LINUX-DISTRIBUTION ZUR SICHEREN ERSTELLUNG VON COLD STORAGE WALLETS

Lucas Johns
Hochschule Mittweida, Technikumplatz 17, D-09648 Mittweida

Dieses Paper beschreibt die Implementierung eines Live-Systems für die Erstellung von Cold Storage Wallets. Ziel soll es sein, einen sicheren und einfachen Erstellungsprozess von Paper-Wallets unter hohen Sicherheitsansprüchen zu ermöglichen. Der Quellcode ist abrufbar unter *https://github.com/envake/vinktar-live*.

This paper describes the implementation of a live distribution for the creation of cold storage wallets. The aim of this implementation is to simplify the process of creating paper wallets under high security requirements. The source code is available under *https://github.com/envake/vinktar-live*.

1. Einleitung

Um Kryptowährungen sicher aufzubewahren, hat sich das Speichern der Private Keys in sogenannten Cold Storage Wallets bewährt. Das sind Wallets, die zu keinem Zeitpunkt mit dem Internet verbunden sind. Die Private Keys werden dabei entweder auf spezieller Hardware gespeichert oder können alternativ auch ausgedruckt werden. Letzteres wird als Paper-Wallet bezeichnet. Da die Anschaffung eines Hardware-Wallets in der Regel mit einem Kostenaufwand verbunden ist, werden häufig einfach Paper-Wallets zur längeren Lagerung von Kryptowährungen eingesetzt. Um diese Paper-Wallets zu erstellen, muss ein gültiges Schlüsselpaar, bestehend aus Private- und Public Key generiert werden. Für alle gängigen Kryptowährungen stehen dafür längst Werkzeuge zur Verfügung, meistens in JavaScript implementiert. Unzählige Artikel und Tutorials erklären, wie beispielsweise ein Bitcoin Paper-Wallet erstellt werden kann. Immer wieder gibt es hier aber auch Angebote von Betrügern, die die Schlüsselerzeugung manipuliert haben. Sicherer ist dagegen, die Schlüsselgenerierung offline durchzuführen. Letztlich hat sich dafür die Generierung in einem schmalen Linux-System bewährt, welches offline und einmalig gestartet wird. Hier kommen meist Distributionen, wie Tails oder Ubuntu Live-USB zum Einsatz. Diese sind aber in keiner Weise auf eine so spezielle Aufgabe optimiert. Es ist davon auszugehen, dass ein Großteil der Endbenutzer eher auf die zahlreich verfügbaren Online-Angebote zur Aufbewahrung zurückgreift. Im Umkehrschluss kann die Offline-Speicherung attraktiver werden, wenn der Einrichtungsprozess einfacher ist. Daraus erschließt sich die Motivation dieses Projekts. Ziel soll es sein, eine Lösung zu entwickeln, die einen sicheren und einfachen Erstellungsprozess von Paper-Wallets ermöglicht. Verfolgt ein unerfahrener Anwender den grundsätzlich richtigen Gedanken, sein Guthaben offline zu speichern, soll ihm ein entsprechend optimiertes Werkzeug an die Hand gegeben werden können.

