**Kurzfassung der Dissertation „On the Impact of Memory Corruption Vulnerabilities in Client Applications“ von Robert Gawlik**

In unserem digitalen Zeitalter sind clientseitige Anwendungen allgegenwärtig. Insbesondere Webbrowser werden von sehr vielen Benutzern für etliche Tätigkeiten verwendet. Darunter fällt die Informationsgewinnung, die Aktivität in sozialen Medien oder die Kommunikation der Benutzer untereinander. Da die Popularität von Webbrowsern gewachsen ist, haben auch Angreifer ihre Aufmerksamkeit auf diese Clientanwendungen gerichtet, um in Computersysteme einzubrechen. Wegen ihrer großen Codebasis und enormen Komplexität existieren viele ausnutzbare Speicherfehler in diesen Programmen.

In dieser Dissertation werden verschiedene Auswirkungen von Speicherfehlern in Clientanwendungen aus offensiven und defensiven Gesichtspunkten untersucht. Der Ausnutzungsprozess von Schwachstellen (engl. *Exploit*) kann in verschiedene, aufeinanderfolgende Schritte unterteilt werden. Der Angreifer benötigt meist Informationen über den Adressraum des verwundbaren Programms. Dieser wichtige Schritt wird auch *Informationsleck* (engl. *Information Leak*) oder *Speicherenthüllung* (engl. *Memory Disclosure*) genannt. Sobald der Angreifer genügend Wissen über den Adressraum des Programms gesammelt hat, ist er in der Lage, den Kontrollfluss des Programms zu übernehmen. Dieser Schritt wird auch als *Control-Flow Hijacking* bezeichnet. In dieser Dissertation werden *Information Leaks* und *Control-Flow Hijacking* aus der Perspektive von Angreifern als auch Verteidigern betrachtet.

Wir kombinieren die Technik von Information Leaks mit einem Verhalten in Browsern, welches bisher in diesem Ausmaß unbekannt war (*Absturzresistenz*, engl. *Crash Resistance*). Dabei wird das Programm am Laufen gehalten, obwohl es aufgrund kritischer Speicherfehler, wie beispielsweise eines illegalen Lesezugriffs, terminieren sollte. Dieser Ansatz erlaubt es uns, aus der Angreiferperspektive Verteidigungsansätze zu beurteilen, die versprechen, den Adressraum vor Angreifern geheimzuhalten. Aus der Sicht eines Verteidigers werden Information Leaks ebenfalls betrachtet. Um diesen Schritt eines Angriffs zu erkennen wird ein Konzept für sog. *Scripting*-Umgebungen, wie *JavaScript*, vorgestellt. Dabei werden zwei Prozesse des gleichen Programms in ihrer Ausführung synchronisiert. Da wir in beiden einen unterschiedlichen Adressraum erzwingen, manifestiert sich ein Information Leak unterschiedlich in beiden Prozessen und kann detektiert werden.

Auch widmet sich diese Dissertation dem Schritt der Kontrollflussübernahme. Die Wiederverwendung von Code ist momentan die gängigste Methode beliebige Berechnungen durchzuführen, sobald der Programmfluss kontrolliert wird. Aus der Sicht eines Angreifers ist es wichtig zu wissen, wie viel Code wiederverwendbar ist. Daher wird ein System vorgestellt, welches helfen soll, spezielle Defensivmaßnahmen – sog. *Control-Flow Integrity*-Lösungen (CFI) – zu beurteilen. Dabei wird architekturunabhängig versucht, die Menge an wiederverwendbarem Code, der CFI-Regeln entspricht, zu maximieren.

Angreifer können in Browsern den Kontrollfluss über sog. *Vtable Hijacking* übernehmen. Diese Dissertation betrachtet diese weitverbreitete, offensive Technik aus der Perspektive eines Verteidigers. Spezielle Funktionstabellen (sog. *Vtables*), die von Angreifern im Adressraum abgelegt werden, unterscheiden sich von echten Vtables. Unter der Benutzung verschiedener Heuristiken und Techniken zeigen wir als Erste, dass eine Abschwächung von Vtable Hijacking-Angriffen für binäre Programme möglich ist.