Der Einstieg ins Anti-Cheating

Version 1.02 by Thorium

1.0) Bevor es losgeht

1.1) Vorwort
Ich habe, bevor ich
diesen Artikel geschrieben habe, nach Informationen zum Thema Anti-Cheating in Computerspielen im World Wide Web gesucht. Leider fand ich dazu nur wenig, und das wenige zu dem Thema ist von miserabler Qualität. Es gibt viele Missverständnisse, Unklarheiten und auch Unwahrheiten im Netz über das Thema Anti-Cheating.
Dieser Artikel richtet sich in erster Linie an Spieleentwickler, die sich noch keine oder nur wenige Gedanken um das Thema gemacht haben. Es geht hierbei nicht darum, wie ein externes Anti-Cheating-Programm
für ein existierendes Spiel entwickelt wird, sondern wie Spieleentwickler selbst ihre Spiele vor Cheatern schützen können.
Ich habe wissentlich auf Codebeispiele verzichtet, da dieser Artikel zum einen unabhängig von Programmiersprachen sein soll und
Anti-Cheating zum anderen eine Sache ist, die von Spiel zu Spiel anders implementiert werden muss. Anti-Cheating kann man nicht
"Copy&Past'en"!

1.2) Der Autor
Für alle, die sich für meine Kompetenz zu diesem Thema interessieren: Mein Name ist Janos David Ommert, bekannt auch als Thorium. Ich wurde am 09.11.1983 geboren und beschäftige mich seit meinem zwölften Lebensjahr mit der Computerprogrammierung.
Angefangen habe ich mit einem C64, der damals zwar schon längst vom PC abgelöst worden war, aber dennoch bis heute mein Lieblichsrechner geblieben ist. Als ich mir meinem ersten PC zulegte, fing ich an mit Turbo Basic zu programmieren, später stieg ich auf Visual Basic um und seit 2006 programmiere ich in PureBasic. Zudem habe ich mir grundlegende Assemblerkenntnisse angeeignet.
Ich bin Mitglied der ersten Stunde bei der Coding Crew "SacredVault" SacredVault.de/, die im Jahre 2005 gegründet wurde und auf Tools spezialisiert ist, welche die Funktionen von kommerziellen Spielen erweitern.
Ich bin der Entwickler des ersten und bis jetzt einzigen Anti-Cheating-Tools für ein Action RPG. Das Tool ist bekannt als "Cheater's Nightmare" und erkennt gehackte Items auf Servern des Spiels "Sacred Underworld" und kickt Cheater automatisch vom Server.

2.0) Der Einstieg ins Anti-Cheating

2.1) Grundlegende Tatsachen
Es ist absolut unmöglich einen unknackbaren Cheatschutz zu entwickeln. Die einzige Art von Spielen, die man absolut cheatsicher machen kann, sind Onlinespiele, bei denen alle Berechnungen zu 100% auf dem Server stattfinden. Doch selbst bei diesen Spielen kann oft gecheatet werden, durch sogenannte Exploits. Auf Exploits werde ich später noch zu sprechen kommen.

Anti-Cheating wird nicht in ein Spiel implementiert um Cheaten zu verhindern, sondern um es so zu erschweren, dass Cheaten
nur mit sehr großem Aufwand und fundierten Kenntnissen möglich ist. Dadurch scheiden schon nahezu alle Cheater aus. ;) Ein unknackbarer Cheatschutz ist allerdings unmöglich zu entwickeln!

Ein externes Anti-Cheating-Tool ist minder wirksam und kann immer umgangen werden. Der Cheatschutz sollte immer Bestandteil des Spiels sein und auch nicht als externe DLL mitgeliefert werden.

2.2) Die Möglichkeiten eines Cheaters
Damit man sein Spiel vor Cheatern schützen kann, muss man natürlich erstmal wissen, welche Möglichkeiten diese haben und wie sie vorgehen. Es gibt sehr viele Arten des Cheatens und ich werde mich hier auf die grundlegenden

Beschränken.

