Bundesamt
für Sicherheit in der
Informationstechnik

Analyse der Zufallszahlenerzeugung in virtualisierten Umgebungen

Zusammenfassung

Gute Zufallszahlen sind eine Voraussetzung für die Sicherheit von Daten in Behörden und Unternehmen; sie werden insbesondere bei der Erzeugung von Schlüsseln für die bei Übertragung und Speicherung sensitiver Daten eingesetzten kryptographischen Verfahren benötigt. Mit dem zunehmenden Einsatz virtueller Maschinen, vor allem in Cloud-basierten Lösungen, stellt sich die Frage, ob auch hier Zufallszahlen von ausreichender Qualität bereitgestellt werden können.

In einer Studie des BSI wurde daher untersucht, wie die Virtualisierung die Entropie der Rauschquellen, die die Zufallszahlengeneratoren speisen, beeinflusst, und was getan werden kann, um die Versorgung der virtuellen Maschinen (VM) mit genügend Zufall sicherzustellen. Exemplarisch wurde dabei der quelloffene Zufallszahlengenerator von Linux in virtuellen Maschinen untersucht, die auf verschiedenen virtuellen Maschinenmonitoren (VMM) wie KVM, VirtualBox, Microsoft Hyper-V und VMWare ESXi liefen.

Im Ergebnis war in allen Kombinationen bei entsprechender Konfiguration eine ausreichende Entropie-Versorgung der Linux-VMs möglich. Unterschiedliche Rauschquellen erfüllten ihre Aufgabe allerdings unterschiedlich gut, so dass sich je nach Einsatzszenario durchaus Probleme ergeben können, zum Beispiel für die Qualität der Zufallszahlen kurz nach dem Systemstart. Mit einem Fragenkatalog werden Anwender daher in die Lage versetzt, selbst zu analysieren, ob solche Probleme auf sie zukommen und vorab die kritischen Informationen von ihrem Systemlieferanten einfordern.

Prinzipiell müssen Anwender ihrem VMM (und dessen Betreiber) vertrauen und sollten sich nicht auf eine einzige Rauschquelle verlassen. Software-basierte Rauschquellen, die Hardware-Unterstützung für ihre Entropiegewinnung brauchen, können am ehesten problematisch sein und müssen daher am genauesten auf ihre Eignung für die Einsatzumgebung geprüft werden. Software-basierte Rauschquellen, die hochauflösende Zeitstempel zu Systemereignissen auswerten, funktionieren in virtuellen Umgebungen genauso gut und bisweilen sogar besser als in nicht-virtualiserten Umgebungen. Hardware-Rauschquellen bleiben von der Virtualisierung meist unberührt. Bei einer geeigneten Kombination kann der VMM sein Gastsystem bei der Gewinnung von Entropie auch unterstützen.

Die vorliegende Analyse beginnt mit einer Untersuchung der Architektur verschiedener Rauschquellen, betrachtet dann den Einfluss der Virtualisierung auf den bereitgestellten Zufall und erläutert abschließend die Ergebnisse der Tests mit dem Linux-RNG.

Autoren

Stephan Müller, atsec information security GmbH

Gerald Krummeck, atsec information security GmbH

Helmut Kurth, atsec information security GmbH

BSI-Referenz

BSI-Titel: Analyse der Zufallszahlenerzeugung in virtualisierten Umgebungen

BSI-Projektnummer: 213

Dokumentenhistorie

Version	Änderungs-datum	Autor(en)	Änderungen zur vorherigen Version
1.0	2016-10-21	Müller, Stephan	Erste Veröffentlichung

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Zielsetzung

Random numbers should not be generated with a method chosen at random.

Donald E.Knuth
The Art of Computer Programming

Kryptographische Verfahren sind heute unerlässlich für die Gewährleistung von Vertraulichkeit, Integrität und Authentizität digital verarbeiteter Daten. Zufallszahlen sind unverzichtbare Kernbestandteile dieser Krypto-Systeme. Sie werden für die Erzeugung kryptographisch sicherer Parameter, insbesondere des Schlüsselmaterials, benötigt und müssen daher auf ihre kryptographische Eignung hin untersucht und beurteilt werden. Nur so kann die Sicherheit der entsprechenden Krypto-Systeme gewährleistet werden.

Die Beurteilung der Eignung und Qualität kryptographischer Systeme ist in Deutschland Aufgabe des Bundesamtes für Sicherheit in der Informationstechnik (BSI), das daher auch diese Studie über die Gewinnung von Entropie in virtuellen Maschinen in Auftrag gegeben hat. Virtuelle Maschinen werden immer häufiger, vor allem in Cloud-basierten Lösungen, verwendet, die in sensiblen Bereichen von Wirtschaft und Verwaltung einschließlich der kritischen Infrastrukturen zum Einsatz kommen. Gute Zufallszahlen sind eine Voraussetzung für die Sicherheit der Daten in Behörden und Unternehmen und bedingen eine oder mehrere Rauschquellen, die ausreichend Entropie bereitstellen.