2. Risikofaktoren und Angriffsmöglichkeiten

Zunächst besteht das Risiko, keine seriöse Software zu verwenden. Vor allem unerfahrene Anwender sind davon betroffen. Bei Softwareprodukten, die etwas mit Kryptowährungen zu tun haben, taucht immer wieder auch Spyware auf. So beispielsweise auch bei Wallet-Anwendungen für das Smartphone [1]. Außerdem werden Phishing-Seiten verwendet, um Nutzer dazu zu bringen, sich eine Adresse dort generieren zu lassen. Die Betreiber der Phishing-Seiten haben verschiedene Methoden, um es so aussehen zu lassen, als seien die Private Keys tatsächlich zufällig generiert. Oft verfahren die Betrüger so, dass sie erst eine gewisse Zeit abwarten, bis genug Nutzer Opfer des Phishing-Angriffs wurden, um dann das Guthaben aller Wallets auf eigene Adressen zu transferieren [2]. Für so einen Angriff ist es noch nicht einmal erforderlich, dass der Rechner des Anwenders mit dem Internet verbunden ist. Es reicht aus, dass ein manipulierter Zufallsgenerator im verwendeten Programm zum Einsatz kommt. Wenn dieser keine echte Entropie liefert, sondern vorbestimmte Werte, kann die Anzahl der möglichen Adressen stark begrenzt werden. Der Angreifer kann die Erzeugung dann reproduzieren und alle privaten Schlüssel berechnen. Wenn ein Programm beispielsweise nur etwa 10000 verschiedene Adressen generiert, ist das entsprechend einfach möglich. Als Referenz hierfür dient ein Fall, durch den Betrüger etwa vier Millionen US-Dollar stehlen konnten, indem sie ein manipuliertes Tool zur Generierung von IOTA-Seeds anboten [2]. Obwohl die JavaScript-Anwendung auf GitHub veröffentlicht wurde, blieb die Manipulation unentdeckt.

Ein weiteres Risiko geht von Spyware im allgemeineren Sinne aus. Speziell Rechner, die bereits mit Malware infiziert sind und für die Generierung von Kryptoadressen verwendet werden, stellen ein sehr hohes Risiko dar. Die Chancen sind dann recht hoch, dass dies ebenfalls zu einer kompromittierten Adresse führt. Moderne Spyware, die permanent den Hauptspeicher des Opfers durchsucht, verfügt häufig auch über einen Payload zur Ermittlung möglicher Kryptoadressen. Ein Payload ist der Teil der Schadsoftware, der den tatsächlichen Schaden anrichtet. Im Falle der Spyware, stellt die Beschaffung der Informationen den Schaden dar. Ein Beispiel dafür ist Clipboard-Hijacker-Malware, die den Zwischenspeicher überwacht und bei einer entdeckten Bitcoin-Adresse,

diese mit einer eigenen austauscht [3]. Überprüft das Opfer eine eingefügte Adresse nicht noch einmal, gehen die Transaktionen auf ein Wallet des Betrügers.

Die erstellten Paper-Wallets sollen in der Regel ausgedruckt werden und hier entsteht mit der Benutzung des Druckers ein weiteres Risiko. Die Drucker können für sensitive Daten ein enormes Risiko darstellen, da die heutigen Multifunktionsdrucker meist alle über einen Netzwerk-Stack verfügen. In dem Artikel *"System & Application Security"* der Information Security and Policy von Berkeley wird über diese Problematik wie folgt berichtet.

"*Multifunction printers (MFPs) are experiencing an identity crisis: IT administrators don't always see them as the full-fledged networked computers they really are. But attackers do - and they are finding them increasingly very attractive!*" [4]

Das verwendete Programm zur Schlüssel-generierung kann dann noch so sicher sein und ordnungsgemäß funktionieren, wenn der ganze Prozess in einer unsicheren Umgebung ausgeführt wird. So, wie es in der IT-Sicherheit häufig formuliert wird, kann sich der Angreifer das Ziel aussuchen. Um an die Private Keys zu kommen, gibt es für Kriminelle demnach viele Wege. Die Umsetzung einer Lösung, die all diese Angriffsvektoren berücksichtigen soll, stellt daher keine triviale Aufgabe dar. Das Risiko, welches von klassischer Spyware ausgeht, kann mit einem Live-System sehr effizient minimiert werden. Von unsicheren Druckern hingegen geht auch in einem isolierten Betriebssystem weiterhin eine Gefahr aus.

