

WEF - Web Exploit Finder

Automatische Drive-By-Download Erkennung in einer virtualisierten Umgebung

Thomas Müller Benjamin Mack Mehmet Arziman
Hochschule der Medien (HdM), Stuttgart
{tm026,bm022,ma018}@hdm-stuttgart.de

Abstrakt

In regelmäßigen Abständen kann man über neue Sicherheitslücken im Microsofts Internet Explorer oder in Mozillas Firefox lesen. Einige dieser Lücken eignen sich um den Browsern schadhafte Quellcode unterzuschieben. Diese Angriffe werden auch als Remote-Code- Execution-Attack bezeichnet und basieren in der Regel auf der gezielten Ausnutzung von Buffer-Overflows (Speicherüberlauf) in den Applikationen. Diese Schwachstellen treten nicht nur bei Browsern auf, sondern bei fast allen Diensten und Applikationen, die Teil des Internets sind oder dieses als Kommunikationsplattform nutzen.

Wir richten den Fokus jedoch auf Internet Browser begründet durch zwei entscheidende Problematiken. Erstens stellen sie die Schnittstelle zum WorldWideWeb dar indem die Rendering-Engine Hypertext und andere Webinhalte für den Menschen sichtbar macht. Dies hat zur Folge, dass alle Teile einer Internetseite vom Browser interpretiert und weiterverarbeitet werden müssen, was eine komplexe und fehleranfällige Architektur zur Folge hat, vor allem was mobilen Code (JavaScript, Java, ActiveX, XUL etc.) betrifft. Zweitens ist der Browser wohl das am häufigsten verwendete Programm aus der Familie der potentiell gefährdeten Software. Im Gegensatz zu Server-Software wird der Browser häufig von technisch nicht versierten Benutzern verwendet, denen oft weder die Risiken noch mögliche Gegenmaßnahmen bekannt sind. Und selbst versierte Nutzer setzen sich oft dem Risiko eines Angriffs aus.

Unser Ziel war es deshalb, ein System zu entwickeln, das automatisiert schadhafte Internetseiten aufspürt und identifizieren kann.

Zusätzlich ist es möglich das entwickelte System als Plattform für andere Security/Sandbox-Tests zu verwenden. Ein Szenario hierfür wäre die automatische Analyse von Malware jeder Art in einer gesicherten und leicht wartbaren virtualisierten Umgebung.

1 Einleitung

Zu Beginn des Projektes waren zunächst einige wichtige Fragen und Anforderungen zu klären.

- Wie soll der Begriff „**schadhaft**“ für unser Projekt definiert sein?
- Welche Möglichkeiten gibt es schadhafte Internet Inhalte zu erkennen?
- Wie muss ein entsprechendes Software-System konzipiert sein?

Als „**schadhaft**“ versteht unser Projekt Webseiten, welche Schadsoftware (Viren, Würmer, Trojanische Pferde, Keylogger, ...) auf einem Client-Rechner installieren und ausführen. Wir konzentrieren uns auf Schadsoftware, die sich ohne Benutzerinteraktion installiert und somit sogar für versierte Benutzer nicht ohne weiteres erkennbar ist (Drive-By-Downloads).

Diese Einschränkung wurde bewusst gewählt um nur Seiten zu finden, welche tatsächlich Sicherheitslücken im Browser ausnutzen. Seiten, die den Benutzer überlisten oder Seiten, die modifizierte Software zum Download anbieten, sind nicht Ziel unserer Suche. Es ist aber im Konzept berücksichtigt, diese Funktion zu einem späteren Zeitpunkt nachzurüsten.

Um die Frage beantworten zu können, wie man schadhafte Seiten erkennen kann, muss man sich in die Rolle des Angreifers versetzen. Was sind seine Ziele und wie werden diese erreicht? Ein Angreifer will den Rechner dauerhaft kompromittieren und muss hierfür den Zustand des PCs in irgendeiner Form verändern. Ein typisches Szenario ist:

- Der Angreifer bringt mit Hilfe eines Buffer-Overflows im Browser seinen eigenen Code (Shellcode) zur Ausführung.
- Da die Funktionalität durch die geringe mögliche Größe stark eingeschränkt ist, wird in der Regel nur versucht, weiteren ausführbaren Code aus dem Internet nachzuladen und auf dem Rechner zu starten.

Diese Mini-Applikation wird häufig als „Dropper“ oder „Downloader“ bezeichnet, da sie die eigentliche Malware auf den Rechner lädt und diese mittels Registry-Einträgen bei folgenden Systemstarts zur Ausführung bringt.