Betriebssysteme nutzen verschiedenartige Rauschquellen, die iinnerhalb einer virtuellen Maschine Verhaltensweisen zeigen können, welche teilweise erheblich von denselben Rauschquellen auf nicht-virtualisierten Systemen abweichen. Im Rahmen dieser Analyse wird untersucht, wie sich die Entropie einer Rauschquelle verändert, wenn sie in virtuellen Umgebungen eingesetzt wird. Ziel dieser Analyse ist es Maßnahmen aufzuzeigen, wie Rauschquellen in virtuellen Maschinen so eingesetzt werden können, dass ausreichend Entropie aufgenommen wird.

Von besonderem Interesse ist bei dieser Untersuchung neben der Untersuchung des generellen Einflusses von virtuellen Umgebungen auf Rauschquellen die Diskussion des Linux-Zufallszahlengenerators /dev/random und /dev/urandom. Des Weiteren werden Möglichkeiten erörtert, Zufall vom Monitor für virtuellen Maschinen (VMM) zu erhalten, als auch Rauschquellen, die unbeeinflusst von der virtuellen Umgebung Entropie anbieten. Ziel ist es, Entropie in ausreichender Größe in virtuellen Umgebungen bereitzustellen.

Dieser Bericht wurde von der atsec information security GmbH im Auftrag des BSI unter der BSI-Projektnummer 213 angefertigt.

1.2 Struktur der Analyse

Diese Analyse ist in zwei Teile – und damit auch in zwei Dokumente – aufgeteilt:

1. Der erste, deutschsprachige Teil enthält eine Zusammenfassung des zweiten Teils inklusive der Resultate zu den quantitativen Tests des Linux-Zufallszahlengenerators in virtuellen Umgebungen. Dieser erste Teil der Analyse liegt in diesem Dokument vor.
2. Der zweite, englischsprachige Teil beinhaltet eine Zusammenfassung, welche konsistent mit der deutschen Zusammenfassung ist, und umfasst die detaillierte Analyse aller Aspekte und die daraus abgeleiteten Aussagen. Der englischsprachige Teil ist in einem separaten Dokument zu finden.

Die vorliegende Zusammenfassung enthält die folgenden Teile, welche größtenteils aufeinander aufbauen. Dabei bieten diese Kapitel einen generellen Überblick über die verschiedenen Themenbereiche. Eine eingehende Untersuchung wird mittels des englischsprachigen Dokuments bereitgestellt, welches eine ähnliche Struktur aufweist.

- Kapitel 2 erläutert die Architektur von Rauschquellen

- Im Kapitel 3 wird mit der Kenntnis der Architektur der Rauschquellen analysiert, in wieweit ein VMM die Funktionsweise der Rauschquellen beeinflussen kann.
- Kapitel 4 wendet die gewonnenen Erkenntnisse auf den Linux-Zufallszahlengenerator an.

2 Architektur der Rauschquellen

Bevor der Einfluss eines virtuellen Maschinenmonitors (VMM) auf eine Rauschquelle untersucht werden kann, wird ein klares Modell einer Rauschquelle benötigt.

Mittels dieses Modells können die Bestandteile einer Rauschquelle identifiziert und später genauer untersucht werden, die von einem VMM beeinflusst werden. Bestandteile, die hingegen unbeeinflusst von einem VMM arbeiten, können bei einer späteren Analyse außen vor gelassen werden.

Zusätzlich kann ein klares Modell einer Rauschquelle und Kenntnisse von der Art der, wie ein VMM eine Rauschquelle beeinflusst, dem Leser helfen, die Resultate auf andere, nicht in diesem Dokument besprochene Rauschquellen zu übertragen.

Im Folgenden wird die generelle Architektur einer Rauschquelle dargestellt. Der englischsprachige Teil erweitert diese Darstellung, indem dieses Modell realen Rauschquellen gegenübergestellt wird.

2.1 Generelle Architektur einer Rauschquelle

Rauschquellen haben verschiedene Formen, einschließlich:

- Physische Rauschquellen, welche rein für die Generierung von entropischen Datenströmen verwendet werden. Solche Arten von physischen Geräten finden sich in Smartcards, speziellen Schaltkreisen, Hardware Security Modules (HSMs), etc.
- Rauschquellen, welche das Verhalten von Hardwareereignissen auswerten. Dies schließt die Messung des Zeitverhaltens von Eingabegeräten (Mausbewegungen, Tastaturnutzung), Blockgeräte (Festplatten) oder Interrupts ein.
- Rauschquellen, die Mechanismen der CPU für ihre Funktion nutzen, einschließlich zeitbasierter Rauschquellen, CPU Instruktionen wie Intel's RDRAND und andere.

