}. Somit fasst der Reduktionsfaktor β alle folgenden Reduktionen des *Ambiguity Sets* zusammen. Am Ende kann die Benutzereingabe identifiziert werden. Es ist offensichtlich, dass die Reduktionsfaktoren durch Zustandsübergänge entstehen und dass sie anwendungsabhängig sind.

Die Methode, das *Ambiguity Sets* zu reduzieren, funktioniert umso besser, je weniger Möglichkeiten der Benutzer für seine Eingabe hat. Zum Beispiel erlauben Drop-Down Listen nur eine eingeschränkte Auswahl, wohingegen die Reduktion des *Ambiguity Sets* bei Freitexten aufgrund des größeren Informationsgehalts schwieriger ist.

2.3.3 Wahrscheinlichkeit eines Seitenkanalangriffs gegen Web-Anwendungen

Die Wahrscheinlichkeit eines Seitenkanalangriffs ist abhängig von der Eingabe, den Reduktionsfaktoren und signifikanter Traffic-Unterschiede.

- Die Gefahr eines Seitenkanalangriffs steigt, je weniger Möglichkeiten der Benutzer bei der Eingabe hat - das heißt die Eingabe besitzt eine geringe Entropie. Es stellt sich die Frage, ob der Angreifer effizient testen kann, um welche Benutzereingabe es sich anhand des Ergebnisses, das produziert wird, handelt.

- Bei einer zustandsbehafteten Kommunikation entstehen Reduktionsfaktoren. Je größer die Reduktionsfaktoren, desto größer die Gefahr eines Seitenkanalangriffs; denn der Angreifer kann mit großen Reduktionsfaktoren Mehrdeutigkeiten ausschließen und auf die ursprüngliche Eingabe des Benutzers schließen. Fraglich ist, wie groß das Ausmaß der Informationen ist, die der Angreifer erfahren kann.

- Je mehr signifikante Traffic Unterschiede für den Angreifer erkennbar sind, desto größer ist die Wahrscheinlichkeit, dass der Angreifer auf personenbezogene Daten rückschließen kann. Web-Traffic entsteht zum Beispiel, wenn eine Ressource auf eine Webseite hochgeladen wird.

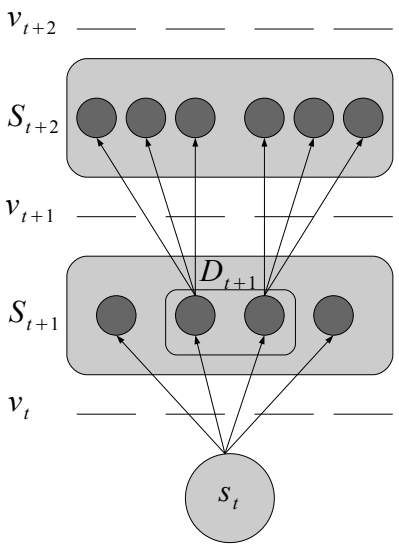

Abbildung 1: Reduktion des *Ambiguity Sets* [1]

3. BEISPIELE FÜR SEITENKANALANGRIFFE GEGEN WEB-ANWENDUNGEN

In diesem Kapitel wird auf Beispiele zu Seitenkanalangriffen speziell gegen Web-Anwendungen eingegangen, die trotz Verschlüsselung möglich sind. Vor allem das Szenario, das *Auto-Suggestion* betrifft, wird genauer erklärt.

3.1 Aktuelle Herausforderungen des Web 2.0

Mit der Entwicklung des Web 2.0 und der enormen Online-Bereitstellung von Diensten und Anwendungen, wächst die Gefahr von Seitenkanalangriffen. Die Hauptursache von Seitenkanalangriffen sind Features, die eine Web 2.0 Anwendung ausmachen, wie zum Beispiel *AJAX GUI Widget*. Dies ist ein Konzept der Datenübertragung zwischen Browser und Server, bei dem Web Traffic mit jedem Mausklick oder mit jedem Tastendruck generiert wird [7].

doi: 10.2313/NET-2011-07-2_19

Datenschutz und Sicherheit müssen trotzdem gewährleistet sein, vor allem bei extrem sicherheitsbedürftigen Web-Anwendungen und für vertrauenswürdige Daten, die zum Beispiel Gesundheitsstand, Einkommen der Familie, Investmentgeheimnisse oder Steuerinformationen betreffen.

