Angriff auf verschlüsselte Linux System-Partitionen

Tobias Köhler[1] · Alexander Dörsam[2]

[1]Hochschule Darmstadt
Fachbereich Informatik

[2]Antago GmbH
sicherheit@antago.info

Zusammenfassung

Zunehmend spezialisieren sich Angreifer auf das zielgerichtete Beschaffen von kritischen Informationen. Unternehmen speichern diese Informationen, geschützt vor Zugriffen aus dem Internet heraus, zentral auf ihren Systemen. Um auch vor lokalen Attacken sicher zu sein, schützen sich Unternehmen unter anderem durch die Verschlüsselung der Datenträger. Da diese Datenträger somit zwangsläufig Ziel der Angreifer werden, setzt sich diese Arbeit mit der Verwundbarkeit verschlüsselter Betriebssysteme auseinander. Basierend auf der Forschungsarbeit von Tobias Köhler aus dem Jahre 2014 im Rahmen des Bachelor-Studiengangs Informatik an der Hochschule Darmstadt, beschreibt diese Ausarbeitung einen erweiterten Angriff auf verschlüsselte Linux System-Partitionen. Die Entwicklung einer neuen Anwendung ermöglicht es einem Angreifer, durch kurzzeitigen physischen Zugriff auf ein System, verschiedene Angriffe, unabhängig der verwendeten Verschlüsselungsalgorithmen oder der Stärke des Benutzerpasswortes, auf das verschlüsselte Zielsystem durchzuführen. Aktuell gängige Schutzmechanismen, beispielsweise die zusätzliche Verschlüsselung der Boot-Partition und die Entsperrung durch den Bootloader, verlagern das Problem lediglich auf eine andere Ebene. Dies hat zur Folge, dass aktuell keine angemessenen, praxistauglichen Mechanismen zur Absicherung existieren, sodass über den entwickelten Angriff auch technisch nicht-versierte Personen zielgerichtete Angriffe auf verschlüsselte Linux-Systeme, beispielsweise zur Beweisfälschung oder zum Passwortdiebstahl, ausführen können.

1 Grundlagen

1.1 Festplattenverschlüsselung

Ziel einer Festplattenverschlüsselung ist, nicht nur die darauf gespeicherten Informationen, sondern auch die während der Verarbeitung, beispielsweise in temporären Verzeichnissen, anfallenden Daten zu schützen. Damit dieses Ziel effizient erreicht werden kann, sollten nicht nur einzelne Dateien, sondern ganze Partitionen und, falls möglich, die gesamte Festplatte verschlüsselt werden. Da auf Festplatten große Datenmengen gespeichert und verarbeitet werden, kommen hier nur symmetrische Verfahren zum Einsatz. Zur Nutzung der verschlüsselten Daten muss der Benutzer autorisiert werden. Dies geschieht in der Regel mittels einer Passwortabfrage zum Start des Systems (Abbildung 1).

Aus technischer Sicht als problematisch gilt, dass einige Komponenten, die zur Entschlüsselung der Festplatte notwendig sind, fast ausschließlich in einem ungeschützten Bereich liegen, der

P. Schartner (Hrsg.) · D•A•CH Security 2016 · syssec (2016) 54-62.

Boot-Partition (Abbildung 2). Dadurch ergeben sich neue Angriffsmöglichkeiten. Eine Ausnahme ist für den Startvorgang in Hardware ausgelagerte Software, die nicht veränderbar und so nur sehr schwer zu manipulieren ist. Dies stellt in der Realität jedoch die Ausnahme dar.