Um derartige Veränderungen am Zustand des Rechners zu erkennen gibt es mindestens zwei unterschiedliche Möglichkeiten:

- *Intrusion Detection: Vor und nach dem Besuch einer verdächtigen Internetseite den Zustand des Rechners zu ermitteln und die beiden Ergebnisse zu vergleichen.*
Zur Ermittlung des Zustandes kann ein Auflisten aller relevanten Dateien und Registry-Einträge in Verbindung mit Prüfsummen über diese Daten verwendet werden. Dadurch können neu erzeugte sowie veränderte Dateien erkannt werden. Die entscheidenden Nachteile dabei sind die große zeitliche Verzögerung der Entdeckung sowie die schlechte Performance und Skalierbarkeit.
- *Rootkit: Erkennung mittels Modifikation des Betriebssystems.*
Hierbei werden die relevanten Systemaufrufe (Systemcalls) von Anwendungen überwacht und ausgewertet. Eine derartige Modifikation des Betriebssystems ist aber standardmäßig nicht vorgesehen und benötigt einen tiefen Eingriff in den Kernel. Diese Technik findet häufig auch in so genannten *Rootkits* Verwendung und wird deshalb meist selbst als Rootkit bezeichnet.

Wir haben uns aufgrund der erwähnten Performance-Vorteile für die Rootkit-Lösung entschieden.

Um ein solches System später auch für Forschungszwecke sinnvoll einsetzen zu können, muss es folgenden Anforderungen genügen:

- Es sollte vollständig autonom arbeiten um so gut wie keine Benutzerinteraktion zu benötigen.
- Es sollte sich aus der Ferne überwachen und steuern lassen, zum Beispiel mit Hilfe eines Webinterfaces.
- Es sollte skalierbar und erweiterbar sein.
- Komponenten für die eigene Sicherheit des Systems verhindern, dass es selbst durch schadhafte Seiten infiziert wird.

2 Einschränkungen und Annahmen

Bei der Erkennung von schadhafte Seiten haben wir uns zunächst für folgende Konfiguration des Testsystems entschieden:

- Windows XP Professional ohne Service Packs
- Keine Sicherheitsupdates installiert
- Ausführung unter dem Administrator-Account

- Verwendung des MS Internet Explorers 6
- Scripting und Java sind aktiviert

Diese Konfiguration wurde gewählt um eine möglichst große Angriffsfläche zu bieten. Sie simuliert den *Worst-Case*, der sich aber durchaus in der Konfiguration des Standard-Nutzers widerspiegelt.

3 System Architektur

Um den gestellten Anforderungen gerecht zu werden, enthält unsere System-Architektur folgende Komponenten:

- Zum Schutz der Architektur und um parallel mehrere Seiten zu prüfen eine **Virtualisierungsschicht**: den VMware Server.
- Zur Erkennung der schadhafte Seiten die Betriebssystem-Modifikation: unser **Rootkit**
- Zur Steuerung des Rootkits und des Internet Explorers sowie die Kommunikation mit der Management Console eine Steuerungskomponente: das **Browser Control**
- Zur Steuerung und Konfiguration des ganzen Systems: Die **Management Konsole**

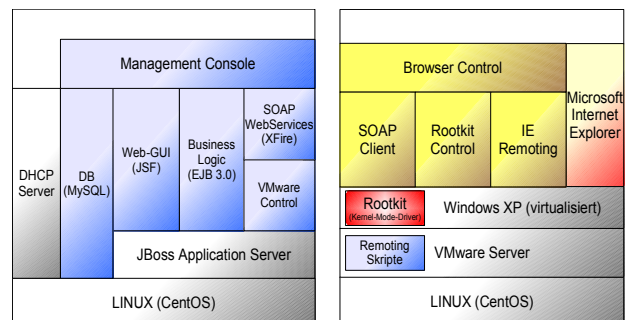


Abbildung 1: Architektur des Systems

4 Eingesetzte Technologien

Auf den beiden eingesetzten PCs läuft als Betriebssystem die Linux-Distribution CentOS. Für die Realisierung der Management Console haben wir uns für J2EE/Java entschieden. Hier kommt der JBoss Application Server als Middleware, EJB 3.0 für die Business Logik und das Datenmodell sowie die Java Server Faces (JSF) für die Web-GUI zum Einsatz.

Das Browser Control bzw. dessen Komponenten sind in C++ als Windows-MFC-Anwendung realisiert. Für die Kommunikation zwischen Management Console und Browser Control wird SOAP verwendet.

Die Virtualisierungsschicht wurde mittels dem kostenlosen VMware Server umgesetzt. Die Skripte zur Steuerung des VMware Server sind eine Kombination

aus C-Anwendungen und Bash-Skripten.

Da wir uns bei der Wahl der Hooking-Technologie (Art der Umleitung von Systemaufrufen) für das SSDT-Hooking entschieden haben muss das Rootkit Zugriff auf Speicherbereiche des Kernels haben. Deshalb ist es als Systemtreiber in C unter Einsatz des MS-DDK (Driver Development Kit) realisiert.

