Reaktive Sicherheit (WS 2007/2008)

Dr. Ulrich Flegel, Dr. Michael Meier

Betreuung der Übungen: Armin Büscher

Übungsblatt Nr. 5

Aufgabe 1 (Good News: Quellcode-Analyse, Compiler-Virus)

In Ken Thompson's Artikel *Reflections on Trusting Trust* (vgl. Übungsblatt 4, Aufgabe 4) wurde beschrieben, wie ein Compiler so modifiziert werden kann, dass er ein Trojanisches Pferd in kritische Software einschließlich sich selbst einfügt. Wenn dieser Angriff (die Compiler-Modifikation) unbemerkt bleibt, dann ist auch durch vollständige Analyse des Systemquellcodes der laufende maliziöse Code nicht zu erkennen. Lange Zeit waren keine praktikablen Methoden bekannt, um diesen Angriff zu erkennen. 2005 stellte David A. Wheeler eine praktikable Technik zur Erkennung dieses Angriffs vor. Erläutern Sie die als *Diverse Double-Compiling* bekannte Technik.

Literatur:

- `http://blog.slash-me.net/archives/139-Reflections-on-Trusting-Trust-revisited.html`
- `http://www.schneier.com/blog/archives/2006/01/countering_trus.html`
- David A. Wheeler: Countering Trusting Trust through Diverse Double-Compiling. ACSAC 2005. `http://www.acsac.org/2005/abstracts/47.html`

Aufgabe 2 (SQL Injection)

1. Wodurch wir SQL-Injection prinzipiell ermöglicht?
2. Wie sieht ein einfacher Test aus, um zu überprüfen, ob eine Web-Anwendung anfällig für SQL-Injection ist?
3. Wie kann die Authentifizierung umgangen werden, wenn folgender SQL-Code von einer Web-Anwendung ausgeführt wird, wobei `$username` und `$password$` Benutzereingaben sind.

```
SELECT id FROM logins
  WHERE username = '$username' AND password = '$password';
```

Aufgabe 3 (Race-Conditions)

Betrachten Sie das folgende Java-Servlet, das einen Zähler implementiert:

```
import java.io.*;
import java.servlet.*;
import java.servlet.http.*;

public class Counter extends HttpServlet {
  int count = 0;

  public void doGet(HttpServletRequest in, HttpServletResponse out)
    throws ServletException, IOException {
    out.setContentType("text/plain");
    Printwriter p = out.getWriter();
    count++;
    p.println(count + " hits so far!");
  }
}
```

1. Durch welche Eigenschaft von Java-Servlets wird hier ein Problem verursacht?
2. Finden Sie ein Szenario, das zu einem nicht vom Programmierer beabsichtigten Ergebnis führt.
3. Wie kann das Servlet verbessert werden, sodass es sich wie gewünscht verhält?

Aufgabe 4 (Format-Strings)

1. Was wird unter einem Format-String verstanden?
2. Welcher C-Code bewirkt die folgende Ausgabe?: `Variable x hat den Wert: 7` (d.h. der Wert von x wird als Dezimalzahl ausgegeben).
3. Gegeben sei folgendes Quell-Code:

   ```
   int main(int argc, char argv[]) {
     if (argc > 1)
       printf(argv[1]);
     return 0;
   }
   ```

   Wie können die obersten drei Einträge des Stacks ausgegeben werden (in hexadezimalem Format)?
4. Geben Sie eine Anweisung an, mit der auf den Stack geschrieben werden kann?

Aufgabe 5 (Nicht-technische Aspekte der Erforschung künftiger Bedrohungen)

Machen Sie sich mit dem Papier "Future Threats" von John Aycock und Alana Maurushat vertraut: `http://pages.cpsc.ucalgary.ca/~aycock/papers/vb2007future.pdf`

1. Können Sie sich Umstände vorstellen, unter denen Ideen bzw. Gedanken (im Zusammenhang mit künftigen Bedrohungen der IT-Sicherheit) zu gefährlich sind, um Sie öffentlich auszudrücken? Nennen Sie Beispiele.

Aufgabe 6 (Hausaufgabe zu SQL-Injection)

Eine Web-Anwendung, zu deren Nutzung eine Authentifizierung mit Benutzername und Passwort erforderlich ist, erlaubt den Nutzern im Fall eines vergessenen Passwortes, ein neues Passwort an die registrierte Email-Adresse verschicken zu lassen. Dazu ist die Eingabe des Benutzernames (`username`) erforderlich, der als Eingabe für folgende SQL-Anfrage dient:

```
SELECT email FROM users WHERE name = '$username';
```

Sollte der angegebene Benutzername in der Datenbank gefunden werden, wird dem Benutzer eine Ausgabe der folgenden Form angezeigt:

```
Ihr neues Passwort wurde Ihnen an die Emailadresse <email-adresse> geschickt.
```

1. Wie kann ein Angreifer die Email-Adressen der in der Datenbank gespeicherten Nutzer auslesen?
2. Welche Gegenmaßnahmen gegen SQL-Injection existieren?

Aufgabe 7 (Hausaufgabe: Race-Condition)

Betrachten Sie folgenden Quellcode in C/C++:

```
#include "sys/types.h"
#include "sys/stat.h"
#include "unistd.h"
#include "fcntl.h"
const char *filename = "/tmp/splat";
if (access(filename, R_OK) == 0) {
  int fd=open(filename, O_RDONLY);
  handle_file_contents(fd);
  close(fd);
} else {
  //handle error
}
```

1. Welche Art Angriff ist hier möglich?
2. Wie sieht ein entsprechendes Angriffs-Szenario aus?
3. Verändern Sie den Code, so dass er nicht mehr angreifbar ist?

Aufgabe 8 (Hausaufgabe: Format-Strings)

1. Ist folgendes Code-Stück angreifbar?

```
...
char tmpbuf[512];

snprintf (tmpbuf, sizeof (tmpbuf), "foo: %s", user);
tmpbuf[sizeof (tmpbuf) - 1] = '\0';
syslog (LOG_NOTICE, tmpbuf);
...
```

   Hinweise:

   - Der Inhalt der Variable `user` ist eine Benutzereingabe.
   - `void syslog(int priority, const char *message, ...)` writes `message` to the system message logger. ... The message is identical to a printf format string, ...

2. Wie lassen sich Format String-Attacken verhindern?

Aufgabe 9 (Zusatzaufgabe: Satan Virus)

1. Nennen Sie ihnen bekannte Infektionsmechanismen/Propagationsstrategien von Malware.
2. Machen Sie sich mit dem Konzept des *Satan Virus* vertraut (`http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-666.pdf`).
3. Verwenden Sie Programme wie Kazza, eMule oder Azureus?