3. Aufbau des Systems

Ziele der VINKTAR Live-Distribution:

- minimaler Aufbau
- offline, no root
- open source, auditierbar
- kryptografisch sicheres RNG-Konzept
- intuitives frontend
- hohe Druckerkompatibilität

Für die Erstellung der Distribution wird das Debian Live-Build Framework verwendet [5]. Das Debian Project hat bereits 2010 angekündigt, den Debian Kernel komplett ohne proprietäre Firmware zu entwickeln [6]. Die Basis der Distribution bildet daher ein 32-Bit Debian GNU/Linux der Release-Schiene "testing", da das "stable" Release in der Regel sehr alte Versionen von Anwendungen bezieht. Die Konfiguration der Distribution erfolgt bei Live-Build durch eine umfangreiche Verzeichnisstruktur.

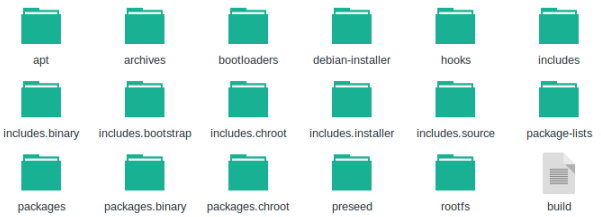

Bild 1: Konfigurationsverzeichnis von live-build, Bildquelle: L. Johns

Aus dieser Struktur kann das komplette System gebaut werden. Als Bezugsquelle der Softwarepakete sind die Debian-Repositorys voreingestellt. Diese werden auch immer für den Debian Kernel genutzt. Debian verwendet das Advanced Packaging Tool (APT) zur Paketverwaltung. Es können aber beliebige APT-Repositorys verwendet werden. In diesem Fall ist das nicht notwendig, da für das Grundsystem nur Debian Pakete verwendet werden sollen und die zusätzlichen Schlüsselgenerierungstools manuell über die live-build-Includes in das System integriert werden. Die Werkzeuge können in der isolierten Umgebung sowieso nicht aktualisiert werden. Nur einmal beim Erstellen des Abbilds werden alle Pakete von ihrer Bezugsquelle geladen. Verwendete Pakete des Live-Systems werden bei live-build in sogenannten Package-Lists organisiert. Das sind einfache Textdateien, die im config-Verzeichnis gespeichert werden. Sie beinhalten alle Bezeichnungen der Pakete, die aus den konfigurierten Repositorys von APT installiert werden sollen. Für dieses Projekt werden vier solcher Listen erstellt. Die Listen sind nach den entsprechenden Bereichen benannt. Diese Trennung ist erforderlich, um größere Änderungen am System zu erleichtern. Soll beispielsweise eine andere Desktopumgebung verwendet werden, ändern sich damit auch viele andere Pakete, die Bestandteil des Desktops sind. Dann muss einfach die *desktop.list.chroot* ausgetauscht werden. Das System beinhaltet einen Browser, Officeprogramme und printer-realted packages, die eine hohe Kompatibilität mit gängigen Druckermodellen ermöglichen sollen.

Package list	Beschreibung	Beispiel
apps.list.chroot	allgemeine Anwendungen	Firefox ESR
desktop.list.chroot	xfce4-spezifischen Pakete	xfce4-terminal
fonts.list.chroot	notwendige Schriftarten	fonts-liberation
live.list.chroot	für den live-Betrieb notwendige Pakete	user-setup
printing.list.chroot	zum Drucken erforderliche Pakete	system-config-printer
rng.list.chroot	Pakete für random number generator	rng-tools

Tabelle 1: Struktur der Paketlisten mit Beispielen

Weiterhin können benutzerdefinierte Skripte zu bestimmten Zeitpunkten ausgeführt werden (hook-scripts). Durch sogenannte *includes* werden eigene Dateien in die Live-Distribution integriert. Dadurch können Konfigurationsdateien für die im System enthaltene Software verfügbar gemacht werden. Auch die Tools zur Schlüsselgenerierung sind manuell in das Dateisystem integriert.