3.2 Probleme trotz HTTPS

HTTPS steht für *Hypertext Transfer Protocol Secure* und ist ein Kommunikationsprotokoll im Internet [5]. Es wird für die Authentifizierung und Verschlüsselung der Kommunikation zwischen Webserver und Browser im Internet verwendet. Dies bedeutet, dass zum einen Daten durch Verschlüsselung geschützt sein sollten vor Sniffing-Angriffen; zum anderen sollten Phishing-Angriffe durch Authentifizierung abgewehrt werden.

3.2.1 Automatische Vorschläge bei Benutzereingabe

Seitenkanalangriffe können bei Webseiten auftreten, die bei Eingabe des Benutzers automatisch Vorschläge für die Vervollständigung der Eingabe anbieten. Das bedeutet, dass der Angreifer Rückschlüsse über die Eingabe ziehen kann, denn sobald der Benutzer einen Buchstaben eingibt, erscheint eine Liste mit Vorschlägen, die abhängig vom vorangegangenen Buchstaben verschieden groß ist und die bei jedem neuen Buchstaben aktualisiert wird.

Der Angreifer kann nun das *Ambiguity Set* - Menge mit Mehrdeutigkeiten - reduzieren, indem er zwischen der eigentlichen Eingabe des Benutzers nach jedem Buchstaben und der Größe der Antwort, also der Anzahl der Elemente in der Liste mit den automatischen Vorschlägen, vergleicht. Jeder Buchstabe generiert einen *Web-Flow* Vektor. Gleichzeitig wird die Liste mit Vorschlägen zur Vervollständigung der Eingabe automatisch angepasst. Diese Liste ist bei jedem Benutzer gleich groß, somit weiß auch der Angreifer über den Umfang der Liste Bescheid.

Zusätzlich ist die Kommunikation zustandsbehaftet. Jeder Buchstabe ist abhängig von dem vorherigen, das heißt jeder Buchstabe ist abhängig von dem eingegebenen Präfix. Somit wird es dem Angreifer weiterhin erleichtert, das Eingangssignal des Benutzers abzuleiten. Die Effektivität, mit der der Angreifer Rückschlüsse ziehen kann, ist abhängig von den Reduktionsfaktoren α und β.

Als konkretes, allerdings nicht allgemeingültiges Beispiel zum ersten Reduktionsfaktor α gibt man alle Buchstaben von ‘a’, ‘b’, ..., ‘z’ einmal als Anfangsbuchstaben ein. Die *Web-Flow* Vektoren, die jeweils dadurch erzeugt werden, werden verglichen: die Größe der Listen mit automatischen Ergänzungsvorschlägen ist verschieden, außer nach beispielsweise den Eingaben ‘h’ und ‘m’. Darauf wird im nächsten Absatz näher eingegangen. Weiterhin werden alle Buchstaben von ‘a’ bis ‘z’ eingegeben; diesmal werden diese allerdings als zweiter Buchstabe geschrieben nach einem ‘a’. Nur 20 Kombinationen erzeugen eine nicht-leere Liste mit Vorschlägen. Die restlichen Kombinationen sind ungültig. Dieses Beispiel zeigt, dass der Reduktionsfaktor α für den Angreifer von großer Bedeutung ist.

Der Reduktionsfaktor β hilft außerdem, das *Ambiguity Set* der Benutzereingabe zu verkleinern. Wie oben beschrieben, erzeugen die Buchstaben ‘m’ und ‘h’ als Anfangsbuchstaben jeweils eine gleich große Liste mit automatischen Ergänzungsvorschlägen, da beide Buchstaben einen identischen *Web-Flow* Vektor produzieren. Wenn man nun nach den Buchstaben ‘h’ oder ‘m’ einen weiteren Buchstaben eingibt, wie zum Beispiel ‘ha’, ‘hz’, ‘ma’ oder ‘mz’, erhält man 52 Möglichkeiten, wovon 20 ungültig sind, da diese eine leere Liste mit Vorschlägen liefern. Alle *Web-Flows* sind verschieden, mit der Ausnahme ‘ha’ und ‘ma’ - aus ihnen ergeben sich jeweils eine Liste, die gleich groß sind. Mit der Ausnahme von ‘ha’ und ‘ma’ kann ein Angreifer entweder ‘h’ oder ‘m’ als Anfangsbuchstaben ausschließen, indem er *Web-Flow* Vektoren belauscht.

