IT-SICHERHEIT praxis

Recht und Praxis · Anwendungs-Tipps für IT-Sicherheit und Datenschutz

1

1. Jahrgang 2006

Web Application Security

Session **Riding**

In diesem ersten Artikel der Reihe „IT-SICHERHEITpraxis" wollen wir uns eine Angriffsmöglichkeit näher ansehen, für die erstaunlich viele Webanwendungen anfällig sind - und erklären, wie man den Angriff verhindert. Die Technik wird als Session Riding bezeichnet und das Ausmaß, in dem Schaden angerichtet werden kann, ist beträchtlich.

Der Angriff läuft so ab: Angreifer schickt präparierten Link auf Anwendung X an Benutzer – Benutzer ist in Anwendung X eingeloggt und klickt auf Link – Klick löst die vom Angreifer vorgesehene Operation auf Anwendung X aus – Benutzer merkt davon in der Regel nichts. Potentiell betroffen von derartigen Angriffen ist jede Webanwendung, die nicht besondere Vorkehrungen dagegen getroffen hat. Und, wie sich zeigt, auch viele browserbasierte Administrationskonsolen, mit denen so sensible Objekte wie Router, Firewalls oder Datenbanken gesteuert und verwaltet werden.

Session Riding im Detail

Als Beispiel für die detaillierte Beschreibung von Session Riding wollen wir eine solche Admininstrationsanwendung hernehmen, nämlich den Webmin [2], mit dem sich Unixsysteme mit dem Browser konfigurieren lassen und der anfällig ist für Session Riding. Wir schauen uns die webmin-Funktion zur Verwaltung der /etc/hosts Datei an, konkret, wie Webmin den Eintrag eines neuen Hosts vornimmt.

Der Admin muss sich zunächst in webmin einloggen und gelangt über einige Navigationsklicks auf diese Seite, die den aktuellen Inhalt der /etc/hosts anzeigt:

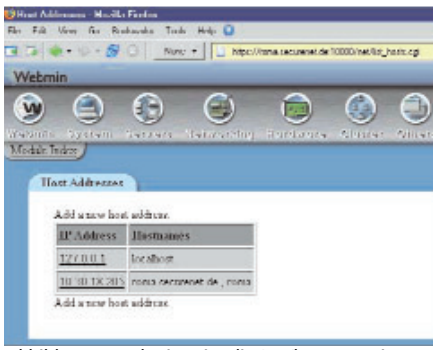

Abbildung 1: Webmin zeigt die /etc/hosts-Datei an.

Ein Klick auf „Add a new host address" führt zum Formular zur Eingabe des Hosts.

Der Admin trägt die neue Host-Zuordnung ein und schickt sie ab. Folgender GET-Request wird daraufhin vom Browser abgesetzt:

https://roma.securenet.de:10000/net/save_host.cgi?new=1&address=192.168.0.1&hosts=b2bportal.firma.com

Webmin nimmt die Eintragung in der /etc/hosts vor und zeigt als Antwortseite den neuen Inhalt an. (siehe Abb. 2)

Woher weiß Webmin, welcher Benutzer hier zugegriffen hat? Ein Parameter, aus dem der Benutzer oder die Session hervorgeht, ist im Request ja nicht

Schwerpunkte

Editorial

Web Application Security ist ... nein, erklären wir es am Beispiel: Wenn Angreifer mit dem Browser Zugangsdaten von Kunden einsehen und verändern (der T-Com ist es passiert), oder unzählige Kreditkartendaten stehlen (im vergangenen Jahr bei der amerikanischen Firma Cardsystems geschehen), oder auf fremde Rechnung online Shoppen gehen (wie kürzlich bei Quelle)... dann haben sie eine Schwachstelle in einer Webanwendung entdeckt und ausgenutzt. Die Web Application Security hat es sich zur Aufgabe gemacht, dem Einhalt zu gebieten.