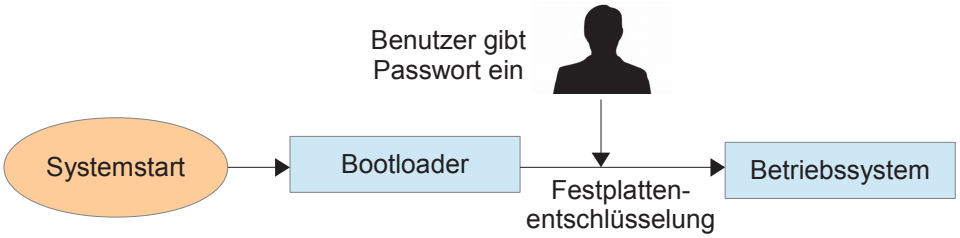

Abb. 1: Prinzip einer Festplattenverschlüsselung

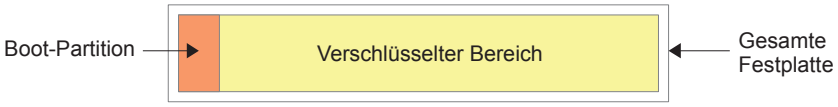

Abb. 2: Aufteilung des Datenträgers

In den folgenden Abschnitten wird auf die beschriebene Problematik detaillierter eingegangen. Gängige Werkzeuge zur Verschlüsselung von System Partitionen sind u.a.

- **Windows:** BitLocker oder VeraCrypt (Ableger von TrueCrypt)
- **Linux:** cryptsetup/LUKS
- **Mac OS X:** FileVault

1.2 Bestehende Angriffe

Im Zuge der Recherche ergab sich, dass nur wenige Angriffe auf verschlüsselte Linux System-Partitionen existieren. Die meisten dieser Angriffe basieren auf einer Analyse der eingesetzten Verschlüsselungsalgorithmen und können daher nur in einem speziellen Kontext oder unter Laborbedingungen angewandt werden. In der Realität hat ein Angreifer jedoch häufig ein begrenztes Zeitfenster zur Verfügung. Ein Brute-Force-Angriff wird daher mit großer Wahrscheinlichkeit scheitern (offline-Angriffe ausgenommen). Windows-basierte Betriebssysteme, die eine Systemverschlüsselung über Truecrypt realisieren, können über das Angriffswerkzeug EvilMaid attackiert werden. Bei EvilMaid handelt es sich um ein Angriffs-Werkzeug, welches auf ein spezielles Szenario abgestimmt ist. Bei dem durch EvilMaid realisierten Angriff benötigt der Angreifer zweimaligen physischen Zugriff auf das Zielsystem, um an das Verschlüsselungspasswort zu gelangen. Ein Angriff ähnlich der Vorgehensweise von EvilMaid existierte zum Zeitpunkt der Ausarbeitung im Jahre 2104 im Linux Umfeld nicht. Mittlerweile existieren jedoch Werkzeuge von Dritten, die den Angriff auf Basis der damaligen Forschungsarbeit kopiert haben [McNa15].

2 Ziele

Diese Forschungsarbeit hat das Ziel, den geheimen Schlüssel zur Entsperrung verschlüsselter Linux System-Partitionen durch kurzzeitigen physischen Zugriff auf ein System abzugreifen. Dabei soll das Passwort nicht aufwändig über einen Brute-Force-Angriff erraten, sondern über

einen Seitenkanal extrahiert werden. Damit der Angriff auch einem wenig versierten Angreifer möglich ist, muss der Angriff ohne Interaktion durchführbar sein. Weiter soll der Erfolg des Angriffs unabhängig von der eingesetzten Verschlüsselungsagorithmen oder der Stärke des Passwortes sein. Abschließend sollen im Detail mögliche Schutzmechanismen evaluiert und darauf basierend ggf. weitere Angriffsszenarien diskutiert werden.