Eine essenzielle Softwarekomponente der Distribution stellt der Browser dar. Viele Werkzeuge zur Schlüsselerzeugung sind in JavaScript implementiert und werden im Browser ausgeführt. Speziell die kryptografischen Funktionen der Browser-API werden im Hinblick auf die spätere Verwendung im System wichtig. Für dieses Projekt wird Mozilla Firefox ESR verwendet. Dieser ist vollständig quell-offen und im Debian main repository verfügbar. Generell wird für das Live-System Wert auf ein minimalistisches Design gelegt. Sowohl die enthaltene Software als auch die Bedienung sind auf den schmalen Anwendungsfall fokussiert. Weitere Merkmale sind das Init-System systemd, der Bootloader SYSLINUX und der verwendete Desktop Xfce4. Die Xfce4-Oberfläche wird von Grund auf neu aufgebaut. Im Menü 'Paper Wallets' wird eine kategorische Zuordnung verwendet. Neue Anwendungen, die in das System integriert werden sollen, müssen so nur der Kategorie *PaperWallets* angehören und werden automatisch angezeigt. Weiterhin wurde für das allgemeine Erscheinungsbild noch ein Theme integriert mit entsprechenden Icons. Die Distribution soll sich möglichst intuitiv bedienen lassen, weshalb nur Usecase-relevante UI-Elemente vorhanden sind. Beispielsweise muss das System niemals in den Ruhezustand versetzt werden. Der Benutzer wird sich auch niemals manuell abmelden müssen. Die Konfigurationen dafür werden in XML-Dateien gespeichert. Diese befinden sich im Home-Verzeichnis des Benutzers unter *config/xfce4/xfconf/xfce-perchannel-xml/*, wobei z.B. die modulare Struktur der Panels in der Datei *xfce4-panel.xml* beschrieben wird [7]. Eine Besonderheit des Systems ist, dass es standardmäßig mit deaktiviertem Root-Benutzerkonto startet. Der Linux-Kernel kann mit der Option *noroot* gestartet werden, um die Verwendung als Root zu verhindern. Der Gedanke ist hier, dass ein Root-Benutzerkonto, welches nicht verwendet werden kann, auch keinen Schaden anrichtet. Weiterhin ist die Distribution permanent vom Netzwerk getrennt. Auch hier mittels Kernelparameter realisiert, da die Trennung möglichst weit unten im System-Stack durchgeführt werden soll.

Bild 2: Xfce4-Desktop mit allen enthaltenen paper wallet tools, Bildquelle: L. Johns

4. Random number generator

Innerhalb des Live-Systems wird eine sichere Zufallsgenerierung benötigt. Aktuell nutzen fast alle enthaltenen Tools die Browserfunktionen. Es gibt aber auch andere Implementierungen, wie zum Beispiel ein Shellscript für die Generierung von IOTA-Seeds. Hier wird der Kernel-RNG verwendet und dieser bedarf weiterer Maßnahmen. Linux stellt als Entropiequelle zwei Gerätedateien zur Verfügung. Das sind */dev/random* und */dev/urandom*. Gefüllt wird dieser Entropiespeicher mit Rauschwerten aus der Umgebung des Kernels, z.B. von Gerätetreibern [8]. Da ein Systemstart ohne zusätzliche Nutzerinteraktion in der Regel stark vorhersehbar ist, wird die gesammelte Entropie normalerweise beim Herunterfahren des Systems auf der Festplatte gespeichert. Da ein Live-System aber jedes mal mit dem exakt gleichen Datenträgerabbild startet, ist genau dieser Schritt nicht möglich.