Ein Beispiel stellt die Suchfunktion im Online-Kaufportal *http://www.amazon.de/* (Abbildung 2) dar. Sobald ein Benutzer nach einem Produkt sucht und in die Suchfunktion den Namen eingibt, schlägt diese Webseite automatisch Produkte vor.

3.2.2 Auswahl mit Mausklick aus Liste

Webseiten, bei denen man zum Beispiel zuerst den Anfangsbuchstaben wählt und anschließend per Mausklick aus einer Liste, die alle Wörter mit dem zuvor ausgesuchten Anfangsbuchstaben beinhaltet, auswählen kann, sind gefährdet.

Dieses Szenario stellt eine zustandsbehaftete Kommunikation in einer Hierarchie dar. Der Angreifer kann mit Hilfe eines Vergleichs zwischen Anfangsbuchstaben und der Größe der resultierenden Liste den Anfangsbuchstaben herausfinden - lediglich 26 Mal muss er testen, welcher Anfangsbuchstabe vom Benutzer eingetippt wurde unter der Voraussetzung, dass allen Benutzern die gleichen Daten zur Verfügung stehen. Das heißt, dass jeweils die einzelnen Listen bei jedem Benutzer gleich groß sind. Somit wird die Entropie der Benutzereingabe deutlich reduziert und die Zustände der Web-Anwendung können vom Angreifer eindeutig identifiziert werden.

Es sind auch andere Vorauswahlmöglichkeiten denkbar. Webseiten, die es dem Benutzer ermöglichen, Musik auszuwählen und anzuhören, wie zum Beispiel *http://www.lastfm.de/music*, beinhalten die Vorauswahl nach Musikgenres. Abhängig von der Auswahl eines Genres durch den Benutzer kann der Angreifer sich mit Hilfe der daraus resultierenden Liste, die Aktivität des Benutzers herleiten.

3.2.3 Drop-Down Liste gekoppelt mit Eingabe eines Namens oder eines Codes

Angreifer können bei einer Webseite, die zum Beispiel eine Drop-Down Liste gekoppelt mit der Eingabe eines Stadtnamen oder einer Postleitzahl enthält, eine Seitenkanalattacke durchführen.

Zum einen gibt der Stadtname oder die Postleitzahl dem Angreifer Aufschluss über die Eingabe des Benutzers. Der Angreifer kann diese Eingabe mit Hilfe der IP-Adresse erraten.

Zum anderen bietet die Drop-Down Liste eine geringe Entropie für die Eingabe. Die Anzahl der Möglichkeiten, ein Element aus der Drop-Down Liste auszuwählen, ist nicht

doi: 10.2313/NET-2011-07-2_19

sehr hoch. Somit ist es für den Angreifer effizient testbar, welches Element der Benutzer aus der Liste ausgewählt hat.

Erneut ist die bekannte Webseite *http://www.amazon.de/* (Abbildung 2) zu nennen, da sie Drop-Down Listen beinhaltet. Hier wird zwar kein Stadtname oder keine Postleitzahl eingegeben; allerdings kann der Benutzer ein Produkt, nach dem er sucht, eingeben. Außerdem bietet die Webseite bei der Benutzereingabe automatische Vorschläge, passend zum gesuchten Produkt, an.

Abbildung 2: Drop-Down Liste auf der Webseite *http://www.amazon.de/*

3.2.4 *Traffic-Analyse einer Anwendung*

Web-Anwendungen, bei denen der Benutzer Formulare ausfüllen und dabei einen Teil des Formulars mehrfach durchlaufen muss, sind von Seitenkanalangriffen betroffen. Ein typisches Beispiel ist ein Steuerformular, bei dem der Benutzer bei jedem einzelnen Kind ein bestimmtes Formular ausfüllen muss.