Unabhängig von der Art der Rauschquellen illustriert Abbildung 1 das Konzept einer Rauschquelle, welches sich auf alle Rauschquellen übertragen lässt.

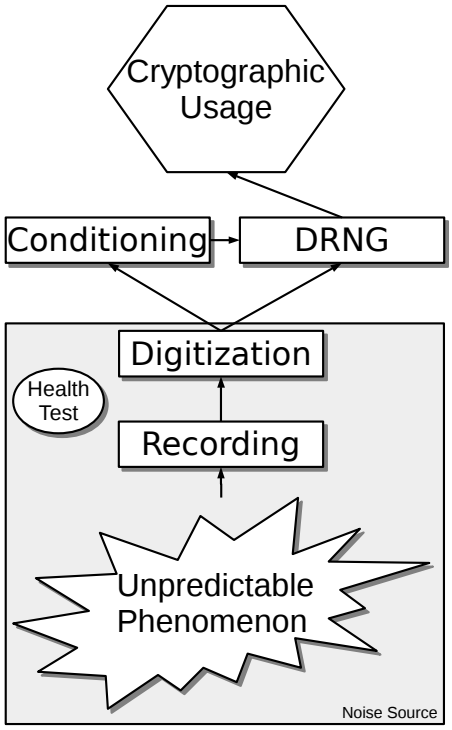

Abbildung 1: Architektur einer Rauschquelle

Abbildung 1 zeigt den gesamten Informationsfluss in einer Rauschquelle einschließlich der Generierung von Zufallszahlen. Die Quelle jeglicher Zufallszahlen ist die Rauschquelle, welche grau in der Abbildung markiert ist. Die von der Rauschquelle generierten Daten werden von einem deterministischen Zufallszahlengenerator (DRNG) weiterverarbeitet. In einigen Implementierungen wird ein „Conditioner“ verwendet, bevor Daten vom DRNG verarbeitet werden.

Es ist möglich und auch oft der Fall, dass mehrere DRNGs verkettet werden, die Daten der Rauschquelle verarbeiten.

Der graue Bereich in Abbildung 1 beinhaltet die konzeptionellen Komponenten einer Rauschquelle. Dieser grau markierte Bereich ist das Thema dieser gesamten Studie. Der Conditioner, der DRNG oder Anwendungen, die die Daten verwenden, werden in dieser Studie nicht betrachtet. Conditioners oder DRNGs fügen keine Entropie hinzu, da sie die erhaltenen Daten ausschließlich verwürfeln. Sie können aber verwendet werden, um das Entropie-pro-Bit Ratio zu erhöhen, indem eine kryptographische Kompression eingesetzt wird.

Die Architektur einer Rauschquelle entsprechend Abbildung 1 enthält folgende Teile:

- Ein Phänomen wird überwacht, welches nicht-vorhersagbare oder teilweise nicht-vorhersagbare Muster aufweist. Es ist wichtig zu verstehen, dass sich die Unvorhersagbarkeit immer auf den Beobachter bezieht und damit unterschiedliche Grade an Unvorhersagbarkeit aufweisen kann, abhängig von der Art und dem Wissen des Beobachters. Damit ist die Unvorhersagbarkeit und die damit verbundene Entropie **relativ** zum Beobachter. Eine Reihe von Phänomenen kann ein vollständig deterministisches Verhalten aufweisen, sofern alle Parameter vollständig bekannt sind, die das Phänomen beeinflussen. Solche Rauschquellen basieren darauf, dass einer oder mehrere dieser Parameter nicht mit der notwendigen Genauigkeit vorhersagbar sind.
  Unvorhersagbare Phänomene können wie folgt klassifiziert werden:
  - ein physisches Phänomen, bei dem die Physik die Unvorhersagbarkeit diktiert, wie zum Beispiel thermisches Rauschen, die Metastabilität in bi-stabilen Schaltkreisen, oder auch radioaktiver Zerfall[1];
  - ein Phänomen basierend auf der Interaktion zwischen Computerhardware und der Umgebung. Als Beispiel dient hier die Interaktion mit menschlichen Benutzern, bei der Interrupts ausgelöst werden und deren zeitliches Eintreffen unvorhersagbar ist.
- Eine Logik zur Messung eines unvorhersagbaren Phänomens ist notwendig, um Daten aus dem Verhalten des Phänomens zu generieren.
- Die aufgezeichneten Ereignisse werden mittels einer Digitalisierungslogik in einen binären Datenstrom umgesetzt, damit diese Daten von nachfolgenden digitalen Komponenten, wie zum Beispiel einem DRNG, weiterverarbeitet werden können. Die Nutzung eines DRNG zur Verarbeitung der Daten aus der Digitalisierungslogik hat nicht das Ziel, die Entropie über ein großes zu generierende Datenvolumen zu verteilen. Hingegen ist das Ziel eines DRNGs an dieser Stelle gleich dem eines Conditioners, der im Folgenden diskutiert wird. Ein Conditioner wie auch ein DRNG wandelt die Daten, welche die gesammelte Entropie erhalten, mittels kryptographischer Operationen in ein Weißes Rauschen um. Wie bereits angedeutet soll hiermit das Entropie-pro-Bit Ratio mittels kryptographischer Kompression erhöht werden. Desweiteren soll ein Conditioner mögliche Schiefen der Rauschquellendaten verschleiern.
- Es ist allgemein empfohlen – und vom BSI gefordert – einen Laufzeittest gegen einen möglichen Ausfall der Rauschquelle zu implementieren. Es ist klar, dass solch ein Test keine Veränderung der Entropierate der Rauschquelle entdecken kann. Dennoch kann solch ein Test, der auf die Rauschquelle zugeschnittene statistische Prüfungen