Die Problematik von Zufallszahlen in Live-Systemen ist selbst für diesen speziellen Fall nicht neu. Das Betriebssystem Tails hat ebenfalls den Anspruch, kryptografische Werkzeuge nutzen zu können. Die Entwickler von Tails versuchen das Seeding des Linux-Kernels mit zwei zusätzlichen Entropiequellen zu steigern [9]. Speziell handelt es sich dabei um zwei Daemons, die in das System integriert werden. Der erste ist rngd, welcher bei einem vorhandenen Hardware-RNG, diesen für das Seeding des Entropiepools nutzt, bis ein definierter Grenzwert

erreicht ist. Nicht jedes System verfügt über einen Hardware-RNG, sodass rngd nicht in jedem Fall hilft. Deshalb gibt es noch eine zweite Quelle, die mit dem haveged-Daemon realisiert wird. Dieser nutzt den HAVEGE-Algorithmus (Hardware Volatile Entropy Gathering und Expansion), der aus nicht vorhersagbaren Stati des Prozessors, Entropie sammelt [10]. Es handelt sich dabei nicht um einen Ersatz für einen Hardware-RNG. Der Entwickler ordnet den Algorithmus wie folgt ein.

“One could theoretically reproduce the sequence if he/she was able to reproduce all the past events on the machine. They are not pseudo-random either since there is no (short) seed which would allow an exact reproduction of the random sequence. The randomness results instead from an inability to control or predict with sufficient accuracy the events involved in the generation process.” - [11]

Laut der Beschreibung des offiziellen Debian-Pakets ist haveged vor allem für den Einsatz auf Server- und Headless-Systemen mit eingeschränkter Benutzerinteraktion gedacht [12]. Live-Systeme, wie Tails oder die hier entwickelte Distribution haben mit Serversystemen insofern gemein, dass zum Zeitpunkt der Verwendung von Zufallszahlen, wenig Nutzerinteraktionen stattgefunden haben. Der Einsatz von haveged in virtualisierten Umgebungen wird teilweise kontrovers diskutiert [13], der HAVEGE-Algorithmus ist hier unter Umständen keine sichere Entropiequelle. Die hier entwickelte Live-Distribution wird daher generell nicht für den Einsatz in virtualisierten Instanzen empfohlen.

5. Integrierte Software

Anfangs sind Werkzeuge für 13 Kryptowährungen eingeplant. Es besteht die Möglichkeit, dies jederzeit zu erweitern. Die Auswahl der Kryptowährungen soll möglichst viele Nutzer ansprechen. Daher werden einfach die Kryptowährungen mit dem derzeit höchsten Handelsvolumen ausgewählt [14]. In Tabelle 2 sind diese entsprechend aufgelistet.

Eine Ausnahme der ausgewählten Werkzeuge stellt die Software 'IOTA-Paper-Wallet' dar. Hier wird der Private Key nicht im Browser generiert. Das Tool erwartet die Eingabe des 81-stelligen IOTA-Seeds, der aus den Zeichen A-Z und 9 generiert wird \cite{src:IIB0}. Um die Benutzung in der Distribution zu vereinfachen, wird ein kurzes Shell-Skript für die Generierung sicherer Seeds geschrieben, welches automatisch mit dem Öffnen des Browsers, in einem zusätzlichen Terminalfenster, ausgeführt wird. Im Skript werden fünf Seeds mit folgendem Shell-Befehl generiert.

cat /dev/urandom | tr -dc A-Z9 | head -c\${1: -81}

6. Typischer Usecase

Im Folgenden wird ein typisches Anwendungsbeispiel konstruiert und schrittweise durchgeführt. Es sollen zwei Paper-Wallets erstellt und ausgedruckt werden, eins für Litecoin und eins für EOS. Dafür wird ein Abbild der Distribution und ein USB-Stick mit mindestens einem Gigabyte benötigt. Zunächst wird das Abbild auf den Stick geschrieben. Bei einem Windows-System ist dafür ein Tool wie Rufus \cite{src:RRR0} sinnvoll. Unter Linux kann das Shellprogramm \texttt{dd} genutzt werden. In diesem Beispiel wird Linux eingesetzt und der USB-Stick befindet sich hier unter \texttt{/dev/sdd}, sodass sich der folgende Befehl ergibt.

sudo dd if="vinktar-live-image-i386.hybrid.iso" of="/dev/sdd"

