Sicherheitsprobleme in der Programmierung

Tonnerre Lombard

18. Oktober 2007

1 Mythen der Sicheren Programmiersprache

In der Programmierung gilt Sicherheit oft als ein von Schamanen betriebenes und mit Zauberkraft gesichertes Geheimnis. Viele Leute predigen verschiedene Wege, sicheren Code zu schreiben. Die meisten dieser Wege laufen auf die verwendung bestimmter Programmiersprachen hinaus. Im Zweifelsfall laufen die Argumentationen jedoch in's Leere. Einige dieser leeren Versprechungen werden im ersten Teil genauer beleuchtet und im Laufe des Textes widerlegt. Dies umfasst die Verwendung von Skriptsprachen, alternativen Bytecodes sowie Hoch- und Niedersprachen.

2 Arten von möglichen Fehlern

Wie nicht anders zu erwarten, gibt es in der komplexen Welt der Programmierung viele verschiedene Dinge, welche man falsch machen kann.

2.1 Buffer Overflow

Ein Buffer Overflow ist eine sehr grundlegende Art von Fehlern, welche aus der Art und Weise resultiert, wie die Daten ausgeführter Programme im Speicher angeordnet werden. Es gibt dabei praktisch zwei verschiedene Arten von Buffer Overflows: Stack Overflows und Heap Overflows. Beiden ist gemeinsam, dass über den vorgesehenen Speicherbereich hinaus geschrieben werden kann, wodurch zur Programmausführung wichtige Daten manipuliert werden. Auf diese Art kann die Ausführung beliebigen Codes erzwungen werden.

2.2 Synchronisierungsprobleme

Wann immer Code parallel ausgeführt wird, welcher auf dieselben Dinge zugreift, kann es zu Problemen kommen. Dies fängt beim Sperren von geöffneten Dateien an und geht über den parallelen Zugriff auf Daten zwischen Threads

bis hin zur Signalbehandlung. Wann immer der Programmablauf keinen roten Faden darstellt, ist eine Form von Synchronisierung vonnöten.

2.2.1 Fehlende Parallelisierung bei geteilten Zugriffen

Greifen mehrere Prozesse auf dieselbe Ressource zu, können unter Umständen verschiedene sicherheitskritische Situationen entstehen, welche durch Angreifer ausnutzbar sein könnten. Voraussetzung dazu ist lediglich fehlende Synchronisierung der Prozesse.

Ebenfalls in diese Kategorie fallen Angriffe, bei denen ein Prozess Objekte mit den falschen Berechtigungen erstellt und diese nachträglich ändert – und somit einen Zeitraum schafft, während dem sich andere prozesse Rechte an dem Objekt sichern können.

2.2.2 Fehlende Threadsynchronisation

In der Synchronisation zwischen Threads ist das Potential für Probleme noch viel grösser, da sie nicht über getrennte Speicherbereiche verfügen. Die Verwendung reentranter Funktionen spielt hier eine grosse Rolle.

2.2.3 Signalbehandlungsangriffe

Eine weitere, oft unterschätzte Form asynchroner Programmausführung sind Signale, und auch diese können unter Umständen zur Codeausführung verwendet werden.

2.3 Formatstringangriffe

Mit Formatstringangriffen kann in den meisten Fällen erst einmal nur Speicher gelesen werden, aber auch dieser kann bereits interessante Informationen enthalten.

2.4 Injectionangriffe

Wann immer mehrere Sprachen ineinander eingebettet werden, ist es ratsam, dafür zu sorgen, dass Elemente der inneren Sprache nicht mit Elementen der äusseren Sprache gemischt werden. Dieses Problem ergibt sich auch und vor Allem bei benutzerkontrollierten Eingaben in Applikationen, welche in der Ausgabe der Applikation oder in erzeugten Befehlen repräsentiert werden.

