sind die Verkehrsdaten nach § 96 Abs. 2 TKG unverzüglich zu löschen, sofern sie nicht zum Zweck des Aufbaus weiterer Verbindungen, zur staatlich angeordneten Telekommunikationsüberwachung oder zur Abwehr eines konkret vorliegenden Anfangsverdachts von Missbrauch benötigt werden.
Ein Verstoß gegen fernmelderechtliche Vorschriften (z.B. durch Nichtzustellung eingegangener E-Mails) ist nach § 206 StGB strafbar. Die Ausfilterung virenverseuchter Mails ist jedoch zum Schutz gegen unerlaubte Zugriffe unter Ausnutzung von § 109 Abs. 1 Nr. 2 TKG erlaubt. Für die Abwehr von SPAM-Mails wird dagegen i.d.R. die Einwilligung des Betroffenen (über eine entsprechende Policy) benötigt.

Datenschutz

Bei der Nutzung elektronischer Kommunikationsmedien fallen mit den Verbindungsdaten (BVerfG-Urteil vom 2. März 2006) bzw. den Nutzungs-Logdaten personenbezogene Daten an. Deshalb dürfen längerfristig nur Beginn und Ende einer Telekommunikationsverbindung und die verursachten Kosten mitprotokolliert werden (BAG-Urteil vom 27. Mai 1986). Einschränkungen gelten zudem, wenn der Beschäftigte gegenüber seinem Kontaktpartner zur Geheimhaltung nach § 203 StGB verpflichtet ist (BAG-Urteil vom 13. Januar 1987).
Wird die Nutzung des Internets nicht den Mitarbeitern in Rechnung gestellt, sind die anfallenden Logdaten auf der Grundlage von § 4 Abs. 4 Ziffer 2 TDDSG bzw. § 18 Abs. 4 Ziffer 2 MDStV unmittelbar nach Beendigung der Nutzung des aufgerufenen Teledienstes bzw. Mediendienstes zu löschen. Zudem unterliegt die Protokollierung der Verbindungs- und Nutzungsdaten nach § 31 BDSG einer strengen Zweckbindung.

Betriebliche Mitbestimmung

Eine Betriebsvereinbarung ist eine vorrangige Rechtsvorschrift und gilt für die Betriebsangehörigen unmittelbar und zwingend. Wenn darin die Erhebung, Verarbeitung oder Nutzung personenbezogener Daten gestattet ist, so ist dies datenschutzrechtlich grundsätzlich zulässig (BAG-Beschluss vom 27. Mai 1986). Eine Betriebsvereinbarung ersetzt insofern kollektivrechtlich erforderliche Einwilligungserklärungen und vereinfacht damit die zu treffenden Regelungen maßgeblich. So ist eine Kontrolle der Nutzung möglich, wenn sie betriebsintern (etwa durch eine Betriebsvereinbarung) mengenmäßig oder zeitlich beschränkt und dies den Arbeitnehmern bekannt gegeben wurde.
Die Protokollierung elektronischer Kommunikation dient der Verhaltenskontrolle der Arbeitnehmer und unterliegt daher der Mitbestimmung. Allerdings bestimmt der Arbeitgeber über Einsatz und Nutzung seiner Arbeitsmittel und darf dies insbesondere zur Kostenkontrolle und zur Optimierung der Telekommunikationsanlagen überwachen. Die entsprechenden Maßnahmen müssen dabei zweckmäßig und verhältnismäßig sowie den Arbeitnehmern bekannt sein. Eine vollständige Überwachung oder Aufzeichnung ist unzulässig.
Dem Arbeitgeber ist es nach § 100 Abs. 1 TKG zur Erkennung, Eingrenzung und Beseitigung von Störungen und Fehlern an Telekommunikationsanlagen bzw. nach § 107 Abs. 2 TKG zur Vermeidung von Fehlübermittlungen und unbefugtem Offenbaren gestattet, entsprechende Analysen des Netzwerktraffics vorzunehmen und damit ungewöhnliche Datenströme oder ungewöhnlich hohe Datenvolumina festzustellen.