Diese noch recht junge, aber dennoch weit fortgeschrittene IT-Disziplin hat einiges zu bieten: Guidelines und Best Practices Anleitungen – z.B. die vom Bundesamt für Sicherheit in der Informationstechnik (BSI)* –, ein Klassifizierungsschema, das sich WASC (Web Application Threat Classification) nennt, spezialisierte Application Firewalls, die auch auf Anwendungsebene eine zentrale Kontrollfunktion bereitstellen, eine Vielfalt an Tools, die den Entwickler, Verantwortlichen und externen Tester beim Aufspüren von Sicherheitsmängeln unterstützen.

Zeit also, diesem Thema verstärkte Aufmerksamkeit zu widmen.

Wir werden uns in den nächsten Ausgaben daher mit den unterschiedlichen Aspekten (un)sicherer Webanwendungen befassen, von der Darstellung der schier unglaublichen, weit verbreiteten Sicherheitslücke in der vorliegenden Ausgabe, über innovative Lösungsansätze, bis hin zu handfesten Best Practices im Schlußartikel in 6 Monaten.

Seien Sie gespannt!

Weitere praktische Hilfestellungen der IT-SICHERHEITpraxis gibt es – heute und in den kommenden Ausgaben – auch zu den folgenden Themenbereichen:
Netzwerksicherheit, Compliance, Imaging, Datenschutz und Content Security.

Thomas Schreiber, Geschäftsführer SecureNet GmbH

*http://www.bsi.de/literat/studien/websec/WebSec.pdf

DATAKONTEXT FACHVERLAG

Abbildung 2: Der Inhalt der /etc/hosts-Datei nach dem Eintrag

vorhanden. Die Antwort: Diese Information wurde von Webmin nach dem Login in Form der SessionID sid im Cookie hinterlegt:
sid=49c9a
3e2ff1c
46700b4a8
c627c748a12

Form-Based Authentication mit Cookies

Zur Verfolgung der Session benutzt Webmin dasselbe Verfahren wie unzählige andere Webanwendungen auch, nämlich die „form-based Authentication" mit Cookies als Träger der Session-Information. Einmal gesetzt, schickt der Browser dieses Cookie nun bei jedem Request an den Server automatisch mit, ohne dass Benutzer oder Webanwendung etwas dazu beitragen müssen. Diese Funktionsweise, die letztlich eine fundamentale Schwäche der Webtechnologie darstellt, wird beim Session Riding ausgenutzt.

Um das zu erkennen, fragen wir uns zunächst, was passieren würde, wenn unser Admin, statt das Formular zu verwenden, obigen Link in die Adresszeile des Browser eintippen und abschicken würde? Richtig, der Eintrag in die /etc/hosts würde in gleicher Weise vorgenommen wie im ersten Fall – das sid-Cookie fügt der Browser auch in diesem Fall automatisch hinzu. Und wenn der Administrator, noch immer eingeloggt in Webmin, dem Link in einer E-Mail folgen würde und dann auf der angezeigten Seite auf den darin enthaltenen Webmin-Link klicken würde? Auch in diesem Fall würde der Aufruf von Webmin zur Ausführung kommen, und auch in diesem Fall würde der Browser automatisch das SessionID-Cookie einsteuern, so dass der Request im Kontext des Admins ablaufen und somit der Eintrag erfolgen würde.

Daraus ergibt sich z.B. folgendes, sicher nicht sehr weit hergeholtes Angriffsszenario: Angreifer X weiß, dass Admin A im Unternehmen U Hosts und Netzwerke mit webmin administriert und er kennt den für ihn interessanten Zielhost. Um dort eine Modifikation der Namensauflösung vorzunehmen – etwa, um den Zugriff auf den Server, von dem regelmäßig Softwareupdates geladen werden, umzulenken – muss es X nun lediglich gelingen, im Browser von A den besagten Link zur Ausführung zu bringen. Die wohl unauffälligste Möglichkeit, dies zu erreichen, benutzt das IMG-Tag von HTML. Der Angreifer erstellt dazu eine HTML-E-Mail, die das folgende vermeintliche Bild anzieht:
<img src="https://roma.securenet.de:10000/net/save_host.cgi?new=1&address=192.168.0.1&hosts=b2bportal.firma.com">