Jeder Benutzer produziert einen individuellen *Web-Flow* Vektor, weiterhin handelt es sich um eine Kommunikation mit Zuständen. Zustandsübergänge können leicht vom Angreifer mit Hilfe der *Web-Flow* Vektoren identifiziert werden. In gewissen Situationen reicht es sogar, wenn der Angreifer zählt, wie oft der Benutzer eine bestimmte Schleife durchlaufen hat.

Durch asymmetrische Ausführungspfade kommt das zusätzliche Problem hinzu, dass der Angreifer leicht Rückschlüsse ziehen kann. Der Angreifer kann sich ein Diagramm mit Zuständen und der internen Entscheidungslogik der Anwendung herleiten. Ein Beispiel wäre, wenn sich der Ausführungspfad einer Web-Anwendung in einem bestimmten Zustand in zwei parallele Pfade aufspaltet. Beide Pfade sind unterschiedlich lang und führen wieder in einen gemeinsamen Zustand. Angenommen der erste Ausführungspfad ist sehr kurz, wohingegen der andere bedeutend länger ist. Der erste, kurze Pfad produziert weniger *Web-Flow* Vektoren als der lange Ausführungspfad. Der Angreifer kann nun ermitteln, welchen Pfad der Benutzer gewählt hat.

3.2.5 *Graphische Visualisierung von Daten*

Ein Beispiel für eine gefährdete Webseite gegenüber Seitenkanalangriffen ist eine Webseite, die die Daten graphisch visualisiert. Der Benutzer klickt aus einer Liste entweder eine Graphik selbst oder einen Link zur Graphik an.

Auf der einen Seite ergibt sich abermals das Problem der geringen Entropie der Nutzereingabe. Der Benutzer hat nur begrenzte Auswahlmöglichkeiten, ein Element anzuklicken. Dadurch hat der Angreifer den Vorteil, dass er testen kann, welches Bild der Benutzer mit größter Wahrscheinlichkeit ausgewählt hat.

Auf der anderen Seite haben die Graphiken verschiedene Größen. Somit kann der Angreifer erheblich die Auswahlmöglichkeiten reduzieren. Dies ist vor allem der Fall, wenn der Angreifer die vom Server verwendeten Paketgrößen kennt.

Zum Beispiel kann sich ein Benutzer auf der Webseite *http://www.sueddeutsche.de/bilder* verschiedene Bilder ansehen. Jedes Bild hat eine individuelle Größe, so dass der Angreifer ableiten kann, welche Bilder der Benutzer angesehen hat. Grundsätzlich ist es für einen Angreifer leichter, einen erfolgreichen Seitenkanalangriff durchzuführen, umso größer die Unterschiede bezüglich der Größe von Bildern sind.

Falls die Graphiken von einer anderen, öffentlichen Webseite geladen werden, ist der Angreifer wieder im Vorteil - er kann die Paketgrößen vergleichen, welche Hinweise auf die Bildgröße geben, und sich somit die Graphik, die der Benutzer zuvor angeklickt hat, erschließen.

3.3 Probleme trotz WPA/ WPA2

WPA beziehungsweise WPA2 sind Verschlüsselungsverfahren für drahtlose Netzwerke des IEEE Standards 802.11i [10]. Der Vorgänger war WEP (= Wired Equivalent Privacy), welches ein WLAN Verschlüsselungsprotokoll darstellt [11]. WEP war allerdings anfällig, da Schlüssel geknackt werden konnten. Somit wird in der heutigen Zeit das etwas sichere WPA bzw. WPA2 verwendet. WPA steht für Wi-Fi Protection Access.

Trotz WPA bzw. WPA2 sind Seitenkanalangriffe auch ohne Anmeldeinformation oder Berechtigungsnachweis für das jeweilige WLAN möglich. Der Angreifer hat somit die Möglichkeit, aktuelle Aktivitäten von Benutzern zu ermitteln.