File Editing: Der Cheater bearbeitet die Dateien des Spiels. Er trägt zum Beispiel eine gewünschte Punktzab in die Highscores ein, indem er die Datei sucht und einfach ändert. Oder er analysiert die Struktur von Spielständen um diese zu manipulieren.
Cheater können auch die Struktur von Leveln analysieren und diese zu ihrem Vorteil verändern. Zum Schutz vor "File Editing" eignen sich Verscbüsslungen und Checksummen.

Memory Editing: Der Cheater manipuliert den Speicher, während das Spiel läuft, um die Werte von Variablen zu ändern. Dadurch
lässt sich z.B. die Lebensenergie manipulieren. Einige werden jetzt vielleicht denken, dass dies ziemlich kompliziert ist. Das ist es aber ganz und gar nicht. Es gibt sogenannte "Memory Scanner" die den Cheater bei seiner Arbeit mit machtvollen Funktionen unterstützen. Zum Schutz vor "Memory Editing" eignen sich Variablenverscbüsslungen, Checksummen von Variablen und DMA (Dynamic Memory Allocation)

Memory Freezing: Der Cheater nutzt ein Programm, welches bestimmte Werte im Speicher "einfriert". Das heißt diese Werte
verändern sich nicht, auch wenn das Spiel sie ändert. Dadurch sind Dinge wie der sogenannte 'God Mode' möglich, bei dem der Cheater kein Leben verliert und dadurch unverwundbar/unsterblich ist. Zum Schutz vor "Memory Freezing" eignen sich Checksummen von Variablen.

Packet Editing: Der Cheater manipuliert die Datenpakete, die während eines Netzwerk- oder Internetspiels gesendet und empfangen werden. Er kann Pakete blocken, neue Pakete einscbeusen oder Packete manipulieren. Packet Editing wird gerne verwendet, um Feber zu verursachen, die für den Cheater zu vorteilhaften Ereignisen führen (Exploiting). In Rollenspieln werden gerne Items per "Packet Editing" dubliziert (Duping). Zum Schutz vor "Packet Editing" eignen sich Packetverscbüsslungen und das Prüfen auf Gültigkeit der Pakete.

Exploiting: Auch als Bugusing bekannt. Der Cheater nutzt Bugs im Spiel zu seinem Vorteil oder verursacht z.B. durch "Packet Editing" Feber. Zum Schutz vor "Exploiting" eignet sich sauberes Programmieren *g* und natürlich eine lange Betatestphase.

Debugging: Der Cheater debuggt das Spiel und ändert den Speicher, Registerinhalte oder den Stack, oder, oder, oder... Zum
Schutz vor "Debugging" eigenen sich Anti-Debugger-Methoden, auf welche ich hier aber nicht näher eingehen will, da es dazu ausgezeichnete Artikel im Netz gibt. Einfach mal unter Google suchen. Es ist ein großer Irrtum zu glauben, dass besonders verwirrender und unsinniger Code einen Cheater abhalten oder verwirren kann. Ein Cheater der so gut ist, dass er "Debugging" verwendet, wird das nichtmal ein müdes Lächeln kosten.

Patching: Der Cheater patcht die .exe des Spiels. Er ändert also den Programmcode zu seinem Vorteil. Zum Schutz vor "Patching" eignen sich Laufzeitpacker wie UPX, obwob eher unbekanntere verwendet werden sollten. Am besten sollte ein eigener entwickelt oder ein bestehender verändert werden. Zu Laufzeitpackern gibt es gute Artikel und auch Sources im Netz. Es ist ein großer Irrtum zu glauben, dass besonders verwirrender und unsinniger Code einen Cheater abhalten oder verwirren kann. Ein Cheater der so gut ist das er "Patching" verwendet wird das nichtmal ein müdes lächeln kosten.