3 Rahmenbedingugen und Vorgehensweise

Als zu untersuchendes System wird das weit verbreitete Server Betriebssystem *Debian GNU/Linux wheezy* gewählt. Auch für Linux-basierte Betriebssysteme existiert VeraCrypt zur Verschlüsselung von Partitionen. Jedoch unterstützt VeraCrypt in dieser Umgebung die Verschlüsselung der Systempartition nicht. Für diesen Zweck kommt die Kombination aus der Anwendung *cryptsetup* und dem Verfahren *LUKS* zum Einsatz. Der hohe Grad an Flexibilität dieser Kombination ermöglicht eine Vielzahl von Anwendungsmöglichkeiten, beispielsweise verschlüsselte Systempartitionen. *cryptsetup/LUKS* hat sich weitgehend als Standard für verschlüsselte Systempartitionen unter Linux-basierten Betriebssystemen etabliert. Der Angriff wird aus Gründen der Flexibilität in einer virtuellen Maschine durchgeführt. Gegenüber einer physischen Maschine existieren zwar Unterschiede im Startverhalten, jedoch beeinflussen diese die Ergebnisse im Kontext der Ausarbeitung nicht.

4 Analyse und Angriff

4.1 Boot-Vorgang

Nach der Definition des technischen Rahmens soll eine Analyse des Startvorgangs zeigen, ob und ggf. wo eine Schwachstelle für einen Angriff existieren kann. Dazu müssen die Komponenten, die bis zur Entschlüsselung der Festplatte am Startvorgang beteiligt sind, untersucht werden. Abbildung 3 zeigt, welche Komponenten ein Linux-basiertes Betriebssystem durchlaufen muss, um zu starten. Hierbei ist explizit zu erwähnen, dass alle Komponenten in einem unverschlüsselten Bereich der Festplatte gespeichert sind. Nachdem die Hardware des Systems durch das BIOS initialisiert wurde, startet dieses den Eintrag des Bootloader (Stage 1) aus dem MBR. Der Bootloader (Stage 2) startet danach von der unverschlüsselten Boot-Partition und zeigt eine Liste der verfügbaren Betriebssysteme. Wurde ein Betriebssystem vom Benutzer ausgewählt, lädt der Bootloader den Kernel in den Speicher und übergibt die Kontrolle. Der Kernel führt essentielle Initialisierungen durch und übergibt abschließend den weiteren Startvorgang an das *init-Script* des Betriebssystems, welches beispielsweise für die Verwaltung von RAID-Verbunden zuständig ist, oder die Eingabe des Entschlüsselungspasswortes des Benutzer entgegen nimmt.

Nachdem in der Analysephase die Komponenten BIOS, Bootloader und Kernel überprüft und dabei keine einfach auszunutzenden Schwachstellen identifiziert werden konnten, sollte in der RamDisk bzw. im darin enthaltenen Initialisierungs-System (*init-Script*) eine solche Verwundbarkeit entdeckt werden.

4.2 RamDisk

Unter Linux bezeichnet eine Ramdisk, auch bekannt als *initrd* oder *initramfs*, ein zumeist komprimiertes Dateiarchiv, das die Funktionalität für den Startvorgang enthält. Nachdem der Kernel gestartet wurde, kann dieser Aufgaben, die zum Start des Betriebssystems notwendig sind, bei-

