Bachelor-/Masterarbeit

Methoden zur Ermittlung der Code-Abdeckung bei Fuzzing-Tests für Embedded Systems

Ziel

Fuzzing ist eine Technik zur Entdeckung von Schwachstellen in Softwareanwendungen und Embedded Systems. Während eines Fuzzing-Laufs werden gezielt fehlerhafte Daten an eine Anwendung übergeben, um zu prüfen, ob die Implementierung diese Daten verarbeiten kann oder ob es zu Abstürzen oder unerwarteten Fehlerzuständen kommt.

Um ein umfassendes Bild davon zu erhalten, wie gut eine Anwendung durch das Fuzzing abgedeckt wurde, werden Daten zur sogenannten Code Coverage erhoben. Diese Daten zeigen, welche Teile des Codes durch den Fuzzer erreicht wurden, wobei das Ziel eine möglichst hohe Abdeckung ist.

Viele Fuzzing-Tools nutzen dazu eine Instrumentierung des Source Codes. Durch das gezielte Einfügen von Messcode während des Kompilierens können Informationen zur Code Coverage gewonnen werden. Allerdings ist dies nur möglich, wenn der Quellcode verfügbar ist und mit einem dafür geeigneten Compiler kompiliert werden kann.

Im Bereich der Embedded Systems sind solche Instrumentierungen jedoch oft schwierig oder gar nicht umsetzbar. Insbesondere bei der Untersuchung von Netzwerkdiensten auf diesen Systemen findet üblicherweise keine Code-Coverage-Auswertung statt.

Embedded Systems verfügen jedoch in der Regel über eine Debugging-Schnittstelle, die genutzt werden kann, um Breakpoints zu setzen. Wenn ein Breakpoint erreicht wird, lässt sich feststellen, dass die entsprechende Code-Zeile ausgeführt wurde. Durch geschicktes Setzen von Breakpoints kann daher ermittelt werden, welche Teile des Firmware-Codes ausgeführt wurden. Mit GDBFuzz existiert bereits ein Werkzeug, das diesen Ansatz in einen Fuzzer integriert.

Ziel dieser Arbeit ist die Entwicklung eines eigenständigen Werkzeugs, das einen ähnlichen Ansatz verfolgt und die Erhebung von Code-Coverage-Informationen für Embedded Systems im Betrieb ermöglicht. Das Werkzeug soll unabhängig von einem bestimmten Fuzzer arbeiten und einige der Einschränkungen von GDBFuzz überwinden. Im Anschluss wird das entwickelte Werkzeug anhand verschiedener kleiner Anwendungen auf Embedded Systems evaluiert.

Anforderungen und Voraussetzungen

- Einarbeitung in das Thema Debugging auf Embedded Systems und Reverse Engineering Werkzeugen
- Erstellung von Test-Anwendungen für ESP32-Boards
- Entwicklung eines Tools zur Instrumentierung eines Embedded Systems mittels Breakpoints
- Auswertung anhand verschiedener Beispielanwendungen
- Voraussetzungen: Kenntnisse der Sprache Python und Bereitschaft sich in das Thema einzuarbeiten

Ansprechpartner

Prof. Dr. Lothar Braun | ✉ lothar.braun@tha.de | ✆ +49 8215586-3378

THA_innos

Das Institut für innovative Sicherheit (THA_innos) bietet eine Vielzahl von Abschluss- und Projektarbeiten im Themenfeld der Cyber Security an. Unser Team unterstützt Studierende dabei mit Know-How und Praxiserfahrung und ist zudem offen für eigene Themenvorschläge. Durch die enge Zusammenarbeit mit der Forschungsgruppe vor Ort im MRM-Gebäude lernen Studierende sowohl das Institutsleben, als auch die aktuelle Forschung von THA_innos kennen.

Weitere Informationen auch unter `https://innos.tha.de`