Automatisierte Erkennung von Sicherheitslücken in Stud.IP

Michael Hoffmann

Universität Trier

Warum Automatisierung?

- Softwareprojekte werden immer größer und komplexer
- Sicherheitslücken können oft leicht übersehen werden (Limitierung durch Erfahrung)
- Zeitersparnis durch maschinelle Unterstützung

Mögliche Lösung: **Statische Quellcodeanalyse**

Vorteile:

- Unkomplizierte Anwendung

Nachteile:

- Verwendete Tools müssen die neusten Standards der Programmiersprache beherrschen (z. B. OOP in PHP)
- Das Problem Schwachstellen in Programmcode zu finden ist **unentscheidbar** (Beweis durch Beweisidee des Satzes von Rice)

Open-Source-Tools für die statische Quellcodeanalyse:

- Pixy
  - Geschrieben in Perl/Java
  - Analysiert PHP 4 Code
- RIPS
  - Entwickler: Johannes Dahse (Universität Bochum)
  - Vortrag beim *Bundesamt für Sicherheit in der Informationstechnik*
  - Geschrieben in PHP
  - Analysiert PHP 5 Code (mit Einschränkungen)
  - Seminararbeit (Sommersemester 2010): http://www.nds.rub.de/teaching/BestStudentPaperAward/

Erfahrungen mit Pixy:

Nachteile

- benutzerunfreundlich (keine GUI)
- keine PHP-5-Unterstützung (viele *Parse-Error*)
- beim Scan von Stud.IP gelang kaum ein „Include“
- viele falsche XSS-Warnungen

Vorteile

- mögliche Scaneinstellung „register_globals=On“
- sehr schneller Scanvorgang
- sehr einfache Installation (unter Ubuntu)

main.php:

```
<?php
    include “origin.php“;
    echo $var;
?>
```

origin.php:

```
<?php
    $var = “Hello World!“;
?>
```

Erfahrungen mit RIPS:

Nachteile

- fehlende Scaneinstellung „register_globals =On“
- keine Unterstützung von OOP in PHP (Dunkelziffer bei fehlgeschlagenen Includes?, Flexi-Templates?)

Vorteile

- sehr benutzerfreundlich (GUI: Übersichtlich, Anzeige von Code)
- einfache Installation
- bessere Include-Rate (~87%) (neuste Version ~98%)

Erzeugung eines Flexi-Templates aus der Datei „QuickSearch.class.php“:

```
$template = $GLOBALS['template_factory']
    ->open('quicksearch/inputfield.php');
```

Funktionsweise von RIPS basiert auf der *Taint analysis:*

- Variableninhalte, die die Parameter potentiell gefährlicher Methodenaufrufe (PVF) bilden, werden zu ihren „Ursprüngen“ zurückverfolgt
- Die Ausgabe wird nach verschiedenen *Verbosity Level* festgelegt

Übersicht über **Verbosity Level**:

- Verbosity Level **1**: Userinput
- Verbosity Level **2**: Userinput + Dateien und Datenbank
- Verbosity Level **3**: Alle aus Level 2 + durch Sicherheitsfunktionen gesicherte PVF-Aufrufe
- Verbosity Level **4**: Es werden zusätzliche Informationen über den Scan erfasst (z.B. Include-Erfolge)
- Verbosity Level **5**: Alle PVF-Aufrufe werden ausgegeben unabhängig von dem Ursprung der Parameter

Funktionsweise der *Taint analysis* an einem Beispiel:

```
<?php
   $a = $_GET[‘a‘];   //Userinput
   $b = $a;
   echo $b;           //PVF für XSS
?>
```

Installation von RIPS:

- Benötigt einen PHP-fähigen Webserver (z.B. Apache)
- Wird über eine Webschnittstelle in einem Browser gesteuert und bedient
- **Hinweis:** Änderung in der **php.ini** des Webservers:
  - **max_input_time = -1**
  - (**max_execution_time = 0**
  - bzw. **set_time_limit(0);** in **general.php** von RIPS)

RIPS in der Praxis (Live-Vorführung)

Aktuelle Probleme von RIPS mit Stud.IP:

- Einstellung „register_globals=On“ (steht in PHP 5.4 nicht mehr zur Verfügung)
- OOP in PHP (wird voraussichtlich innerhalb dieses Jahres vom Entwickler hinzugefügt)
- fehlschlagende Includes von Dateien (deutliche Verbesserung in der aktuellen Version 0.53)

Ausblick:

- Johannes Dahse vermeldet Fortschritte in der Weiterentwicklung von RIPS

- Praktische Einsatzmöglichkeit für RIPS:
  1. Grobe Prüfung der Sicherheit mit RIPS
  2. Detaillierte Prüfung durch Entwickler

Johannes Dahse, RIPS – free PHP security scanner using static code analysis

http://rips-scanner.sourceforge.net/

Vielen Dank

für die Einladung und Ihre Aufmerksamkeit!