Binary Security
Übungsblatt 6

Hackerpraktikum

FAU Erlangen-Nürnberg, Lehrstuhl für Informatik 1

Game Labrys (Aufgabe 1)

Willkommen bei VulnGames Industries! Als IT-Sicherheitsbeauftragter sind Sie allein für die Sicherheit der entwickelten Spiele, wie auch des Studios verantwortlich.
Als Ihre erste Aufgabe sollen Sie eine Sicherheitsanalyse unseres neusten Titels "Labrys" durchführen. Laden Sie sich dazu das Spiel "Labrys" aus dem StudOn herunter. Für Hinweise zur Installation und Steuerung des Spiels lesen Sie bitte die `README.txt`. Neue Level können Sie mit einem Text-Editor erstellen, bspw. indem Sie die Datei `./level5/labyrinth` editieren. Alle folgenden Aufgaben können Sie wahlweise mit `labrys.32` oder `labrys.64` lösen.

1. **Cracking**: Zuerst überprüfen Sie den Kopierschutz des Spiels und versuchen es zu cracken.
   a) *Keygen*:
      i. Erstellen Sie einen gültigen Lizenzschlüssel für das Spiel. (0.5 P.)
      ii. Schreiben Sie einen Key-Generator, der mehrere Schlüssel generiert. (0.5 P.)
   b) *Patch*: Modifizeren Sie das Spiel, so dass es auch ohne Lizenz startet. (0.5 P.)
2. **Cheating**: Als nächstes Versuchen Sie sich Vorteile gegenüber Mitspielern durch Cheaten zu verschaffen.
   a) *Wallhack*: Modifizieren Sie das Spiel, so dass Sie horizontal durch Wände laufen können ohne dabei herunterzufallen. (0.5 P.)
   Tipp: Verändern Sie die Methode(n) `collide`.
   b) *Flyhack*: Modifizieren Sie das Spiel, so dass Sie fliegen können. (0.5 P.)
   Tipp: Verändern Sie die Methode `gravity`.
   c) *Speedhack*: Modifizieren Sie das Spiel, so dass Sie schneller werden. (0.5 P.)
   Tipp: Verändern Sie den Multiplikator in `step_forward` oder `save_state`.
3. **Exploitation**: In den letzten Monaten hat sich in der Betatester-Community bereits ein großer Markt für selbsterstellte Levels etabliert. Zeigen Sie nun, dass sich über ausgetauschte Level, das Spiel exploiten lässt.
   a) *Shellcode Injection*: Gestalten Sie Level 5 derart, dass Schadcode auf dem Stack des Spielers ausgeführt wird. Beachten Sie, dass ASLR aktiviert ist. (2.5 P.)
   Tipp: Verwenden Sie zur Demonstration den Shellcode von den Folien oder beliebigen Shellcode aus dem Internet, um `/bin/sh` auszuführen.
   b) *Compiler Security*: Aufgrund Ihres Exploits aus der ersten Teilaufgabe, hat die Entwicklungsabteilung neue Binaries (`labrys_rop.{32,64}`) erstellt, welche

vom Quellcode identisch sind, nun aber mit "allen nur erdenklichen Sicherheitsfeatures" kompiliert wurden. Nun sei eine Ausnutzung der Schwachstelle "vollkommen unmöglich". Welche Compilereinstellung wurde tatsächlich verändert? Nennen Sie (mindestens) eine weitere Compileroption, die genutzt werden könnte, um das Ausnutzen der von Ihnen aufgezeigten Sicherheitslücke zu erschweren und begründen Sie ihre Antwort kurz. (1 P.)

c) *Return Oriented Programming*: Verwenden Sie nun die neuen Binaries (`labrys_rop.{32,64}`) und entwickeln Sie ein Exploit, was ihnen trotz der neuen Sicherheitsfeatures erlaubt eine Shell zu spawnen. (3.5 P.)
Tipp: Tools wie `github.com/JonathanSalwan/ROPgadget` und `github.com/sashs/Ropper` unterstützen Sie bei der Suche nach Gadgets.

10 P.

Malware Analyse (Aufgabe 2)

Am späten Abend ruft Sie ein verzweifelter Kollege aus der Entwicklungsabteilung an. Er habe sich "zur Inspiration" ein gecracktes Spiel heruntergeladen und sich dabei wohl einen Virus eingefangen. "Alles" sei "plötzlich weg", inklusive des Konzepts für das neue Projekt an dem er gerade arbeite und jetzt werde er auch noch erpresst. Um dem neuen Kollegen zu helfen, sichern Sie ihm Ihre Unterstützung zu, woraufhin dieser Ihnen ein VM-Abbild seines Computers schickt.

Warnung: Bei dem zu untersuchenden Binary handelt es sich um echte Schadsoftware! Führen Sie diese daher niemals auf Ihrem eigenen Produktivsystem aus!

Warnung: Bitte überweisen Sie keine Kryptowährung, selbst wenn Sie die Malware dazu auffordern sollte!

Tipp: In dieser Aufgabe geht es lediglich um die zu untersuchenden Binaries. Eine forensische Analyse des Systems ist weder verlangt noch zielführend.

Tipp: Die Teilaufgaben bauen kaum aufeinander auf und können daher größtenteils unabhängig voneinander bearbeitet werden!