Bernhard C. Witt
(Diplom-Informatiker, Berater für Datenschutz und IT-Sicherheit bei der it.sec GmbH & Co. KG, geprüfter fachkundiger Datenschutzbeauftragter und Lehrbeauftragter an der Universität Ulm für Datenschutz und IT-Sicherheit)

Web Application Security

Cross-Site Scripting sperrt Benutzer aus

Cross-Site Scripting, kurz XSS, ist die absolute Nummer 1 unter den Sicherheitsproblemen in Webanwendungen – in neun von zehn von uns untersuchten Websites treffen wir sie an. Mal ganz offensichtlich im Eingabefeld für die Suche, mal an versteckter Stelle und nur für den Experten erkennbar.

Mit XSS lässt sich gleich eine ganze Handvoll von Angriffen führen: *Session-Hijacking* und die Variante *Session-Fixation*, die das Eindringen in ein fremdes Konto ermöglichen, *Website-Spoofing*, also das Fälschen von Webseiten, was z.B. das Phishing erleichtert, die Umgehung der in den Browsern eingebauten *Same-Origin-Policy*, welche eine Webseite davor schützt, dass vertrauliche Daten von Dritten ausgelesen werden können.
Und schließlich das *Session Lockout*, mit dem ein Angreifer verhindern kann, dass ein Benutzer sich in eine Webanwendung einloggen kann. Diese Angriffstechnik ist im Unterschied zu den anderen genannten Schwachstellen unseres Wissens nach noch nirgendwo beschrieben. Wir wollen wir sie uns daher in diesem Artikel näher ansehen. Zur Erklärung der anderen, durch XSS verursachten Schwachstellen und für Hinweise, wie sie vermieden werden, verweisen wir auf den "Maßnahmenkatalog und Best Practices für die Sicherheit von Webanwendungen" des Bundesamts für Sicherheit in der Informationstechnik (BSI) [1]. Im Folgenden wird außerdem die Kenntnis der Funktionsweise von XSS vorausgesetzt, in [2] kann dies nachgelesen werden.
Wie Anwendungen mit ungültigen SessionIDs umgehen.
Um Session Lockout zu verstehen, müssen wir zunächst einen Blick auf folgenden Aspekt des Session Handlings werfen:
Eine typische Webanwendung mit Login setzt irgendwann während des Login-Prozesses ein *SessionID*-Cookie im Browser des Benutzers. Hat der Benutzer sich korrekt angemeldet, so verwendet die Anwendung für alle folgenden Aufrufe nur noch die

SessionID, um den Benutzer zu identifizieren. Was passiert nun, wenn der Browser der Anwendung eine nicht existierende SessionID sendet? Die Anwendung kann den Request keiner bestehenden Sitzung zuordnen und geht in der Regel davon aus, dass hier ein Benutzer zugreift, der sich vor langer Zeit einmal eingeloggt hat und dessen Session nun nicht mehr gültig ist. Sie löscht oder überschreibt das alte SessionID-Cookie im Browser und fordert den Benutzer zur erneuten Anmeldung auf, verzweigt also erneut zur Loginseite. Die Altlast ist damit bereinigt, ohne dass der Benutzer behindert worden wäre. Wenn jedoch, aus Gründen, die wir gleich erkennen werden, das Cookie sich nicht löschen oder überschreiben lässt, es also immer wieder an die Anwendung geschickt wird, dann ist der Benutzer gefangen in dieser Ablehnungsschleife: Die Anwendung verzweigt zur Loginseite und setzt das neue SessionID-Cookie, dieses gelingt jedoch nicht und es kommt erneut die ungültige SessionID bei der Anwendung an, worauf wieder der Login angefordert wird.

Ein Angreifer kann genau diese Situation in vielen Fällen geschickt herbeiführen. Erforderlich ist dazu lediglich das Vorhandensein einer XSS-Schwachstelle. Das Vertrackte für den, der einen solchen Angriff verhindern will, ist dabei, dass die XSS-Schwachstelle sich nicht notwendigerweise in der Anwendung selbst oder auf dem Host, auf dem sie läuft, befinden muss. Sie kann auf einem beliebigen Host innerhalb derselben Domain auftreten. Also zum Beispiel auch auf dem Server laengst-vergessener-alter-testhost.firma.de.

Angriff

Wir schauen uns dieses Beispiel an: Die Anwendung A ist über den Link https://b2b.firma.de/AnwA erreichbar. Sie benutzt für das Session-Management ein Cookie namens JSESSIONID, welches sie als sogenanntes *Host-Cookie*, also mit einer Gültigkeit nur für den Host b2b.firma.de beim Login definiert. Wir nehmen zusätzlich an, dass sich auf dem Host test.firma.de eine XSS-Schwachstelle unter
http://test.firma.de/testscript?Suche=
XSS befindet.