Hacking: Sehr seltene Art des Cheatens aber dennoch möglich. Der Server, auf dem der Gameserver läuft, wird gehackt um dem
Gameserver Anweisungen zu geben. Z.B. "Kicke Spieler xy vom Server." Oder aber der Server, auf dem die Online-Highscore liegt wird gehackt, um diese zu manipulieren. Es ist auch möglich, dass der Gamesever selbst eine Sicherheitslücke im Server öffnet. Z.B. könnte man durch manche Spiele Code per "Buffer Overrun" in den Gameserver injizieren.
Zum Schutz vor "Hacking" sollte man die Highscore niemals per FTP zum Server übertragen, da die Zugangsdaten zu dem FTP von einem Cheater mit Leichtigkeit aus dem Spiel extrahiert werden können. Auch eine Verscbüsslung des Passwortes schütz NICHT! Den Server selbst vor Hackern zu sichern ist aber natürlich nicht die Aufgabe des Spieleentwicklers, sondern die des Serverbetreibers.

2.3) Die Möglichkeiten eines Spieleentwicklers
Der Spieleentwickler hat eine Möglichkeit, die weder ein Cheater noch der Entwickler eines externen Anti-Cheating-Tools hat: Er kann seine Anti-Cheating-Maßnahmen direkt in den Source seines Spiels einflechten und kennt alle Informationen seines Spiels. Meiner Meinung nach sitzt der Spieleentwickler daher am längeren Hebel als der Cheater. Cheats und

Hacks können vom Spieleentwickler immer sehr schnell durch einen Patch unwirksam gemacht werden. Daher empfiebt sich der Einbau eines Webupdates für das Spiel, ist aber nicht zwingend nötig.
Ausserdem besteht noch die Möglichkeit, Hilfsmittel von externen Anbietern zu verwenden. Da wären z.B. Laufzeitpacker und Anti-Debugger.

3.0) Anti-Cheating-Methoden

3.1) Verscbüsselung
Eine Verscbüsselung ist die Basis jedes Anti-Cheating-Konzepts. Allerdings ist der Sinn einer Verscbüsselung beim Anti-Cheating ein etwas anderer als bei einer herkömmlichen Anwendung von Verscbüsselungen.
Normalerweise will jemand der etwas verscbüsselt verhindern, dass andere das Verscbüsselte lesen können. Beim Anti-Cheating ist das nicht der Fall! Hier wird die Verscbüsselung genutzt, um Dinge zu verstecken. ;) Wenn ein Cheater verscbüsselte Daten gefunden hat, kann er sie auch entscbüsseln. Der Entscbüsslungsalgorythmus und der Scbüssel müssen zwangsweise im Programm enthalten sein. Und damit hat der Cheater ja schon alles um zu entscbüsseln: Scboß und Scbüssel.
Der Trick ist es, den Cheater diese Daten nicht finden zu lassen. Als erstes wird jeder Cheater versuchen, nach Zielen zu suchen.
Verscbüsseln wir nun potenzielle Ziele, so wird der Cheater sie nicht durch die Suchfunktionen von Hexeditoren, etc. finden.

Was sollte man verscbüsseln?
Spielstände und Charakterdateien enthalten beliebte Ziele eines Cheaters. Diese Dateien sollten komplett verscbüsselt werden. Es ist wichtig, nicht nur bestimmte Werte in den Spielständen zu verscbüsseln, sondern die gesammte Datei, damit der Cheater nicht im Ausscbussverfahren alle Wichtigen Offsets sammeln kann.
Die Highscore sollte auch verscbüsselt werden. Variablen, die potenzielle Cheaterziele darstellen, sollten ebenfalls verscbüsselt werden. Dazu zäben Varibalen, die z.B. die Lebensenergie oder die Anzab der verbleibenden Leben enthalten.
In einem Netzwerk oder Internetspiel sollten die Pakete unbedingt verscbüsselt werden.
Auf jeden Fall müssen sämtliche Zeichenketten, die auf kritische Prozeduren im Programm hinweisen, verscbüsselt werden. Als

Beispiel:
Wir wollen bestätigen das der Spielstand gespeichert wurde. Zu diesem Zweck geben wir die Textnachricht "Spielstand wurde gespeichert." auf dem Bildschirm aus. Der Cheater wird nun versuchen die Prozedur zu finden, die den Spielstand speichert um z.B. die Verscbüsslung des Spielstandes zu knacken. Dazu muss er nur in einem Dissasembler nach der Zeichenkette "Spielstand wurde gespeichert." suchen. Haben wir diese Zeichenkette aber verscbüsselt, so wird die Suche nicht erfolgreich sein.