1 Auch wenn radioaktiver Zerfaln ein gutes Beispiel für ein unvorhersagbares Phänomen mit einer bewiesenen physikalischen Theorie ist, ist den Autoren klar, dass Rauschquellen auf Basis radioaktivem Zerfalls in normalen IT Systemen nicht praktikabel sind. Dieses Beispiel soll nur zur Untermauerung der Aussagen dienen.

durchführt, nicht-tolerierbare Defekte im stochastischen Verhalten der Rauschquelle innerhalb einer akzeptablen Zeit erkennen.

2.2 Designs realer Rauschquellen

Das beschriebene Modell der Rauschquellen kann nun auf verschiedene reale Rauschquellen angewendet werden. Dabei ist immer wieder erkennbar, dass die im Modell stilisierten Komponenten einer Rauschquelle immer in realen Rauschquellen sichtbar werden. Die folgenden Rauschquellen wurden dahingehend untersucht, ob das definierte Modell auf sie andwendbar und damit korrekt ist.

Generische Rauschquellendesigns, die analysiert wurden und mit dem Modell korrespondieren, sind:

- Hardware-Rauschquelle: Ringoszillator
- Software-Rauschquelle: Messung der Eintrittszeit von Ereignissen

Darüber hinaus sind folgende real existierende Rauschquellen geprüft worden:

- Linux /dev/random und /dev/urandom
- Intel RDRAND und RDSEED Instruktionen
- CPU Execution Time Jitter RNG
- Apple Mac OS Rauschquelle

Die Untersuchung der Rauschquellen hat neben einem Nachweis der Tauglichkeit des Modells weitere Informationen geliefert:

- Implementierungsart, also ob die Rauschquelle eine reine Hardwarekomponente, eine reine Softwarekomponente oder ein Hybridsystem bestehend aus Hard- und Softwarekomponenten ist.
- Für eine Hardware-Implementierung wurden die Schnittstellen zur Software aufgelistet.
- Für Software-Implementierungen wurden in der Analyse jeweils eine Liste von speziellen Hardwaremechanismen angefertigt, die die Rauschquelle unterstützen.

3 Einfluss eines virtuellen Maschinenmonitors auf Rauschquellen

Ein virtueller Maschinenmonitor (VMM) stellt eine Abstraktionsschicht zwischen Betriebssystemen mit deren Software und der Hardware bereit. Der VMM hat als Ziel, Teile oder die Gesamtheit der Hardware zu virtualisieren, um eine gleichzeitige Ausführung von mehreren und gegebenenfalls unterschiedlichen Betriebssystemen zu gewährleisten.

Als erster Schritt in der Analyse zum Einfluss von VMMs auf Rauschquellen werden die relevanten Eigenschaften eines VMM beleuchtet, welche einen Einfluss auf Rauschquellen haben können. In einem zweiten Schritt werden diese Eigenschaften mit dem in Kapitel 2 vorgestellten Modell einer Rauschquelle verknüpft, um den theoretischen Einfluss eines VMM zu verstehen. Im Anschluss an die theoretische Betrachtung werden reale VMMs mit deren praktischem Einfluss auf Rauschquellen analysiert.

3.1 Abbildung von Ressourcenzugriffen durch VMM

Damit ein paralleler Zugriff auf Ressourcen seitens der Betriebssysteme möglich ist, muss der VMM die Zugriffe kontrollieren und steuern. Diese Steuerung kann mittels Abbildung 2 visualisiert werden.

