Open C³S
Open Competence Center for Cyber Security

FAU
Friedrich-Alexander-Universität
Technische Fakultät

Modulbeschreibung:

Systemnahe Programmierung / Reverse Engineering

Modulbezeichnung:	Systemnahe Programmierung / Reverse Engineering
Zertifikatsabschluss:	Hochschulzertifikat
Verwendbarkeit:	Bachelorstudiengang Informatik/IT-Sicherheit (Pflichtmodul)
Modulverantwortliche(r):	Dr. Werner Massonne
Dozent(in):	Dr. Werner Massonne
Zeitraum:	Nächster Angebotszeitraum: Sommersemester 2025 Dauer ca. 6 Monate
Leistungspunkte:	5 ECTS-Punkte
Zielgruppe:	Forensische Ermittler und Sicherheitsanalysten, Berufspraktiker/-innen mit und ohne Abitur, die sich in den spezifischen Fachbereichen auf akademischem Niveau passgenau im Bereich Cyber-Sicherheit weiterbilden möchten.
Studien- und Prüfungsleistungen:	Hausarbeit: Reverse Engineering eines Malware Binary, Verfassen eines Projektberichts
Notwendige Voraussetzungen:	Programmierkenntnisse in der Sprache C, Kenntnisse über digitale Zahlendarstellungen und Kodierungen (z.B. ASCII, Zweierkomplementdarstellung), Grundkenntnisse in Betriebssystemen und Rechnerarchitektur
Empfohlene Voraussetzungen:	
Sprache:	Deutsch
Arbeitsaufwand bzw. Gesamtworkload:	Wie viel Arbeitszeit (Workload) ist für das Modul insgesamt vorgesehen?

Präsenzstudium	15	Zeitstunden
Fernstudienanteil:	135	Zeitstunden
davon Selbststudium:	70	Zeitstunden
davon Aufgaben und Hausarbeit:	55	Zeitstunden
davon Online-Betreuung:	10	Zeitstunden
Summe:	**150**	**Zeitstunden**

30 h = 1 Leistungspunkt nach ECTS

Lerninhalte	In diesem Modul werden die folgenden Themengebiete behandelt: • Rechnerstrukturen und Betriebssysteme ◦ Von-Neumann-Architektur ◦ Allgemeine Prinzipien der Assemblerprogrammierung ◦ Innere Strukturen des Betriebssystems Microsoft Windows • Intel x86-IA-32-Architektur und IA-32-Assembler ◦ Architekturmerkmale ◦ Registersatz ◦ Befehlssatz ◦ Adressierung ◦ Stack und Unterprogramm-Aufrufkonventionen ◦ Speicherverwaltung ◦ Interrupts und Exceptions • Programmanalyse ◦ Codeerzeugung durch Compiler und Dekompilierung ◦ Optimierungsverfahren, die eine Dekompilierung erschweren • Softwaresicherheit ◦ Buffer Overflows ◦ Gegenmaßnahmen zur Vermeidung von Buffer Overflows ◦ Gegen-Gegenmaßnahmen (z.B. Return Oriented Programming) • Malwaretechniken und Malwareanalyse ◦ Statische und dynamische Analyse von Binaries mittels IDA ◦ Obfuscation ◦ Verfahren zur Verhinderung der Disassemblierung ◦ Malwaretechniken, Packer, Anti-Reverse-Engineering-Methoden ◦ Analyse realer Malware in einer virtuellen Analyseumgebung **Hausarbeit:** • Im Rahmen der Hausarbeit soll ein individualisiertes Malware Binary vollständig analysiert werden. Dabei kommen die im Modul gelehrten Techniken und Analysemethoden zur Anwendung. Die durchgeführte Analyse soll in einem möglichst vollständigen Bericht zusammengefasst werden.
Angestrebte Lernergebnisse:	*Fachkompetenz:* Die Studierenden kennen die Einsatzszenarien der systemnahen Programmierung, und ihre Prinzipien und Methoden sind ihnen bekannt. Sie können die Grundprinzipien aktueller Rechnerarchitekturen und Betriebssysteme benennen und einordnen. Die für eine Programmanalyse wesentlichen Strukturen von Microsoft Windows sind ihnen bekannt. Die Studierenden haben fundierte Kenntnisse in der Programmierung von IA-32 auf Maschinenebene. Sie können Maschinencode aus der Hochsprache C erzeugen und können die Methoden zur Dekompilierung von Maschinenprogrammen benennen und anwenden. Sie haben einen Überblick über Verfahren zur Codeoptimierung und Codeverschleierung (Obfuscation). Den Studierenden die Schwächen - bzgl. Softwaresicherheit - der Programmiersprache C bekannt. Einige der bedeutendsten Sicherheitsprobleme/Sicherheitslücken, die insbesondere durch die Verwendung von C auf heutigen Rechnerarchitekturen entstehen können, können Sie erklären. Des Weiteren können Sie übliche Gegenmaßnahmen beschreiben, die die Ausnutzung von Sicherheitslücken unterbinden sollen sowie weiterführende Maßnahmen, die diese Gegenmaßnahmen ihrerseits auszuhebeln versuchen.

	Das Analyseprogramm IDA können die Absolventen einsetzen, Vorteile und Nachteile einer statischen und dynamischen Programmanalyse sind ihnen bekannt, und sie können diese bedarfsabhängig anwenden. Die Absolventen haben einen Einblick in die Funktionsweise von Malware auf Systemebene gewonnen und können „einfache“ Malware für Windows-Systeme selbstständig analysieren. *Methodenkompetenz:* Die Studierenden haben die Fähigkeit, systemnahe Programme zu erstellen und zu verstehen. Die Studierenden können Probleme auf dieser Ebene der Programmierung erkennen und Schwachstellen identifizieren und analysieren. *Sozialkompetenz:* Aufgrund der Teamarbeit, unter anderem am Präsenzwochenende, erweitern die Studierenden ihre Teamfähigkeit und Sozialkompetenz. *Selbstkompetenz:* Durch das eigenverantwortliche Entwickeln von Programmen und die Programmanalyse erweitern die Studierenden ihr selbstständiges Handeln. Durch das Verfassen eines Berichts wird die Selbstsicherheit der Studierenden gestärkt.
Lehrveranstaltungen und Lehrformen:	Präsenzveranstaltung: Vorlesung, Übungen: Analyse verschleierter Binaries, Analyse von Malware, Vorbereitung auf die Hausarbeit Onlineveranstaltung: flexible Vertiefung wichtiger Themen, Lernen im Dialog, Übungen
Medienformen:	Studienbriefe in schriftlicher und elektronischer Form, Onlinematerial in Lernplattform, Übungen und Projekt über Lernplattform, Online-Konferenzen, Chat und Forum, Präsenzveranstaltung mit Rechner und Beamer
Literatur:	Als weiterführende Literatur wird empfohlen: • Intel 80386, Programmers Reference Manual, 1987 • Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, Sikorski and Honig, 2012 Weitere Literatur wird in der Lehrveranstaltung bekannt gegeben.