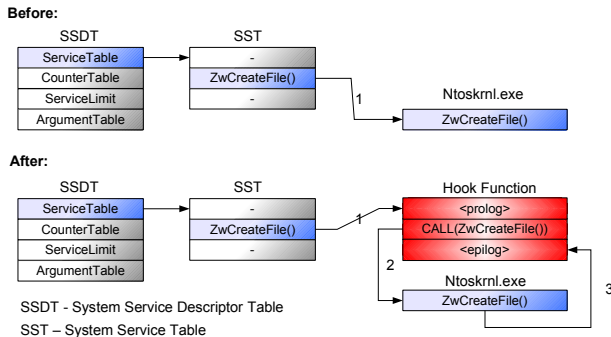


Abbildung 2: Funktionsweise des SSDT-Hooking

5 Funktionsweise

Das System wird über das Webinterface der Management Console (MC) gesteuert und überwacht. Zunächst kann hier eine Liste der zu überprüfenden URLs eingetragen bzw. importiert werden. In einer der kommenden Versionen soll diese Aufgabe zum Teil von einem Crawler übernommen werden, welcher auf der Basis von bereits als schadhaft identifizierten Seiten weitere verlinkte URLs extrahiert.

Um das System selbst nicht zu gefährden und um das System nicht durch ständige Neuinstallation zu verlangsamen, findet der Besuch dieser URLs innerhalb einer Sandbox statt. Diese Sandbox ist als VMware Server Image realisiert und wird mittels von uns entwickelten Bash Skripten und der VMware eigenen API geklont und konfiguriert. Dieser Vorgang kann n-mal wiederholt werden, je nach dem wie viele virtuelle Windows XP Instanzen gleichzeitig URLs überprüfen sollen bzw. je nach Performance des VMware Host Systems. Um den Fortschritt überwachen zu können melden die Skripte den aktuellen Status der Sandbox an die Management Console welche diese auf dem Webinterface anzeigt.

Diese n-Images werden beim VMware Server registriert und gebootet. Ist der Bootvorgang abgeschlossen startet ein weiteres Skript welches einen Snapshot des aktuellen Zustands erstellt und die aktuelle Version des Browser Control sowie das Rootkit in die Sandbox kopiert und startet. Durch den Einsatz der Snapshot-Technologie kann im Falle einer Infektion der Sandbox ein Rollback durchgeführt werden. Im Gegensatz zur Erzeugung einer neuen Sandbox (ca. 1 min) dauert dieser Vorgang nur wenige Sekunden.

Das Browser Control lädt zunächst das Rootkit, in Form eines Windows Treiber, in den Windows XP

Kernel. Danach registriert sich das Browser Control (BC) unter Angabe seiner IP-Adresse bei der Management Console. Ab diesem Zeitpunkt läuft das System in einer Schleife nach folgendem Schema:

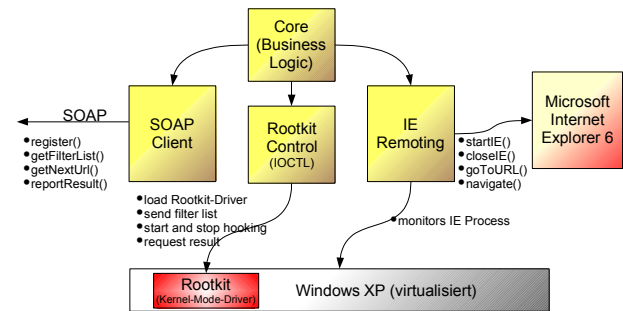


Abbildung 3: Aufbau und Funktion des Browser Control

1. Das BC erfragt mittels SOAP die nächste zu überprüfende URL bei der MC.
2. Das BC sendet dem Rootkit die Nachricht zum Aktivieren der Überwachung. Dadurch können alle relevanten Systemcalls (CreateFile, DeleteFile, Execute, ...) umgeleitet und überwacht werden.
3. Das BC startet den Internet Explorer und übergibt diesem die empfangene URL.
4. Ist die Seite vollständig geladen oder der Timeout für diese Aktion abgelaufen wird der Browser wieder beendet und die Überwachung wird deaktiviert.
5. Nun erfragt das BC beim Rootkit die Ergebnisliste. Im Falle einer sauberen Seite ist diese Liste leer ansonsten enthält sie alle verdächtigen Systemcalls mit Uhrzeit und PID der zugehörigen Anwendung.
6. Ist die Liste leer meldet das BC dies an die MC, wo diese URL in der Datenbank als sauber markiert wird. Enthält die Liste Einträge werden diese ebenfalls an die MC übertragen. Da diese Sandbox nun als infiziert markiert ist ruft die MC ein weiteres Skript auf. Dieses Skript stellt den anfangs erzeugten Snapshot wieder her. Nach dem Kopieren von BC und Rootkit und erneuter Registrierung bei der MC beginnt das System in beiden Fällen wieder mit Schritt eins.

6 Ausblick

Das System wird im Rahmen einer weiteren Semesterarbeit weiterentwickelt und anschließend als Open-Source-Projekt veröffentlicht. Hierbei wird der Fokus auf die Integration eines URL-Crawlers sowie auf die Weiterentwicklung des Rootkit liegen. Das Einsatzgebiet des Systems wird das automatische Suchen von schadhafte Seiten sowie die Ermittlung von Beziehungen zwischen den schadhafte Seiten und deren Betreiber sein.