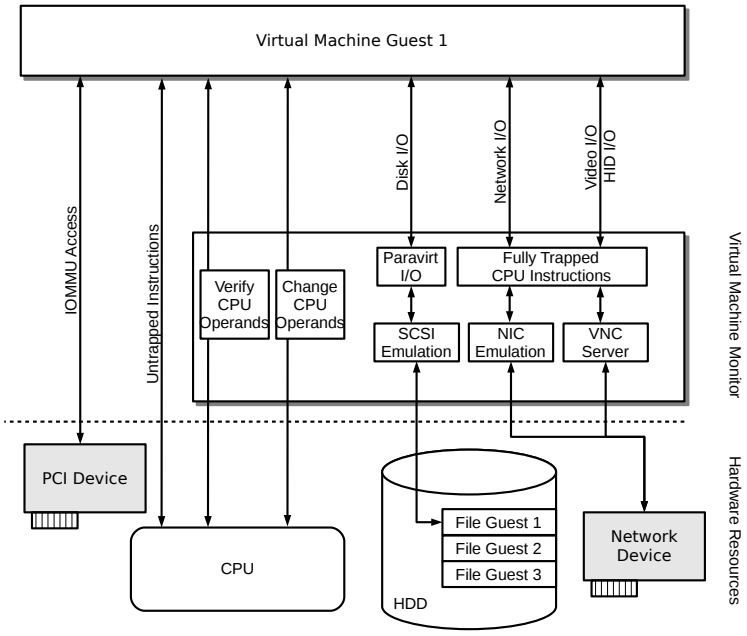

Abbildung 2: Steuerung der Ressourcenzugriffe seitens der Betriebssysteme durch den VMM

Abbildung 2 zeigt einen Gast, welcher verschiedene Zugriffe auf Hardwareressourcen durchführt, die mittels Pfeilen dargestellt sind. Im Folgenden werden die verschiedenen Zugriffe analysiert.

3.1.1 VMM Zugriffssteuerung zu Ressourcen

Um die Zugriffe auf Ressourcen, welche von Gast-Betriebssystemen initiiert wurden, zu steuern, muss der VMM die verfügbaren Ressourcen wie folgt klassifizieren:

- Hardwareressourcen, die ausschließlich vom VMM erreichbar sind: Der VMM kontrolliert diese Ressourcen vollständig und kann sie wie folgt den Gast-Betriebssystemen verfügbar machen:
  Der VMM emuliert ein Gerät und dessen Schnittstelle, und stellt diese Schnittstelle einem oder mehreren Gast-Betriebssystemen zur Verfügung. Dieses emulierte Gerät existiert aber ausschließlich in Software. Dabei unterscheidet man folgende grundlegende Ansätze der Virtualisierung:

    - Paravirtualisierung: Der VMM exportiert eine Geräteschnittstelle, die speziell auf die Verwendung von Gast-Betriebssystemen ausgelegt ist. Ziel ist, den Aufwand für die Nutzung dieser Schnittstelle sowohl auf Gast-, als auch auf VMM-Seite zu minimieren. Dabei verfügt der Gast über einen auf diese Geräteschnittstelle zugeschnittenen Treiber.
      Ein Beispiel hierfür ist VirtIO.
    - Volle Virtualisierung: Der VMM exportiert eine Geräteschnittstelle, die ein Hardwaregerät emuliert. Dabei kann das Gast-Betriebssystem einen vorhandenen Treiber verwenden.
      Ein Beispiel für diesen Ansatz ist eine VESA Graphikkarte, die vom VMM emuliert wird und vom Gast-Betriebssystem verwendet wird. Dabei ist es nicht notwendig, dass eine VESA Graphikkarte physisch im System präsent ist.

- Hardwareressourcen, die ausschließlich einem Gast zugeordnet sind: Eine 1:1 Verbindung zwischen einer Ressource und einem Gast wird vom VMM konfiguriert und erzwungen. Bei der Konfiguration seitens des VMM wird die Hardware so eingerichtet, dass die Hardware die notwendige Isolation bereitstellt – beispielsweise kann dies mittels Nutzung der Input/Output Memory Management Unit (IOMMU) umgesetzt werden. Dabei kann beispielsweise eine PCI-Karte exklusiv einem Gast zugeordnet werden. In diesem Falle fängt der VMM die Kommunikation zwischen Ressource und Gast nicht ab.
- Hardwareressourcen, die von Gästen geteilt werden: Solch eine Teilung einer Ressource ist normalerweise vom VMM mittels Serialisierung der Zugriffe implementiert. Dabei weist der VMM die Ressource für einen definierten Zeitraum einem Gast zu, während dessen der Gast diese Ressource exklusiv verwenden kann. Wenn der Zeitraum abläuft, unterbricht der VMM die Verbindung zum Gast und ordnet die Ressource einem anderen Gast zu.
  Das Paradebeispiel für diese Art der Ressourcenverwaltung ist die CPU selbst. Ein weiteres Beispiel ist der Arbeitsspeicher.