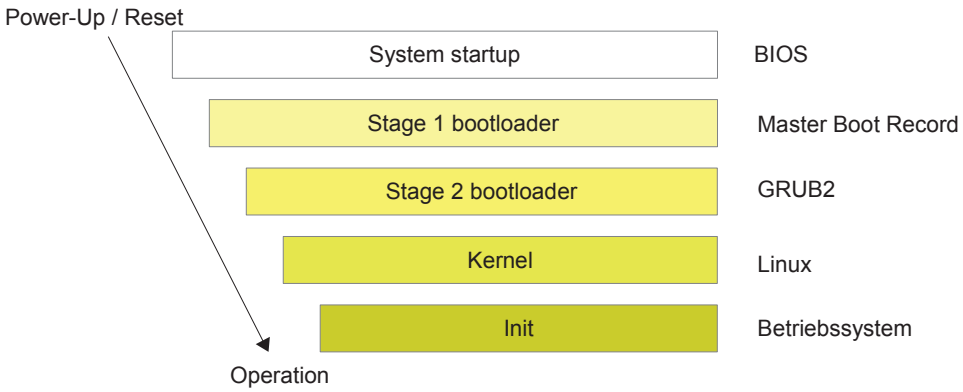

Abb. 3: Linux Boot-Prozess

spielsweise das Starten eines RAID-Verbundes, der Ramdisk übergeben. Der Kernel entpackt das Dateiarchiv im Speicher und startet das Init-Script der Ramdisk. Dieses Script enthält jegliche zum Systemstart notwendige Logik, beispielsweise auch Hilfsmittel zur Entschlüsselung von Partitionen. Damit der Kernel entlastet wird und flexibel bleibt, hat sich die Ramdisk als unterstützende Komponente im Startvorgang etabliert.

Wird auf einem Debian-basierten Betriebssystem ein Kernel installiert, wird anschließend durch das Werkzeug *initramfs-tools* eine Ramdisk inklusive der für den Startvorgang benötigen Scripte erstellt. Zur Untersuchung wird nun die Ramdisk auf der unverschlüsselten Boot-Partition des anzugreifenden Systems entpackt. Das zur Entschlüsselung verwendete Script befindet sich im Verzeichnis */scripts/local-top/cryptroot*.

```
281 (...)
282 if [ ! -e "$NEWROOT" ]; then
283  if ! crypttarget="$crypttarget" cryptsource="$cryptsource" \
284    $cryptkeyscript "$cryptkey" | $cryptcreate --key-file=- ; then
285    message "cryptsetup: cryptsetup failed, bad password or options?"
286    continue
287  fi
288 fi
289 (...)
```

Der Auszug des Quellcodes zeigt die Stelle, an dem der Benutzer das Entschlüsselungs-Passwort eingibt, das Script die Eingabe entgegen nimmt und den Schlüssel in der Variablen *$cryptkey* speichert (Zeile 284). Hier entsteht eine Schwachstelle. Sollte ein Angreifer in der Lage sein, diesen Programmabschnitt, der sich in der Ramdisk befindet, zu verändern, kann die Eingabe des Passwortes umgeleitet bzw. im Klartext abgefangen werden. Da die Ramdisk zum Start des Systems allgemein notwendig ist, liegt diese unverschlüsselt auf der Boot-Partition und ist zum Zeitpunkt des Startvorgangs nicht vor Manipulationen geschützt. Die Manipulation der Ramdisk soll anschließend über ein externes Medium, beispielsweise einen USB-Stick, erfolgen. Während der Entschlüsselung der Festplatte durch den Benutzer soll dass Passwort an eine unverschlüsselte Stelle der Festplatte gespeichert werden. Dazu ist es notwendig, die Eingabe des Passwortes abzufangen und umzuleiten. Die Manipulation soll über kurzzeitigen physischen Zugriff auf das System vollautomatisch und ohne jegliche Interaktion des Angreifers geschehen (Abbildung 4).

Existierende Live-CDs würden zur Umsetzung in Frage kommen, jedoch wäre zum Ausnutzen

