Exploit Detektion mittels Memory Tracing

Fachgebiet: Informatik (IT Security)
Betreuer: Dr. Endre Bangerter, Dominic Fischer
Experte: Dr. Federico Flueckiger (Eidgenössisches Finanzdepartement EFD)

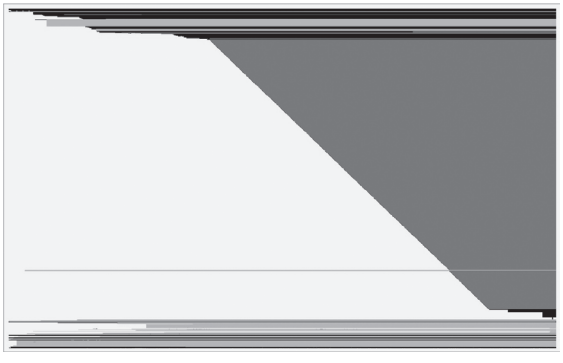
Computersysteme dringen immer tiefer in unseren Alltag vor. Mobiltelefone, intelligente Stromzähler, automatische Insulinpumpen, Kraftwerke, Autopiloten, usw. Alle diese Systeme basieren auf Software und Software enthält immer Fehler. Im Gegensatz zu früher, als sich nur Computerfreaks für Fehler und Sicherheitslücken interessierten, sind diese heute im Fokus einer ganzen Industrie und im Interesse von kriminellen, militärischen oder nachrichtendienstlichen Abnehmern.

Relevanz

Der bekannteste Schutz gegen Malware sind Antivirenprodukte. Dabei erfolgt die Detektion der Schadsoftware über eine Signatur. Heute werden täglich mehrere 100'000 neue Signaturen generiert. Durch die stetig steigende Menge und Komplexität von Malware, benötigt dieser Ansatz immer mehr Ressourcen. Wenn eine Schadsoftware vor der Ausführung der boshaften Bestandteile erkannt wird, spricht man von einem präventiven Schutz, andernfalls ist von einer postum Detektion die Rede. Der Vorteil von postum Mechanismen liegt in der möglichen Verwendung von dynamischer Analyse für Software. Die dynamische Analyse ist in der Lage, bestimmte Verhaltensmuster einer Software während deren Laufzeit zu erkennen. Durch die Reduktion auf die prinzipiellen Erkennungsmerkmale einer Schadsoftware entstehen generisch verwendbare Detektionen.
Das Memory repräsentiert das unverfälschte Abbild einer Software und dessen Verhalten. Beim Memory Tracing werden die Memoryzustände zu bestimmten Zeitpunkten aufgezeichnet, sodass sie postum zur Analyse und Detektion von Malware genutzt werden können.

Resultate

Recherchen zu Beginn dieser Arbeit ergaben, dass sich bestimmte Exploittechniken durch sehr hohe Erfolgschancen auszeichnen. Dabei konzentrierte sich die Arbeit auf zwei stark vertretene Techniken. Diese wurden im Detail analysiert, die Charakteristiken ihrer Repräsentation im Memory festgehalten und daraus versuchsmässige Detektoren realisiert.



Memory Tracing über die Laufzeit eines Prozesses. Ein detektierter Heap Spray ist in grau sichtbar.

Heap Spray

Klassische Exploits missbrauchten den Determinismus von Betriebssystemen, um ihren Schadcode an einer vorhersehbaren Stelle abzulegen und diesen aufzurufen. Die Einführung von Randomisierung in modernen Betriebssystemen verhindert diese Praktik. Um dem entgegenzuwirken, wurde der Heap Spray entwickelt. Skriptsprachen wie JavaScript erlauben eine dynamische und kontrollierte Allokation von beliebigen Daten. Indem ein grosser Teil des Memorys mit gewünschten Daten gefüllt wird, erhöht sich die Wahrscheinlichkeit den Schadcode an einer vorhersehbaren Stelle zu finden. Obwohl der Heap Spray von sich aus keinen Schaden anrichtet, wird er oft in Kombination mit anderen Exploits verwendet, um deren Erfolgschancen zu erhöhen.

Return Oriented Programming

ROP entstand als Reaktion auf die Einführung neuer Schutzmechanismen, welche die Ausführung von eingeschleustem klassischem Schadcode verunmöglichen. ROP verwendet als Lösung für diese Problematik keinen systemfremden Code mehr, sondern greift auf Code-Fragmente bestehender Applikationen zu. Die einzelnen Fragmente bilden in sinnvoller Anordnung eine neue Instruktionsfolge.
Attacken können entweder vollständig in ROP realisiert oder dazu verwendet werden, Schutzmechanismen zu deaktivieren und dadurch die Ausführung von klassischem Schadcode wieder zu ermöglichen.

Fazit

Für beide Techniken konnten generische Heuristiken zu deren Detektion entwickelt werden. In den durchgeführten Experimenten waren die Detektoren in der Lage, alle Exploittechniken zu erkennen. Diese Arbeit bestätigte die Hypothese, dass sich Memory Tracing dazu eignet, Exploits im Memory zu analysieren und zu detektieren.

Ramon Spahr

Jonas Wagner