3.2 VMM Einfluss auf Rauschquellen

Um nun ein Verständnis zu erlangen, wie ein VMM die Rauschquellen beeinflusst, muss folgende Frage geklärt werden: Unterliegen die Zugriffe auf Ressourcen, welche für die Rauschquelle und dessen kryptographischer Stärke benötigt werden, einem Eingreifen des VMMs?

Diese Frage kann nur für eine gegebene Rauschquelle beantwortet werden, für welche alle Zugriffe auf die Hardware seitens der Software erkannt werden.

Generell kann auf theoretischer Ebene aber folgendes festgehalten werden:

1. Im Falle eines Gasts, der eine reine Software-Logik implementiert, welche nur unprivilegierte CPU Instruktionen mit möglichen Speicherzugriffen verwendet, ist die Wahrscheinlichkeit, dass der VMM in diese Operation eingreift, gering oder gar nicht existent. Falls ein Eingreifen stattfindet, wird der VMM nur die CPU Instruktion dahingehend prüfen, ob sie oder deren Operanden die Regeln des VMM verletzen. Dabei wird keine Änderung der Instruktion oder der Operanden vorgenommen. Beispielsweise wird der VMM bei einer Softwareimplementierung von SHA-256 mit großer Wahrscheinlichkeit nicht eingreifen.
2. Operationen, welche ausschließlich in Hardware passieren, sind ebenfalls von einem VMM-Eingreifen geschützt. Um es klarer auszudrücken: der VMM hat keine Möglichkeit, diese Operationen zu beeinflussen. Ähnlich verhält es sich mit Logik, welche von einer Firmware ausgeführt wird und Hardwaregatter verwendet, welche für den VMM nicht erreichbar sind. Beispielsweise implementieren manche Netzwerkkarten eine Firmware und haben einen eigenen Prozessor. Die Ausführung der Firmware auf diesem dezidierten Prozessor kann seitens VMM nicht beeinflusst werden.
3. Ein VMM kann nur dann in Operationen eingreifen, wenn diese die Schnittstelle zwischen Hardwareressourcen und Software nutzen. Da ein Gast immer eine Softwarekomponente ist, kann und wird der VMM den Zugriff des Gasts auf die

Hardware steuern und beeinflussen. Wenn nun die Rauschquelle solch einen Zugriff auf Ressourcen benötigt, unterliegt sie dem Einfluss des VMM.

3.3 VMM Einfluss auf die Entropie der Rauschquellen

Neben dem Einfluss des VMM auf die Funktionsweise einer Rauschquelle stellt sich naturgemäß auch die Frage, in wieweit der VMM die Entropie der Daten aus einer Rauschquelle verändert. Diese Frage kann man nur für jeweils eine spezifische Kombinationen aus Rauschquelle und eingesetztem VMM beantworten.

Die vorliegende Studie stellt solch eine Analyse für den Linux-Zufallszahlengenerator /dev/random und /dev/urandom bereit. Dabei verfolgt diese Analyse zwei Ziele:
Zum einen soll der Linux-Zufallszahlengenerator innerhalb virtueller Umgebungen analysiert werden. Zum anderen soll dem Leser aufgezeigt werden, wie solch eine Analyse durchgeführt werden kann, damit andere Rauschgeneratoren in einer ähnlichen Art und Weise untersucht werden können.

4 Verhalten des Linux-Zufallszahlengenerators in virtuellen Umgebungen

Mit den Kenntnissen aus den Überlegungen der vorangegangenen Kapitel kann nun der Linux-Zufallszahlengenerator untersucht werden.

Die Untersuchung beinhaltet sowohl qualitative Analysen, in wieweit ein VMM den Linux-Zufallszahlengenerator beeinflusst, als auch quantitative Messungen zur Abschätzung der Größe der Beeinflussung. Mit dieser Untersuchung werden auch Hinweise zur Konfiguration eines VMMs gegeben.

Bevor die Analysen und Tests durchgeführt werden, ist eine Klärung des Testansatzes wichtig.

4.1 Testansatz

Um quantitative Messungen vom Verhalten des Linux-Zufallszahlengenerators zu erhalten, muss dieser über die Laufzeit überwacht werden. Solch eine Überwachung kann nur mit einer Instrumentierung des Linux Kerns durchgeführt werden, welche die genutzten Speicherinhalte extrahiert. Wichtig hierbei ist aber auch, dass solch eine Instrumentierung die Messungen nicht verfälscht.

Moderne Linux Kern-Versionen implementieren eine Reihe von Analysemechanismen, welche zur Laufzeit Teile des Kerns überwachen können.