Wie verscbüsselt man?
Beim Anti-Cheating zäbt nicht wie "gut" die Verscbüsslung ist sondern wie schnell sie ist. Das Verscbüsseln von Variablen kostet Performance, die wir ja fürs eigentliche Spiel brauchen. Die Hauptsache ist, das die Daten unkenntlich sind. Wie leicht die Verscbüsslung zu knacken ist, spielt eine geringere Rolle. Allerdings empfeben sich für Variablen und Pakete wechselnde Scbüssel.
Variablen werden während der Laufzeit gefüllt und dann verscbüsselt. Vor jeder Operation müssen sie entscbüsselt werden und danach wieder verscbüsselt, so dass sie nur während den nötigen Operationen kenntlich sind.
Spielstände und Highscores werden verscbüsselt, bevor sie auf die Festplatte geschrieben werden, nicht danach! Hier kann man auch ruhig komplexere Verscbüsslungsalgorythmen verwenden.
Pakete werden logischerweise direkt bevor sie gesendet werden verscbüsselt und nach dem Empfang wieder entscbüsselt.
Zeichenketten im Programm müssen bereits verscbüsselt im Sourcecode vorliegen, müssen also vor dem Kompilieren erscbüsselt werden. Sie werden dann während der Laufzeit temporär für den direkten Gebrauch entscbüsselt.
Über die genaue Arbeitsweise von Verscbüsslungen will ich hier nichts Näheres schreiben, da es dazu massenhaft Sources, Tutorials und Artikel im Netz gibt. Allerdings sollte auf keinen Fall eine Crypto-API verwendet werden, keine externe DLL. Da sich zu leicht abfangen lässt, was an die jeweilige API-Funktion übergeben wird.

Weiterführende Links
CODY2 - brauchbarer Game-Data encrypter Ist in PureBasic geschrieben.

Inline asm: Traditional XOR-with-password Xor ist die perfekte Verscbüsslung für Variablen, und Zeichenketten, da

extrem schnell. Der Source ist Inline-ASM für PureBasic.

Data Encrpytion Standard (DES) in PureBasic.

3.2) Checksummen
Checksummen sind ein machtvolles Instrument, um zu überprüfen, ob Daten manipuliert wurden. Es gibt verschiedene Arten von Checksummen. Die wob bekanntesten sind CRC32 und MD5. Für Anti-Cheating-Zwecke empfebe ich CRC32, da dieser Algorythmus recht schnell ist. Hier gilt wie bei der Verscbüsslung: Performance ist wichtiger als Sicherheit.

Was sollte mittels Checksummen geprüft werden?
Man kann die .exe des Spiels prüfen, Variablen, Spielstände, die Highscore und auch sonstige Spieldaten wie z.B. die Level.

Wie prüfe ich Daten mitteln Checksummen?
Eine Checksumme wird direkt nach dem erstellen der Daten, aus den Daten gebildet. Bevor die Daten wieder verwendet werden wird erneut eine Checksumme aus ihnen gebildet. Nun werden die alte und die neue Checksumme verglichen. Unterscheiden sie sich wurden die Daten manipuliert. Sources zum CRC32 gibt es im Internet.

3.3) (DMA) Dynamic Memory Allocation
Mit DMA ist gemeint, das der Speicher für Variablen während der Laufzeit dynamisch allokiert wird. Dadurch haben die Variablen bei jedem Programmstart eine andere Speicheradresse, was natürlich großartig fürs Anti-Cheating ist. Viele Cheater haben Probleme, Trainer für DMA-nutzende Spiele zu erstellen. Wie genau DMA funktioniert kann man in vielen Artikeln nacbesen, einfach nach "dynamic memory allocation" bei Google suchen.