1. **Anti-Reversing**: Zuerst werden Sie einen ersten Blick mit einem Disassembler (z.B. Ida, Ghidra, etc.) Ihrer Wahl auf das Programm. Allerdings müssen Sie schnell feststellen, dass sich die Malware mit verschiedenen Techniken vor statischer Analyse schützt.
   a) *String-Obfuscation*: Das Programm verschleiert nahezu alle genutzten Strings. Wie werden die Strings vor dem statischen Auslesen geschützt? (0.5 P.)
   b) *Obfuscation*: Zeigen Sie an zwei weiteren Beispielen, wie das Programm die statische Analyse erschwert und erklären Sie kurz die verwendeten Techniken. Die in den nächsten Aufgaben aufgeführten Techniken (Anti-Sandbox, Anti-Debugging, Nachladen & Packer) sind davon ausgenommen. (1 P.)
2. **Anti-Sandbox**: Sie versuchen die Malware in der gelieferten VM zu starten. Allerdings verhält sich das Programm nicht so, wie von Ihrem Kollegen geschildert.

Inwiefern verhält sich das Programm anders, wenn es eine VM (bzw. Debugging-Versuche) erkennt? Erläutern Sie die beiden implementierten Mechanismen zur VM-Erkennung und entfernen Sie diese, sodass sich das modifizierte Binary in der mitgelieferten VM ausführen lässt. (1.5 P.)

3. **Anti-Debugging**: Nachdem Sie das Programm in einer VM zum Laufen gebracht haben, versuchen Sie die Funktionsweise der Malware mit einem Debugger zu erkunden. Dabei müssen Sie ziemlich schnell feststellen, dass sich das Programm auch gegen Debugging zu schützen versucht.

   a) *Debugger-Erkennung*: Erläutern Sie 5 Techniken, mit denen die Malware Debugging-Versuche erkennt und patchen Sie diese Versuche, sodass Sie das Programm normal mit einem Breakpoint auf main debuggen können. (2.5 P.)

   b) *Anti-Debugging*: Erläutern Sie, wie sich das Programm zusätzlich zur Erkennung gegen Debugging schützt und entfernen Sie diesen Schutz, sodass Sie das Programm effizient debuggen können. Geben Sie ein modifiziertes Programm ab, dass sich effizient mit einem Breakpoint auf main debuggen lässt und dabei maliziöses Verhalten zeigt. (1 P.)

4. **Nachladen & Packer**: Glücklicherweise ist in der Schadsoftware auch eine Entschlüsselungskomponente enthalten. In dieser Aufgabe müssen Sie drei Schichten von Schutz durchbrechen, um diese Funktionalität freizuschalten. In jeder Schicht wird jeweils einfach ein Passwort überprüft. Allerdings nutzen die Schichten verschiedene Möglichkeiten, um sich vor einer statischen Analyse zu schützen.
   Tipp: Sollten Sie Stufe 1 nicht lösen können, können Sie dennoch die nächste Stufen lösen. Nutzen Sie hierzu die im StudOn verfügbare Datei *Decryptor*.

   a) *Stufe 1*: Das Programm lädt zur Laufzeit dynamisch Code nach. Erläutern Sie, wo sich der nachgeladene Code auf dem System befindet und geben Sie das darin verwendete Passwort im Klartext an. (0.5 P.)

   b) *Mini Exploitation*: Stufe 1 enthält eine potenzielle Schwachstelle. Worum handelt es sich, bzw. wofür können derartige Sicherheitslücken beispielsweise ausgenutzt werden? (1 P.)

   c) *Stufe 2*: Aus Stufe 1 heraus wird das Programm Decryptor aufgerufen. Wie ist dieses Programm geschützt? Entfernen Sie den Schutz und geben Sie das verwendete Passwort im Klartext an. (0.5 P.)
      Tipp: Nutzen Sie den Befehl *strings*.

   d) *Stufe 3*: Wo verbirgt sich die 3. Stufe und wie ist Sie geschützt? (0.5 P.)

   e) *Exploitation*: Welches Sicherheitsfeature des Compilers musste für die Implementierung von Stufe 3 deaktiviert werden? Nutzen Sie diesen Umstand und entwickeln Sie ein Exploit, das Ihnen eine Shell öffnet. (1.5 P.)

5. **Analyse**: Nachdem Sie nun alle Schutztechniken der Malware beseitigt haben, können Sie nun Ihre Analyse durchführen.

a) *Verhalten*: Beschreiben Sie kurz das Verhalten und den Aufbau (3 Komponenten/Dateien) der Malware. Welche Dateien sind von der Verschlüsselung durch die Malware betroffen? (1 P.)

b) *Verschlüsselung*: Erläutern Sie den Verschlüsselungsmechanismus, beantworten Sie dabei folgende Fragen: Welche beiden Technologien werden in welcher Art und Weise für die Verschlüsselung der Dateien verwendet? Welche 3 Elemente enthält eine von der Malware verschlüsselte Datei? Ist es möglich ohne die in der Malware enthaltene Entschlüsselungskomponente (Decryptor) die Dateien wieder zu entschlüsseln? Begründen Sie Ihre Antwort kurz. (1.5 P.)

c) *Datenabfluss*: Wird Kontakt durch die Malware mit dem Internet aufgenommen? Wenn ja mit welchem Server/welcher URL? Sind Daten abgeflossen? Wenn ja, welche? (1 P.)

d) *Ein böses Spiel*: Das Programm verlangt, dass Sie sehr lange spielen müssen, um Ihre Dateien entschlüsseln zu können. Patchen Sie das Programm, sodass Sie weniger lange spielen müssen. Werden nun direkt Ihre Dateien entschlüsselt?(0.5 P.)

e) *Entschlüsselung*: Unter Beachtung Ihrer Erkenntnisse aus der Teilaufgabe 4(a-d) und der letzten Teilaufgabe: Entschlüsseln Sie die verschlüsselten Dateien auf dem Desktop. An welchem neuen Projekt hat Ihr Kollege gearbeitet?(0.5 P.)

15 P.

10 + 15 = **25 Punkte**