Für die Tests des Linux-Zufallszahlengenerators wurde der SystemTap Mechanismus ausgewählt. SystemTap erlaubt es, Variablen zur Laufzeit nach Erreichen vordefinierter Funktionen auszulesen oder sogar zu verändern. SystemTap ist darüber hinaus einfach zu bedienen, da es eine Shell-artige Programmiersprache verwendet. Es stellt damit eine flexible Testumgebung bereit, die alle Anforderungen zum Testen des Linux-Zufallszahlengenerators umsetzt.

SystemTap wurde bereits erfolgreich für die Analyse des Linux-Zufallszahlengenerators in nativen Umgebungen verwendet.

4.2 Testdurchführung

Der Linux-Zufallszahlengenerator nutzt verschiedene Rauschquellen, die von jeweils einer C-Funktion implementiert werden. Jede dieser Rauschquellen nutzt das Auftreten von Ereignissen bei Hardwareressourcen als Entropiequelle.

Zur Analyse des Linux-Zufallszahlengenerators werden die verschiedenen Rauschquellen separat getestet, da sie auch unabhängig voneinander operieren. Folgende Rauschquellen werden detailliert untersucht:

- Blockgeräte
- Human Interface Device (HID) – Tastatur, Maus u.ä.
- Interrupts

Es bleibt festzuhalten, dass der Linux-Zufallzahlengenerator zwei weitere Rauschquellen implementiert:Nutzung von Gerätetreiberdaten und Hardware-basierte Zufallszahlengeneratoren.
Daten aus ersterer Quelle werden immer Null Bit Entropie beigemessen. Damit wird diese Rauschquelle als irrelevant angesehen und nicht weiter betrachtet. Die Quelle aus Hardware-basierten Zufallszahlengeneratoren wird von spezieller Hardware gespeist, welche jedoch nicht weit verbreitet ist. Auf den Testsystemen sind diese Hardwarekomponenten nicht vorhanden und können deshalb auch nicht getestet werden.

Jede der identifizierten Rauschquellen wird nun separat auf den folgenden VMMs getestet:

- KVM/QEMU konfiguriert vom libvirt Verwaltungswerkzeug
- Oracle VirtualBox
- Microsoft Hyper-V

- VMWare ESXi

4.3 Testresultate

Die Testresultate erlauben die folgenden Interpretationen:

- Die Rauschquellen des Linux-Zufallszahlengenerators zeichnen jeweils verschiedene Werte auf. Allen gemeinsam ist, dass ein hoch-auflösender Zeitstempel verwendet wird. Im Rahmen der Untersuchung stellt sich heraus, dass nur dieser ausreichend Entropie bereitstellt. Die anderen aufgezeichneten Werte (Jiffies, Ereigniswert) enthalten kaum oder gar keine Entropie.
- Der Entropieschätzer basierend auf der ersten, zweiten und dritten Ableitung der Jiffies Zeit von /dev/random funktioniert innerhalb einer virtuellen Umgebung. Die durchschnittlichen heuristischen Entropiewerte, welche mit dem Entropieschätzer ermittelt werden, liegen signifikant unterhalb der Entropiewerte, die vom hoch-auflösenden Zeitstempel bereitgestellt werden. Demzufolge wird die vorhandene Entropie signifikant unterschätzt.
- Die Blockgeräte-Rauschquelle wird für auf VirtIO basierende Blockgeräte deaktiviert.
- In virtuellen Gästen ist es sehr wahrscheinlich, dass verschiedene Ressourcen nicht verfügbar sind und damit die Rauschquellen kaum oder gar keine Entropie liefern können. Zum Beispiel ist es zu erwarten, dass dem Gast keine Konsole bereitgestellt wird und somit auch keine HID-Geräte vorhanden sind. Damit fällt somit die Rauschquelle basierend auf HID-Geräten aus.
- Im Rahmen der Linux Unterstützung für Hyper-V wurde ein Programmierfehler entdeckt. Unter Hyper-V mit paravirtualisierten Geräten werden keine Interrupts als Rauschquelle verwendet. Ein Patch zur Behebung dieses Fehlers ist bereitgestellt worden. Dieser Patch wird derzeit in den Kernel aufgenommen und auf ältere Kern-Versionen zurückportiert.
- Verschiedene VMM-Konfigurationen haben einen messbaren Einfluss auf die gesammelte Entropie-Rate. Dies betrifft speziell die Blockgeräte, bei denen der VMM gegebenenfalls mit aktiviertem und deaktiviertem Puffer-Cache betrieben werden kann. Dennoch wurde keine Konfiguration der getesteten VMMs entdeckt, bei der alle Rauschquellen ausfallen.