Jetzt kann der Stick an den Rechner angesteckt werden, an dem das Live-System verwendet werden soll. Als Nächstes wird das Netzwerkkabel entfernt. Sind manuelle Schalter für WLAN oder Bluetooth vorhanden, können diese zusätzlich ausgeschaltet werden. Außerdem wird der Drucker per USB mit dem Rechner verbunden. Der Computer wird nun gestartet und soll das Live-System booten. Geschieht dies nicht automatisch, muss in der Regel eine bestimmte Taste zur manuellen Auswahl des Bootmediums während des Startvorgangs gedrückt werden. Es wird nun der BIOS-Start gewählt, der dann zum Menü des Bootloaders führt. Hier wird mit der Eingabetaste das System regulär gestartet. Nach nur wenigen Sekunden erscheint der Desktop des Live-Systems. Mit einem Klick auf das Menü *Paper Wallets* werden alle Kryptowährungen aufgelistet, für die das System Paper-Wallets erstellen kann. In diesem Fall wird zunächst EOS ausgewählt. Der Browser öffnet sich und zeigt die generierten Schlüssel inklusive QR-Codes an. Mit einem Klick auf *Print* oder mit der Eingabe von *Strg + P* kann das Paper-Wallet gedruckt werden. Ist der Drucker hier nicht gelistet, lässt er sich in der Regel über die Druckereinstellungen mit *Add* konfigurieren. Anschließend erscheint er in der Druckerauswahl. Bei bestimmten Anforderungen an das Layout des Paper-Wallets, können alternativ auch die im System enthaltenen Office Programme genutzt werden. Beispielsweise können so auch mehrere Schlüssel auf eine Seite gedruckt werden. Ist der Vorgang abgeschlossen, wird über das Menü *Paper Wallets* der Punkt *Litecoin (LTC)* gewählt. Da *liteaddress* noch die Mausbewegung als Entropiequelle hinzuzieht, erscheinen die Schlüssel nicht sofort. Ansonsten kann hier analog verfahren werden. Abschließend wird das System heruntergefahren. Als weitere Maßnahmen sind noch die Trennung vom Stromnetz für einige Sekunden und das Zurücksetzen des Druckers auf Werkseinstellungen zu empfehlen.

Kryptowährung	Software	Umsetzung	RNG
Bitcoin	bitaddress.org	JavaScript	crypto.getRandomValues, user generated entropy
Ethereum	myetherwallet	JavaScript	crypto.getRandomValues
Ripple XRP	ripply.eu	JavaScript	crypto.getRandomValues
Bitcoin Cash	bitcoin.com	JavaScript	crypto.getRandomValues, user generated entropy
EOS	eoscafe paper-wallet	JavaScript	crypto.getRandomValues
Stellar	stellar-paper-wallet	JavaScript	crypto.getRandomValues
Litecoin	liteaddress.org	JavaScript	crypto.getRandomValues, user generated entropy
Monero	monero-walletgenerator	JavaScript	crypto.getRandomValues, user generated entropy
Tron	tronpaperwallet.org	JavaScript	crypto.getRandomValues
IOTA	IOTA-Paper-Wallet	JavaScript	nicht vorhanden
Dash	paper.dash.org	JavaScript	crypto.getRandomValues, user generated entropy
NEO	Ansy	JavaScript	crypto.getRandomValues
Ethereum Classic	myetherwallet	JavaScript	crypto.getRandomValues

Tabelle 2: Integrierte Software zur kryptografischen Schlüsselerzeugung

7. Ausblick

Letztlich ist die Entwicklung einer Linux-Distribution ein sehr umfangreicher Prozess und wird für eine einzelne Person überhaupt erst möglich, durch Projekte wie Live-Build von Debian. Bei der Entwicklung bekannter Distributionen sind daher oft große Teams in diesen Prozess involviert. In Zukunft könnte das Live-System dahingehend ausgebaut werden, dass es nicht mehr nur auf das Erstellen von Paper-Wallets beschränkt ist. Denkbar ist hier eine Live-Distribution für generelle Aufgaben im Bereich Kryptowährungen, die in einer isolierten Umgebung sinnvoll sind.