Weiterführende Links
Ein Beispielcode von mir für DMA in PureBasic.

Der Artikel Dynamic Memory Allocation and Dynamic Structures erklärt in englisch und in C++ wie DMA funktioniert.

3.4) Variable Obscuring
"Variable Obscuring" ist englisch und bedeutet soviel wie "Variablen verdunkeln". Gemeint sind damit alle Möglichkeiten, die Variablen eines Spieles zu verstecken. Als Erstes ist da die Variablenverscbüsselung zu nennen.
Aber man kann noch viele andere Dinge tun. Eine Möglichkeit für "Variable Obscuring", die ich mir ausgedacht habe, ist die Verwendung von zwei anstatt nur einer Variablen. Das funktioniert folgendermaßen: In der einen Variable befindet sich immer der verscbüsselte Wert. Die andere wird nur gefüllt, wenn eine Operation mit der Variable durchgeführt werden soll. In diesem Fall wird die verscbüsselte Variable in die andere entscbüsselt. Die Operation wird durchgeführt und dann wird die Variable wieder in die andere verscbüsselt. Danach wird die entscbüsselte Variable genullt.

3.5) Anti-Freezing
Was "Memory Freezing" ist habe ich ja schon bei "2.2) Die Möglichkeiten eines Cheaters" erklärt. Anti-Freezing-Methoden erkennen solche "Memory Freezes".

Eine Methode, die von mir entwickelt wurde, wird beim "Variable Obscurings" integriert. Siehe dazu zuerst "3.4) Variable Obscuring". Die Idee ist simpel: Falls ein Cheater nun totzdem die Speicheradressen der beiden Variablen gefunden hat, muss nur überprüft werden, ob sie genullt werden. Dazu ist es aber nötig, beide Variablen zu nullen.
In der einen Variable ist immer der verscbüsselte Wert enthalten. Die andere wird nur gefüllt, wenn eine Operation mit der Variable durchgeführt werden soll. In diesem Fall wird die verscbüsselte Variable in die andere entscbüsselt. Die verscbüsselte Variable wird nun genullt. Dann wird geprüft, ob das auch tatsäcbich geschenen ist, also ob sie im Augenblick 0 ist. Die Operation wird durchgeführt und dann wird die Variable wieder in die andere Verscbüsselt. Nun wird die entscbüsselte Variable genullt und geprüft ob sie wirklich 0 ist. Das ganze wird etwas kompliziert durch die Funktionsweise eines "Memory Freezers". Denn der macht nichts anderes, als permanent den Wert an der Speicheradresse neu zu überschreiben. Das tut er aber in einem gewissen, wenn auch sehr niedrigen Interval. Die Variablen können also nicht direkt nach dem Nullen auf ihren tatsäcbichen Wert - also 0 - überprüft werden. Demnach muss das beim nächsten Durcbauf der Spielscbeife geschehen.
Wem das zu kompliziert ist, kann natürlich auch einfach Checksummen der Variablen erstellen und beim nächsten Scbeifendurcbauf prüfen. Es gibt sicher viele Möglichkeiten für "Variable Obscuring" und "Anti-Freezing", am besten man findet selbst eine. ;)

3.6) Prüfung auf unzulässige Werte

Eine sehr simple und wenig effektive Methode. Es wird einfach geprüft, ob Werte in Variablen prinzipiell unzulässig sind. Ein Beispiel dafür ist, ob die Lebensenergie mehr ist als das Maximum, welches das Spiel zulässt.

3.7) Data Shadowing
Eine sehr machtvolle und effektive Methode um verschiedenste Cheats zu erkennen.
Ich will nur kurz die grobe Funktionsweise von "Data Shadowing" erläutern und nicht näher darauf eingehen, da dies eine sehr komplexe und für Einsteiger ungeeignete Methode ist.Vielleicht werde ich in einer späteren Version näher darauf eingehen.
"Data Shadowing" Ist eine Multiplayer Cheatdetection und funktioniert so, dass jeder Client und auch der Server die Spieldaten jedes Clients besitzt. Nun kann jeder Client die Spieldaten mit den anderen Clients vergleichen; weichen diese ab, wurde ein Cheat gefunden.