Und so läuft der Angriff ab:

Ein Angreifer schickt in einer E-Mail unter Ausnutzung der XSS-Schwachstelle (in der Suchfunktion) folgenden Link an den Benutzer:

http://test.firma.de/testscript?Suche=
<script>document.cookie="JSESSIONID=123; domain=firma.de; path=/; expires=Saturday, 31-Dec-09 23:59:59 GMT;" </script>

Klickt der Benutzer auf den Link, so wird in seinem Browser ein Cookie namens JSESSIONID für alle Hosts und Anwendungen in der Domain firma.de gesetzt. Das Cookie bleibt bis zum 1.1.2010 gültig, sofern es nicht explizit gelöscht wird. Verantwortlich dafür sind die im Befehl zum Setzen des Cookies, der in obigem Link übergeben wird, befindlichen Attribute.

Wegen der Gültigkeit über alle Hosts einer Domain wird ein solches Cookie *Domain-Cookie* genannt.

Die Anwendung setzt und löscht demgegenüber jedoch typischerweise, und so auch in diesem Fall, ein *Host-Cookie*. Und das enthält als domain-Attribut den Host www.firma.de.

Entscheidend ist nun, dass für den Browser Host-Cookie und Domain-Cookie verschiedene Cookies sind, obwohl sie denselben Namen JSESSIONID tragen. Wenn die Anwendung das Cookie manipuliert oder löscht, dann operiert sie immer auf der zweiten der beiden JSESSIONIDs. D.h. die erste, also die vom Angreifer gesetzte, bleibt unangetastet. Der Browser schickt mit jedem Zugriff auf die Anwendung immer beide Cookies. Im HTTP-Header sieht das so aus:

Cookie:
JSESSIONID=123;JSESSIONID=<SESSIONID>

Wie geht nun die Anwendung mit einer solchen Situation um, für welches Cookie entscheidet sie sich? In vielen Fällen nimmt sie einfach das erste Cookie – und damit die ungültige SessionID. Wenn das Cookie dann, nach dem Zwischenschritt des Logins, aber mit einer frischen SessionID versorgt, so erwischt sie wieder die erste von den beiden. Da die Reihenfolge auf diese Weise erhalten bleibt, geht das Spiel endlos weiter.

Resümee

Zusammengefasst ergibt sich: Verwendet eine Anwendung Cookies als Träger der SessionID und behandelt die ungültige SessionIDs in der 'natürlichen' Weise, also etwa so, wie hier gezeigt, dann ist jeder Benutzer dieser Anwendung in der hier gezeigten Weise angreifbar, sofern auf irgendeinem Host in derselben Domain eine XSS-Schwachstelle vorhanden ist. Das betrifft unseren Erkenntnissen nach gegenwärtig eine Vielzahl von Webanwendungen, auch das Online-Banking einiger Banken.

Session Lockout verhindern

Wie kann der Entwickler einer Webanwendung das Problem lösen? Die enttäuschende Antwort lautet: Es gibt keine einfache und saubere Lösung. Hier aber ein paar Vorschläge, die je nach Anwendungsfall helfen können:

- Dafür sorgen, dass keine XSS-Schwachstellen auftreten.
- Nicht nur das selbst gesetzte Cookie löschen, sondern auch ein gleichnamiges Host-Cookie.
- Alle erhaltenen SessionIDs durchprobieren und erst dann abbrechen, wenn keine gültige dabei ist.

Abschließend noch ein praktischer Rat: Jedes Helpdesk für Webanwendungen sollte für einen Anwender, der sich beklagt, dass er sich nicht mehr einloggen kann, den Hinweis parat haben, erstmal alle Cookies zu löschen.

Thomas Schreiber, ts@securenet.de, ist Berater für Web Application Security und Geschäftsführer der SecureNet GmbH.

Quellen
[1] "Maßnahmenkatalog und Best Practices für die Sicherheit von Webanwendungen", Bundesamt für Sicherheit in der Informationstechnik (BSI), http://www.bsi.de/literat/studien/websec/WebSec.pdf
[2] "Stichwort Phishing", http://www.securenet.de/papers/Phishing.html