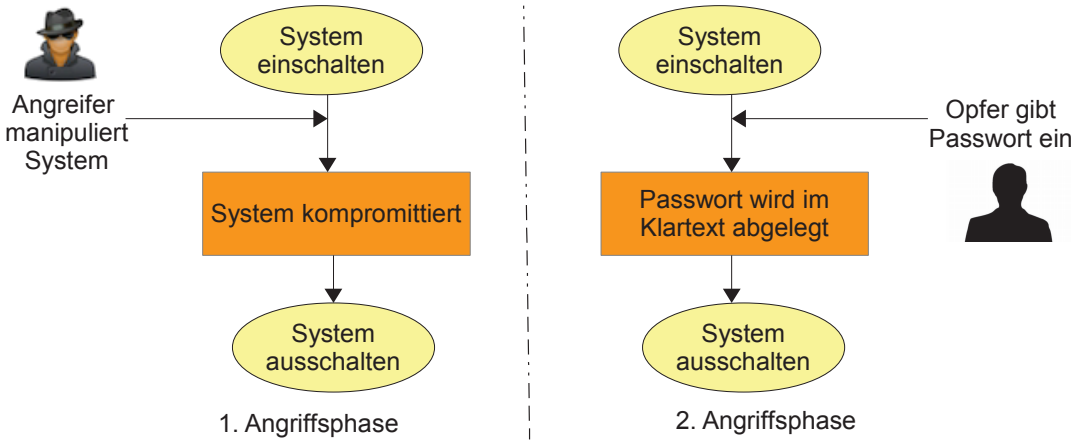

Abb. 4: Linux Boot-Prozess

der Schwachstelle ein hoher Grad an manuellen Eingriffen durch den Angreifer erforderlich. Somit wurde eine eigene Implementierung angestrebt und realisiert. Aus Gründen der Brisanz des Angriffs wurde die Implementierung bisher nicht veröffentlicht.

5 Gegenmaßnahmen

Um sich vor dem Angriff zu schützen, müssen geeignete Maßnahmen ergriffen werden, um die Manipulation der Ramdisk zu erkennen oder, falls möglich, im Voraus zu verhindern. Zu diesem Zweck existieren mehr oder weniger verbreitete Werkzeuge und Methoden, die einem Benutzer unterstützend dienen können.

5.1 Integritätsprüfungen via Prüfsummen

Prüfsummen werden verwendet, um die Integrität von Dateien, beispielsweise der Ramdisk, zu sichern. Jedoch sind Prüfsummen an dieser Stelle nicht angriffsverhindernd, sondern dienen lediglich informierenden Zwecken. Weiter muss sichergestellt sein, dass die Datei, welche die korrekten Prüfsummen beinhaltet, nicht in einem unverschlüsselten Bereich des Datenträgers gespeichert ist, da ein Angreifer auch diese Datei manipulieren könnte. Ein Werkzeug zur Integritätsprüfung der auf der Boot-Partition befindlichen Dateien stellt beispielsweise *chkboot* [Schm12] dar.

5.2 Auslagerung in Hardware (TPM)

Auch in Hardware ausgelagerte Sicherheitsfunktionen können als Gegenmaßnahme dienen. Ein TPM-Chip ist eine Hardware-Erweiterung, die ein Computersystem mit zusätzlichen Sicherheitsfunktionen ausstattet. Das Modul kann beispielsweise verwendet werden, um den Schlüssel zur Entsperrung einer verschlüsselten Festplatte in diesen Chip auszulagern und somit vor Manipulation zu schützen. Auch Prüfsummen zur Sicherung der Integrität der am Startvorgang beteiligten Software-Komponenten können im TPM-Chip ausgelagert werden. Alle diese Komponenten müssen TPM unterstützen, um das sichere Starten zu garantieren. Dazu ist im Linux Umfeld unter anderem ein spezieller Bootloader (trustedgrub) und eine spezielle Ramdisk (dracut) notwendig. Die beiden Anwendungen schränken jedoch erheblich die Flexibilität des Systems ein. Weiter ist TPM unter Linux nur mit gehobenem Aufwand zu konfigurieren. Die Konfiguration ist dabei so statisch, dass bei geringfügigen Änderungen am Startvorgang viele

Komponenten neu initialisiert werden müssen. Eine Migration auf den TPM-basierten Startvorgang wird daher meist als zu aufwändig eingestuft.