Seitenkanalangriffe im WLAN betreffen vor allem Suchmaschinen, wie zum Beispiel *Google* oder *Yahoo* [7]. Auf der einen Seite kann sich der Angreifer die Suchanfragehistorie beschaffen. Damit ist es möglich, vergangene Online-Aktivitäten des Benutzers nachzuvollziehen. Auf der anderen Seite beinhalten die meisten Suchmaschinen Features, so dass automatisch Vorschläge angezeigt werden. Obwohl die Menge der Vorschläge bedeutend groß ist, kann es zu Seitenkanalangriffen kommen, denn eine Folge des Features für automatische Vorschläge ist, dass die damit verbundenen Pakete leicht mit Hilfe der *Web-Flow* Vektoren identifiziert werden können.

Beispielsweise gibt ein Benutzer das Wort ‘Paket’ in eine Suchmaschine ein. Nach jedem Buchstaben wird eine neue Liste von Vorschlägen für ‘P’, ‘Pa’, ‘Pak’, ‘Pake’ und ‘Paket’ erzeugt. Mit jedem weiteren Buchstaben werden die WPA Pakete größer. Die resultierende Größe der Liste der Vorschläge wird mit jedem weiteren Vektor kleiner.

Obwohl die Liste mit den automatischen Vorschlägen enorm

doi: 10.2313/NET-2011-07-2_19

groß ist, funktioniert ein Seitenkanalangriff gegen Suchmaschinen im WLAN. Grundsätzlich hat der Benutzer die Möglichkeit, ein Suchwort über dem Alphabet inklusive des Leerzeichens einzugeben: { a, b, c, ..., z, _ }. Die Anzahl der Möglichkeiten, die der Angreifer für das Erraten des Suchwortes des Benutzers hat, ist linear zu der Länge des Wortes. Der Angreifer kann sich vorab *Google* Suchanfragen generieren, die er dann mit dem Traffic des Benutzers vergleichen und somit die Benutzersuchanfrage in Erfahrung bringen kann.

Die Autoren des Originalpapers (siehe [1]) gehen allerdings nicht auf die Frage ein, wie relevant die Problematik von Seitenkanalangriffen gegen Web-Anwendungen im Kontext von WLAN Verschlüsselungsprotokollen, wie zum Beispiel WPA bzw. WPA2, wirklich ist. Ein solcher Angriff erscheint schwierig, da ein Benutzer mehrere Onlineaktivitäten gleichzeitig durchführen kann oder da mehrere Personen im gleichen WLAN surfen können. Das bedeutet, dass nicht nur ein Datenstrom, sondern mehrere im Netzwerk vorhanden sind. Es ist fraglich, inwieweit der Angreifer den richtigen Datenstrom ermitteln kann, der zur Web-Anwendung gehört - sprich die Pakete passend zur Web-Anwendung.

4. HERAUSFORDERUNGEN

Es ist offensichtlich, dass Seitenkanalangriffe ein großes Problem darstellen, da es durch einen erfolgreichen Seitenkanalangriff gegen Webanwendungen möglich ist, dass ein Angreifer die Privatsphäre des Benutzers verletzen kann. Eine universelle Abhilfe, das Problem zu lösen, existiert nicht. Wissen über jede individuelle Anwendung ist somit unumgänglich. Zusätzlich stehen Softwareentwickler vor der Herausforderung effektive und effiziente Lösungen zu finden.

Grundsätzlich müssen Entwickler die Sicherheitslücken finden und anschließend müssen Regelwerke spezifiziert werden. Diese beiden Aspekte stellen die technischen Herausforderungen dar. Der Vorgang, um Lösungen gegen Seitenkanalangriffe zu finden, ist abhängig von der jeweiligen Anwendung.

Als erstes müssen Sicherheitslücken identifiziert werden. Notwendig ist das Wissen über den Informationsfluss einer Web-Anwendung, so dass die Identifikation von Sicherheitslücken erleichtert wird. Außerdem sollte dies schon während der Entwicklung und während des Testens einer Web-Anwendung geschehen.