2.4.1 Formatinjektion

Formatinjektionen sind die älteste Art von Injection-Sicherheitslücken. Hierbei werden die Begrenzungszeichen eines Formates in einem eingefügten, nicht ge-

prüften Teil verwendet, so dass zusätzliche Daten eingefügt werden. In einem Beispiel wird ein Rootaccount angelegt, wobei der Anlegende lediglich über Benutzerrechte verfügt.

2.4.2 Cross Site Scripting (XSS)

Cross Site Scripting ist ebenfalls das Einbetten von Informationen in eine Sprache in die sie nicht hinein gehören, um JavaScript-Elemente auf Seiten einzublenden, auf die sie nicht gehören, um Kontrolle über die Inhalte zu erlangen.

2.4.3 SQL injection

In diesem Teil wird die Natur der SQL-Injection-Sicherheitslücken erläutert, inklusive Codebeispielen wie eine solche Sicherheitslücke zustande kommt.

2.5 Authentisierungs- und Verifikationsmängel

Eine ganz eigene Klasse von Fehlern liegt in der Logik der Applikation versteckt. Oft werden hier Sicherheitsmerkmale vergessen oder nicht vollständig ausgeführt, oder sie werden aus unsicheren Elementen zusammengesetzt.

2.5.1 Berechtigungsprobleme auf Objekte

Probleme mit den Berechtigungen auf Objekte, welche von mehreren Prozessen gesehen werden können, sind immer wieder eine grosse Fehlerquelle – vor Allem, da zum Beispiel die Benutzerrechte auf Dateien oft nicht nur vom entsprechenden Programm verwaltet werden. Was hierbei zu beachten ist und wie man mit renitenten Benutzern umgeht, wird in diesem Kapitel erläutert.

2.5.2 Unauthentisierte Interfaces

In einigen wenigen Fällen besteht das Sicherheitsproblem darin, dass die Authentisierung oder Autorisierung für ein Interface nicht geprüft wird. Dieser Teil erwähnt den Fall allerdings bloss, da er mehr oder weniger selbsterklärend sein sollte.

2.5.3 Sessiondiebstahl

Eine einfache Möglichkeit, an den Account einer anderen Person zu kommen, sei es um Daten auszuspähen, die Person zu personifizieren, oder um deren Berechtigungen zu missbrauchen, sind oft laufende Sitzungen der Person ein Angriffsziel. Mit Codebeispielen wird darauf eingegangen, auf welchen Wegen man eine Sitzung einer anderen Person übernehmen kann.

Mittels SQL-Injection

Es gibt mehrere Methoden, SQL-Injection auszunutzen, um Zugriff auf fremde Accounts zu erhalten. Ein paar Beispiele werden im Code dargestellt.

Mittels XSS

Hierbei wird darauf eingegangen, wie man mittels Cross Site Scripting das Session-Cookie einer Webseite entwenden kann.

Bei schlechtem Generator

Einige Fälle von Sessiondiebstahl sind auch einfach auf schlecht generierte Cookies zurückzuführen. Es wird beleuchtet, welche Methoden zur Generierung von Session-Cookies als sicher angenommen werden können und welche gar nicht in Frage kommen.

2.5.4 Cross Site Request Forgery (CSRF)

Der modernste unter den modernen Angriffen nennt sich Cross Site Request Forgery. Hierbei wird eine Aktion durch einen bereits angemeldeten Benutzer von einer anderen Seite aus ausgelöst.

3 Spezielle Probleme mit 32-Bit-Code

Dieser letzte Teil des Vortrages behandelt einige Probleme, die nur speziell dann auftreten, wenn Code auf 64-Bit-Prozessoren ausgeführt wird, welcher 32-Bit-Spezifika aufweist.

4 Abschliessende Hinweise

Zuletzt werden noch einige Hinweise zur Architektur sicherer Systeme gegeben. Dies reicht von erneuter Mahnung zum Prüfen gegen Buffer Overflows bis zum Hinweis, wie SSL-Clientzertifikate die nervigen Cookieprobleme ein für alle mal beseitigt werden können.