Literaturverzeichnis

[1] Lookout. blog.lookout.com. 3 fake Bitcoin wallet apps appear in (and are quickly removed from) Google Play Store. [Online] 2017. https://blog.lookout.com/fake-bitcoin-wallet, abgerufen am 07.11.2018

[2] Mirko Ross. heise.de. Kryptowährung: IOTA im Wert von vier Millionen US-Dollar geklaut. [Online] 2018. https://www.heise.de/ix/meldung/Kryptowaehrung-IOTA-im-Wert-von-vier-Millionen-US-Dollar-geklaut-3952723.html, abgerufen am 05.09.2018

[3] Lawrence Abrams. bleepingcomputer.com. Clipboard Hijacker Malware Monitors 2.3 Million Bitcoin Addresses. [Online] 2018. https://www.bleepingcomputer.com/news/security/clipboard-hijacker-malware-monitors-23-million-bitcoin-addresses/, abgerufen am 12.11.2018

[4] Open Berkeley. security.berkeley.edu. Network Printer Security Best Practices | Berkeley Information Security and Policy. [Online] 2018. https://security.berkeley.edu/resources/best-practices-how-articles/system-application-security/network-printer-security-best, abgerufen am 08.11.2018

[5] Debian Live Team. debian developers' corner. Debian Live Project. [Online] 2018. https://www.debian.org/devel/debian-live/index.en.html, abgerufen am 10.10.2018

[6] Debian Projekt. Debian Nachrichten. Debian 6.0 Squeeze wird mit vollständig freiem Linux-Kernel veröffentlicht. [Online] 2010. https://www.debian.org/News/2010/20101215, abgerufen am 27.09.2018

[7] TiberiusT. forum.xfce.org. Panel config files are where?. [Online] 2012. https://forum.xfce.org/viewtopic.php?id=7671, abgerufen am 03.11.2018

[8] Michael Kerrisk Ubuntu Manpage Repository. Ubuntu Manpage: random, urandom - Kernel-Geräte zur Erzeugung von Zufallszahlen. [Online] 2018. http://manpages.ubuntu.com/manpages/precise/de/man4/random.4.html, verfügbar am 01.09.2018, 11:35.

[9] Tails project. tails.boum.org. Tails - Random numbers. [Online] 2018. https://tails.boum.org/contribute/design/random/, verfügbar am 15.11.2018

[10] Olivier Rochecouste. irisa.fr. HAVEGE Hardware Volatile Entropy Gathering and Expansion, Overview. [Online] 2006. http://www.irisa.fr/caps/projects/hipsor/, verfügbar am 15.11.2018

[11] Olivier Rochecouste. irisa.fr. Execution time of a short sequence of instructions and hardware volatile states in a modern microprocessor. [Online] 2006. http://www.irisa.fr/caps/projects/hipsor/misc.php #measure, verfügbar am 15.11.2018

[12] Debian Team. packages.debian.org. Debian -- Informationen über Paket haveged in buster. [Online] 2018. https://packages.debian.org/buster/haveged, verfügbar am 15.11.2018

[13] Nic Waller. security.stackexchange.com. Is it appropriate to use haveged as a source of entropy on virtual machines?. [Online] 2013. https://security.stackexchange.com/questions/34523/is-it-appropriate-to-use-haveged-as-a-source-of-entropy-on-virtual-machines, verfügbar am 16.11.2018

[14] CoinMarketCap. coinmarketcap.com. Cryptocurrencies Market Capitalization. [Online] 2018. https://coinmarketcap.com/, verfügbar am 19.10.2018

[15] Rufus USB. rufususb.com. Rufus - bootable USB flash drive. [Online] 2018. http://rufususb.com/, verfügbar am 13.11.2018