Als Zweites müssen Regelwerke gefunden und durchgesetzt werden. Mit Hilfe des Verständnisses der Entwickler über eine Web-Anwendung ist das Design eines Regelwerkes möglich. Um ein Regelwerk zu spezifizieren und diese Regeln auch durchsetzen zu können, ist die Zusammenarbeit zwischen den Entwicklern der Browser und der Server sowie der Web-Anwendungen zwingend erforderlich.

Letztendlich ist das Wissen über die individuelle Anwendung notwendig. Für Softwareentwickler, die eine Lösung implementieren, ist es erforderlich, jede Web-Anwendung separat zu betrachten. Dazu gehören die Programmstruktur, Zustandsübergänge und Wissen über den Gebrauch der Anwendung. Wie oben erwähnt ist das Verständnis über den Informationsfluss zusätzlich notwendig.

5. VERTEIDIGUNGSMECHANISMEN UND LÖSUNGSANSÄTZE

Es wurde gezeigt, dass Kommunikationsprotokolle und Verschlüsselungsverfahren, wie HTTPS oder WPA/WPA2, nicht ausreichen, um Seitenkanalangriffe zu verhindern.

Allgemeine Lösungsansätze sehen wie folgt aus:

- Padding-Pakete
- Gefälschte, überflüssige Pakete
- Zerlegen von Paketen in Segmente fester Größe
- Zusammenmischen oder Trennen von Zuständen
- Kombination der oben genannten Lösungsansätze

Jedoch ist die Umsetzung und der Einsatz solcher Lösungsansätze nicht immer für Web-Anwendungen realisierbar. Da diese sehr komplex sind, sind spezielle Lösungen für Web-Anwendungen erforderlich. Vor allem müssen Lösungen getrennt voneinander für jede Anwendung entworfen werden, da es zum einem kaum generische Lösungen gibt und zum anderen anwendungsindividuelles Wissen notwendig ist.

Folgender Lösungsansatz ist für Web-Anwendungen sinnvoll: Padding. Dieser ist aber nicht effektiv bei Seitenkanalangriffen gegen Suchmaschinen, sondern bei Seitenkanalangriffen gegen Webseiten, die beispielsweise Features, wie automatische Ergänzungsvorschläge für die Benutzereingabe oder Drop-Down Listen, enthalten. Man kann unterscheiden zwischen Runden und zufälligem Padding. Beim Runden wird die Größe von jedem Paket aufgerundet, so dass das Paket durch eine bestimmte Anzahl von Bytes teilbar ist. Das zufällige Padding meint, dass jedem Paket ein zufällig langes Padding angehängt wird. Allerdings ergibt sich beim Padding das Problem eines zu großen Overheads.

Ein weiterer Lösungsansatz ist das Zusammenmischen von verschiedenen Zuständen. Dies ist vor allem sinnvoll, je länger der Ausführungspfad ist. Beispielsweise gibt es zwei parallele Ausführungspfade - einer sehr kurzer und ein sehr langer Pfad -, wobei beide einen gemeinsamen Anfangszustand und einen gemeinsamen Endzustand haben. Je länger der Pfad ist, desto mehr Zustände sind dazwischen. Der Angreifer kann nun ermitteln, welchen Pfad der Benutzer durchläuft. Denn sobald dieser den längeren Ausführungspfad durchläuft, entstehen bedeutend mehr *Web-Flow* Vektoren, die der Angreifer nutzen kann, um das *Ambiguity Set* zu reduzieren.

Außerdem ist das Produzieren von überflüssigen Paketen, um zusätzliche Zustände zu fälschen, ein weiterer Lösungsansatz. Diese Technik wird vor allem eingesetzt, je kürzer der Ausführungspfad ist. Nimmt man das obere Beispiel eines kurzen und eines langen Ausführungspfades, die parallel verlaufen, das heißt mit gleichem Start- und Endzustand, ist es offensichtlich, dass der Angreifer über die Anzahl der beobachteten *Web-Flow* Vektoren Rückschlüsse über den Inhalt ziehen kann.