In dem Moment, wo A sich diese E-Mail nur ansieht, setzt der Browser den Request zum Laden des Bildes ab und löst so die Modifikation an der Hostsdatei aus. Einzige Voraussetzung: A muss im Moment der Ausführung in Webmin eingeloggt sein (Einstellungen von Browser und E-Mail-Client können die hier beschriebene „elegante" Form der Ausnutzung verhindern; geeignete Modifikationen des Angriffs würden aber in vielen Fällen trotzdem zum Gelingen führen). Dass der Browser hier kein Bild darstellen kann, weil unerwartete Zeichen geliefert werden, spielt keine Rolle, im Gegenteil, die Antwortseite bleibt auf diese Weise sogar unsichtbar für den Benutzer.

Wir sehen, um auf das eingangs skizzierte Szenario zurückzukommen, allein das Öffnen einer E-Mail reicht aus, um über eine mit einer Session Riding-Schwachstelle behafteten Webanwendung im Intranet möglicherweise großen Schaden anzurichten. Im Normalfall sind die Möglichkeiten für einen Angriff per Session Riding weitaus größer als im hier beschriebenen, stark abgesicherten Beispiel, insbesondere wenn es sich um Webanwendungen im Internet handelt.
Unseren Untersuchungen nach stellt Session Riding wegen der weiten Verbreitung – neben dem Cross-Site Scripting – gegenwärtig die größte potentielle Bedrohung im Bereich der Web Application Security dar.

Session Riding verhindern

Um Session Riding wirkungsvoll zu verhindern, ist neben der SessionID ein weiterer geheimer Parameter einzuführen – das sog. Secret –, der jedoch nicht mit dem Cookie, sondern als herkömmlicher Form-Parameter übertragen wird. Der Parameter ist beim Login zu erzeugen und in den Session-Daten zu hinterlgen. Er muss eine ausreichend große „Randomness" besitzen, also hinreichend groß und zufällig sein, so dass er nicht erraten werden kann. Der GET-Request zur Modifikation der hosts-Datei hätte damit dieses Aussehen https://roma.securenet.de:10000/net/save_host.cgi?new=1&address=192.168.0.1&hosts=b2bportal.firma.com&secret=a9823knf9
Das **Secret** wird mit jedem Zugriff mitgeschickt und die Anwendung prüft, ob es identisch ist mit dem in der Session hinterlegten. Ist das nicht der Fall, so wird der Request abgelehnt. Ein Angreifer müsste nun das Secret wissen, um Session Riding durchführen zu können. Das kann er jedoch nicht, wenn nicht widerum andere Schwachstellen ihm dies ermöglichen sollten.
Da die APIs, über die professionelle Webanwendungen das Session Handling abwickeln lassen, die Verwendung von Secrets noch nicht transparent unterstützen, ist ein solches Verfahren mit erheblichem Zusatzaufwand verbunden. Dafür bekommt die Anwendung aber gleichzeitig einen sehr wirksamen Schutz vor einer Vielzahl weiterer Bedrohungen, unter ihnen Session-Hijacking oder Session-Fixation.

Wird statt Cookies die alternative Technik des URL-Rewritings zum Transport der SessionID verwendet, bei der die SessionID im URL übergeben wird, so ist die Anwendung im übrigen ebenfalls sicher vor Session Riding.

Thomas Schreiber, Geschäftsführer SecureNet GmbH

Quellen
[1] "Session Riding – A Widespread Vulnerability in Today's Web Applications", Thomas Schreiber, SecureNet, http://www.securenet.de/papers/Session_Riding.pdf
[2] http://www.webmin.com/