3.8) Laufzeitpacker (EXE-Packer)
Laufzeitpacker sind eigentlich dazu gedacht die .exe zu verkleinern. Das funktioniert, indem die Sektionen der .exe komprimiert werden und eine Dekomprimierungsroutine in eine eigene Sektion in die .exe geschrieben wird. Wird die .exe nun gestartet, läuft zuerst die Dekomprimierungsroutine ab, welche die Sektionen dekomprimiert und dann zum eigentlichen Einstiegspunkt des Programms verweist. Für Anti-Cheating ist das deshalb nützlich, weil man die .exe nicht disassemblieren kann, ohne sie vorher entpackt zu haben. Es empfiebt sich einen etwas unbekannteren Laufzeitpacker zu verwenden, da es für die meisten sogenannte "Unpacker" gibt, welche die .exe entpacken. Am besten ist es, wenn man sich seinen eigenen macht. Sources und Artikel gibt es einige im Internet.

Weiterführende Links
Auf der offizielen Seite des wob beliebtesten Laufzeitpackers UPX, gibt es den Source zum Download.

Ashkbiz Danehkars Artikel: Make your own PE Protector ist ein ausgezeichneter Startpunkt für alle, die sich ihren eigenen Laufzeitpacker entwickeln wollen. Allerdings sind dazu gute Englischkenntnise sowie gute Assemblerkenntnise notwendig.

3.9) Anti-Debugging
Dazu werd ich hier jetzt erstmal nichts schreiben, da es haufenweise gute Artikel, Tutorials und Sources im Netz gibt. Das ist aber ein Punkt, der im Anti-Cheating-Konzept berücksichtigt werden sollte.

Weiterführende Links
Der Artikel Anti-Debugging & Software Protection Advice hat viele gute Tipps und Tricks zum Thema Anti-Debugging. Leider in englisch und auf Assembler beschränkt.

Ashkbiz Danehkars Artikel: Make your own PE Protector enthält auch nützliche Tipps zum Anti-Debugging.

4.0) Spezialfälle

4.1) Online-Highscore
Eine Online-Highscore ist Anti-Cheating-technisch schwierig. Auf keinen Fall sollte die Highscore per FTP übertragen werden. Da die FTP-Zugangsdaten sehr leicht aus dem Spiel zu extrahieren sind, egal ob verscbüsselt oder nicht. Die beste Methode ist es, ein Script (z.B. PHP) auf dem Server abzulegen. Dieses nimmt dann den neuen Highscoreeintrag entgegen und speichert ihn in einer Datenbank. Das Problem ist nun natürlich, dass man dieses Script auch manuell über einen Browser aufrufen kann
und ihm einfach irgendwelche Werte gibt. Damit das nicht so ohne weiteres möglich ist, muss die zu übergebende Zeichenkette gut verscbüsselt werden. Es empfiebt sich hier auch mit Checksumme und zufallsgeneriertem Scbüssel zu arbeiten. Ist zwar nicht so einfach aber es lohnt sich.

5.0) Scbusswort
Ich freue mich über jedes Feedback. Wenn euch dieser Artikel genutzt hat und ihr etwas davon in euren Spielen anwenden konntet, so dankt es mir doch bitte, indem ihr einen coolen About oder Creditsdialog macht und meinen Namen dort erwähnt.

5.1) Verbesserungsvorscbäge und Korrekturen
Ich bin für jede Art von Verbesserungsvorscbägen dankbar, würde auch gerne über das Thema Anti-Cheating in Foren diskutieren. Sollte etwas falsch sein oder Rechtschreibfeber gefunden werden, bitte per PN oder e-Mail an Thorium@SacredVault.de

senden. Ich werde das dann korrigieren.

5.2) Kontakt
Ich beantworte gerne Fragen in den Foren von SacredVault SacredVault.de/ und PureBasic. Ausserdem bin ich über Thorium@SacredVault.de erreichbar.

5.3) Copyright