Suchmaschinen müssen separat betrachtet werden. Diese stellen ein nur schwer lösbares Problem dar, jedoch wird

doi: 10.2313/NET-2011-07-2_19

sich erst in der Zukunft zeigen, wie groß das Ausmaß ist, inwieweit Suchmaschinen von Seitenkanalangriffen gefährdet sind. Es existieren keine universellen Lösungen, um Seitenkanalangriffe im WLAN gegen Suchmaschinen zu verhindern. Aufgrund der Komplexität von universellen Lösungen können diese hohe Kosten verursachen. Eine realistischere Alternative ist, empfindliche, sicherheitsbedürftige Features zu identifizieren und dafür spezielle Lösungen zu finden. Ein Beispiel für ein Feature von Suchmaschinen, sodass Seitenkanalangriffe möglich sind, ist die automatische Ergänzungsfunktion.

Probleme bei Suchmaschinen im Internet sind erstens, dass sie mit einem enorm hohen Volumen an Daten umgehen. Zweitens ist es fraglich, ob Regelwerke durchgesetzt werden können - dies ist abhängig von der Anwendung, das bedeutet, dass jede Web-Anwendung einzeln zu betrachten ist. Somit ist es schwer Lösungen zu finden, um Seitenkanalangriffe gegen Suchmaschinen im Internet zu verhindern.

Allgemein ist es sinnvoll, sich während des Entwicklungsprozesses jeder einzelner Web-Anwendung der Gefahr eines Seitenkanalangriffs zu widmen. Abbildung 3 zeigt eine mögliche Vorgehensweise für den Entwicklungsprozess einer individuellen Anwendung zur Identifizierung von Sicherheitslücken und Definition von Regelwerken. Hierbei ist es von Vorteil, mit einem Traffic-Analysetool zu arbeiten. Die Abbildung beschreibt den Entwicklungsprozess wie folgt: Jede Web-Anwendung hat individuelle Sicherheitsziele, um die Privatsphäre zu schützen. Sowohl statisch als auch dynamisch muss eine Flussanalyse durchgeführt werden, um den Informationsfluss (Daten-, Kontrollfluss und *Web Flow*) von sicherheitsbedürftigen Daten zu verfolgen und damit Verstöße gegen Sicherheitsziele zu finden. Hierbei kann ein automatisches Tool hilfreich sein. Falls Sicherheitslücken gefunden werden, muss der Entwickler herausfinden, ob diese Lücke durch eine geeignete Erweiterung der Regeln, wie zum Beispiel Padding, geschlossen werden kann. Wenn dies der Fall ist, müssen neue Regeln spezifiziert werden. Ansonsten muss das Design der Anwendung angepasst werden. Der beschriebene Vorgang für einen Entwicklungsprozess ist für jede Anwendung separat durchzuführen.

Grundsätzlich sind die Kosten für individuelle Lösungen enorm, so dass die Entwicklung von automatischen Tools unumgänglich ist.

6. FAZIT

Seitenkanalangriffe stellen in der heutigen Zeit eine extreme Gefahr für personenbezogene Daten dar. Mit Hilfe von Seitenkanal-Informationen - Attributen einer verschlüsselten Datenübertragung - kann der Angreifer Rückschlüsse über die Eingaben des Benutzers ziehen. Dies geschieht unter Verwendung der aus der Benutzereingabe resultierenden Ergebnisse.

Auch wenn das Finden von Verteidigungsmechanismen schwer erscheint, ist dies notwendig, da das Problem von Seitenkanalangriffen gegen Web-Anwendung sehr aktuell ist. Zuerst müssen Sicherheitslücken identifiziert und anschließend Regelwerke definiert werden, um die Gefahr von Seitenkanalangriffen zu reduzieren.