5 Gewonnene Erkenntnisse

Im Rahmen der Untersuchung wurden folgende zentralen Erkenntnisse gewonnen:

- Rauschquellen, die komplett in Hardware oder Firmware implementiert sind und logisch unterhalb von virtuellen Maschinenmonitoren (VMM) operieren, werden von VMMs nicht beeinflusst. Dennoch kann der VMM auf die für die Software bereitgestellten Schnittstellen dieser Rauschquellen verändernd einwirken und damit die von der Software gelesenen Zufallsdaten beeinflussen.
- Rauschquellen, die komplett in Software implementiert sind und keine spezielle Hardwarefunktion, wie präziser Zeitstempel, benötigen, werden von einem VMM üblicherweise nur in ihrem Zeitverhalten beeinflusst, wenn überhaupt.
- Rauschquellen, die in Software implementiert sind und Unterstützung von Hardware benötigen sind üblicherweise massiv von der Arbeitsweise eines VMM betroffen. Detaillierte Analysen müssen für eine solche Rauschquelle angefertigt werden, ob sie immer noch ausreichend Entropie in einer virtualisierten Umgebung bereitstellt. Ein Teilaspekt dieser Analyse muss die Untersuchung sein, ob das der Rauschquelle zugrunde liegende unvorhersagbare Phänomen überhaupt sichtbar und nutzbar ist. Dazu ist im Anhang der Analyse ein Fragenkatalog bereitgestellt worden, dessen Beantwortung eine solche Analyse unterstützt.
- VMMs ermöglichen eine breite Konfiguration von verschiedenen Aspekten sowohl der Gastumgebung als auch des VMMs selber. In diesem Rahmen können verschiedene für die Arbeitsweise einer Rauschquelle gefährliche Konfigurationen auftreten. Eine Liste dieser problematischen Konfigurationen und deren Einfluss auf Rauschquellen wird mit der Analyse bereitgestellt.
- Rauschquellen, die Entropie aus der Unvorhersagbarkeit der exakten Eintrittszeit eines Ereignisses beziehen – wie die aller meisten betriebssystem-internen Rauschquellen – werden definitiv durch einen VMM im Zeitverhalten verändert. Die Änderung des Zeitverhaltens – unter Ausblendung aller möglichen anderen Beeinflussungen einer VMM – verändert die gewonnene Entropie im ungünstigsten Fall gar nicht. Normalerweise wird aber die gewonnene Entropie durch die VMM Interaktion erhöht.
- Im Rahmen der Untersuchung sind die verschiedenen Rauschquellen, die dem Linux /dev/random und /dev/urandom zugrunde liegen, qualitativ und quantitativ untersucht worden, wie sie von KVM, VirtualBox, Microsoft Hyper-V und VMWare ESXi beeinflusst werden. Hierbei wurden keine pathologischen Konfigurationen der genannten VMMs entdeckt. Dies bedeutet, dass jegliche Konfiguration immer mindestens eine Rauschquelle aktiv halten. Es wurden darüber hinaus folgende grundlegende Resultate ermittelt:
  - Üblicherweise fallen eine oder zwei der drei vorhandenen Rauschquellen aus. /dev/random wird nur dahingehend beeinflusst, dass die Zeitspannen, in denen ein lesender Prozess blockiert wird, sich massiv vergrößern. Für /dev/urandom hingegen verschlechtert sich die kryptographische Stärke der bereitgestellten Zufallszahlen wie folgt massiv. /dev/urandom erhält während der Initialisierung erheblich später einen einen Seed, der kryptographisch ausreichend stark ist. Während den Messungen wurden Werte von weit über einer Minute nach dem Startvorgang für das ausreichende initiale Seeding ermittelt. Damit werden alle zur Startzeit des Betriebssystems gestarteten Programme, die von /dev/urandom Daten beziehen, kryptographisch erheblich schwächere Zufallszahlen beziehen.
  - Das der Blockgeräte-Rauschquelle zugrunde liegende unvorhersagbare Phänomen (drehende Festplatten mit deren physikalischen Varianzen) wird durch die VMM Interaktion durch ein komplett anderes unvorhersagbares Phänomen (Varianzen in der Ausführungszeit von CPU Instruktionen) ersetzt, wenn die VMM mit einem Puffer Cache arbeitet.
  - Die Heuristik der Entropieschätzung funktioniert fast unverändert. Die vorhandene Entropie wird immer noch signifikant unterschätzt.

- Der VMM kann ein Gast bei der Entropiegewinnung unterstützen, indem er eigene Entropie an den Gast weiterleitet. Die Implementierung dieses Konzepts in KVM/QEMU für Linux wurde im als Beispiel detailliert dargestellt.

Appendix A Literaturverzeichnis

[LRNG] LRNG-Studie