Kurzfassung der Dissertation „Static Detection of Complex Vulnerabilities in Modern PHP Applications“ von Johannes Dahse

Moderne Webseiten haben sich zu interaktiven Applikationen entwickelt, die täglich vertrauliche Benutzerdaten (z. B. Kreditkartendaten und Passwörter) verarbeiten. Diese sensiblen Daten erfordern verlässlichen Schutz vor Angreifern, die Sicherheitsschwachstellen im Programmcode der Applikation ausnutzen. Vor allem Webapplikationen, die in der populärsten serverseitigen Skriptsprache PHP entwickelt wurden, sind anfällig für Schwachstellen. Trotz einer erhöhten Sensibilisierung der Entwickler für traditionelle Schwachstellentypen, wie z. B. *Cross-Site Scripting* und *SQL Injection*, treten diese weiterhin durch fehleranfällige Sicherheitsmechanismen oder missverständliche Spracheigenschaften auf. Zudem sind komplexere Schwachstellentypen, wie z. B. *Second-Order* oder *PHP Object Injection* Schwachstellen, vergleichsweise unbekannt und werden aktiv ausgenutzt.

Eine manuelle Suche nach komplexen Schwachstellen in modernen Applikationen mit mehreren hunderttausend Zeilen Code ist teuer, zeitaufwändig und erfordert Spezialwissen. Mit Hilfe von statischer Codeanalyse können Schwachstellen automatisiert erkannt werden, um sie anschließend zu beseitigen. Bisherige Ansätze in diesem Bereich konzentrieren sich jedoch nur auf die Erkennung von einigen traditionellen Schwachstellentypen und verfehlen kompliziertere Ausprägungen oder Typen von Schwachstellen. Außerdem sind die Ansätze nicht für größere Anwendungen skalierbar und wichtige Spracheigenschaften werden nicht unterstützt.

In dieser Dissertation werden neuartige Methoden für die effiziente und präzise Analyse von PHP-Code präsentiert, die es ermöglichen, sowohl traditionelle als auch komplexe Sicherheitsschwachstellen automatisiert zu erkennen. Eine umfassende Konfiguration und Simulation von über 1.200 PHP-Eigenschaften erlaubt es, die hochdynamische Sprache PHP präzise zu modellieren. Durch die Erstellung von Block- und Funktionssummarien, kann eine effiziente *Taint*-Analyse für 36 verschiedene Schwachstellentypen durchgeführt werden. Dabei wird mit Hilfe einer *String*-Analyse erstmalig das Zusammenspiel von Sicherheitsmechanismen, Kodierungen, Eingaben, Operationen, Markup-Kontexten und PHP-Einstellungen berücksichtigt. Weiterhin werden bisher unauffindbare *Second-Order* Schwachstellen und verwandte *Multi-Step Exploits* detektiert. Durch eine neuartige vorwärts gerichtete Objektanalyse können auch erstmalig mögliche Angriffsvektoren für *PHP Object Injection* Schwachstellen automatisch generiert werden.

Die neuen Analysetechniken wurden in einem Prototypen implementiert. Eine Evaluierung zeigt, dass dieser in der Lage ist, kritische und komplexe Schwachstellen in modernen Applikationen aufzuspüren, die von bisherigen Ansätzen nicht erkannt werden: insgesamt wurden 321 bisher unbekannte Schwachstellen in 23 weitverbreiteten PHP-Applikationen detektiert, u. a. in *Joomla*, *phpBB* und *osCommerce*. Abschließend wurden mit Hilfe des Prototypen gängige Vorgehensweisen von Entwicklern und Angreifern studiert. Zum einen wurde analysiert, welche Sicherheitsmechanismen von Entwicklern für welche Markup-Kontexte in der Praxis eingesetzt werden, und welche Fallstricke existieren. Zum anderen wurde untersucht, welche Funktionalitäten und Hintertüren Angreifer in populären PHP-Shells nutzen.