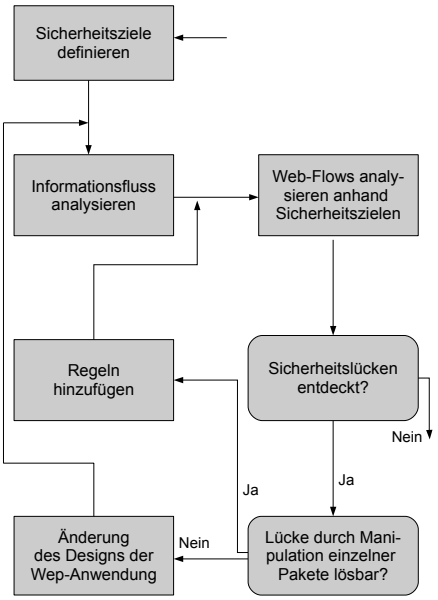

Abbildung 3: Vorgehensweise während eines Entwicklungsprozesses einer Web-Anwendung, wie in Referenz [1]

In Referenz [1] wird von den Autoren behauptet, dass keine generischen Lösungen existieren, so dass Seitenkanalangriffe gegen Web-Anwendungen abgewehrt werden können. Allerdings wird nicht explizit beschrieben, warum es keine generischen Lösungen gibt. Denkbar wäre es zum Beispiel, dass man unabhängig von der Benutzereingabe pro Zeiteinheit immer die gleiche Datenmenge versendet.

Da individuelle Lösungen außerdem sehr teuer sind, müssen automatische Tools entwickelt werden. Solche Tools sollten eingesetzt werden, um während des Entwicklungsprozesses mögliche Schwachstellen zu erkennen. Zusätzlich sollen sie als Hilfsmittel dienen, um den Sourcecode einer Web-Anwendung zu analysieren und den Umfang zu messen, inwieweit interne Daten sichtbar sind.

Fraglich ist, wie groß das Ausmaß der Gefahr von Seitenkanalangriffen gegen Webanwendungen in der Zukunft wirklich ist. Es ist außerdem abzuwarten, wie zukünftige Entwicklungen aussehen werden, um die Gefahr von Seitenkanalangriffen zu verringern und ob diese Entwicklungen es wirklich ermöglichen können, effektiv und effizient vor Seitenkanalangriffen zu schützen.

7. LITERATUR

[1] S. Chen, R. Wang, XF. Wang, K. Zhang: *Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow*, pp.191-206, Proc. IEEE Symposium on Security and Privacy, 2010

[2] J.J. Quisquater: *Side-Channel Attacks*, 2002, `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf`, zugegriffen am 20.05.2011

[3] M. Witteman, M. Oostdijk: *Secure Application*

doi: 10.2313/NET-2011-07-2_19

Programming in the Presence of Side Channel Attacks, RSA Conference, Riscure, Netherlands, 2008, `http://www.riscure.com/fileadmin/images/Docs/Paper_Side_Channel_Patterns.pdf`, zugegriffen am 20.05.2011

[4] K. Tiri: *Side-Channel Attack Pitfalls*, Proc. of the 44th annual Design Automation Conference, Platform Validation Architecture, San Diego, California, USA, 2007

[5] E. Rescorla, A. Schiffman, *The Secure Hypertext Transfer Protocol*, RFC 2660, August 1999, `http://www.apps.ietf.org/rfc/rfc2660.html`, zugegriffen am 20.05.2011

[6] P. Anderson, *What is Web 2.0? - Ideas, Technologies and Implications for Education*, JISC, Technology and Standard Watch, 2007

[7] K. Zhang, Z. Li, R. Wang, XF. Wang, S. Chen *Sidebuster: Automated Detection and Quantification of Side-Channel Leaks in Web Application Development*, Proc. of the 17th ACM conference on Computer and communications security, Chicago, Illinois, USA, 2007

[8] D. F. Song, D. Wagner, X. Tian *Timing Analysis of Keystrokes and Timing Attacks on SSH*, Proc. of the 10th conference on USENIX Security Symposium, Vol. 10, University of California, Berkeley, USA, 2001

[9] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, G. M. Masson *Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations*, IEEE Symposium on Security and Privacy, pp.35-49, John Hopkins University, Department of Computer Science, Baltimore, USA, 2008

[10] C. Eckert, *IT-Sicherheit: Konzepte - Verfahren - Protokolle*, 6. Auflage, Oldenburg Wissenschaftsverlag GmbH, ISBN 978-3-486-58999-3, München, 2009

[11] IEEE Standards *802.11i-2004: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, New York, USA, 2004

doi: 10.2313/NET-2011-07-2_19