5.3 Bootloader mit Kryptographie Unterstützung

Die Schwachstelle der unverschlüsselten Boot-Partition existiert bei einer Festplattenverschlüsselung mit Truecrypt unter Windows nicht, da bereits der Bootloader die Festplatte entschlüsselt. Es ist demnach erwünscht, auch unter Linux-basierten Betriebssystemen die Boot-Partition verschlüsseln zu können. Der Bootloader GRUB2 unterstützt seit einiger Zeit das Entsperren verschlüsselter Systempartitionen. Ein großer Nachteil ist jedoch, dass der Bootloader das vom Benutzer eingegebene Passwort nicht an die Ramdisk weiter reichen kann, damit diese ebenfalls die Systempartition entsperren kann. Es ist folglich notwendig, das Passwort innerhalb eines kurzen Zeitraums mehrfach einzugeben. Zudem ist das Wissen über diese Funktionalität des Bootloaders GRUB2 nur mäßig verbreitet. Weiter gilt es an dieser Stelle zu evaluieren, ob die Entschlüsselung der Festplatte durch den Bootloader die entdeckte Verwundbarkeit der Ramdisk nicht nur in eine andere Komponente verlagert.

6 Fazit

Als Fazit lässt sich zusammenfassen, dass es möglich ist, mit cryptsetup/LUKS verschlüsselte Systempartitionen zu kompromittieren. Dieser Angriff ist unabhängig von der gewählten Verschlüsselungsmethode und der Komplexität des verwendeten Passwortes. Durchgeführt werden kann der Angriff mit der im Rahmen dieser Arbeit entwickelten Anwendung. Diese ermöglicht es einem Angreifer bei physischem Zugriff eine Kopie des Passwortes zur Entschlüsselung in einem für ihn zugreifbaren Bereich der Festplatte abzulegen. Aus den gewonnenen Erkenntnissen kann abgeleitet werden, dass die gängigen Verfahren zur Verschlüsselung von Informationen auf Linux basierenden Betriebssystemen keinen absoluten Schutz vor physischen Angriffen bieten. Dies gilt sowohl im Kontext bereits existierender Angriffe auf eingeschaltete und entschlüsselte Systeme, als auch für den bisher als sicher eingestuften Zustand einer heruntergefahrenen und damit verschlüsselten Maschine.

Mechanismen zur Abwehr der gezeigten Konzepte basieren in der Regel auf spezieller Hardware (TPM) oder erfordern erweiterte Kenntnisse in der Konfiguration. Schlussfolgernd kann ein versierter Angreifer mit einer sehr hohen Erfolgsquote bei dem gezeigten Angriff rechnen. Da bei verschlüsselten Datenträgern von entsprechend sensitiven Informationen ausgegangen werden kann, werden sich Anwender und Spezialisten in Zukunft mit einer neuartigen Bedrohung konfrontiert sehen, welche nur schwer abzuwenden sein wird.

7 Ausblick

Ein Angreifer wird in der Realität nicht ausschließlich auf Debian basierende Betriebssysteme stoßen. Auch die Anzahl der physischen Zugriffe auf ein System kann stark begrenzt sein. Somit verbleiben nicht viele Gelegenheiten, einen fehlgeschlagenen Angriffsversuch zu wiederholen.

Die Implementierung der in dieser Forschungsarbeit beschriebenen Angriffsmethodik wirkt daher auf den ersten Blick statisch. Um die Qualität des Angriffs mittelfristig zu erhöhen, wird ein kurzer Ausblick auf einige potenzielle Aspekte der Weiterentwicklung gegeben, um die Erfolgschancen in der Realität zu steigern.

7.1 Nachweisbarkeit des Angriffs

Nach einem erfolgreichen Angriff existiert auf der Boot-Partition eine Datei mit dem Inhalt des Verschlüsselungspasswortes. Einerseits wird nur eine geringe Anzahl an Personen regelmäßig den Inhalt der Boot-Partition prüfen, zumal der Angriff bislang weitläufig unbekannt ist. Jedoch ist die Existenz der Datei leicht festzustellen und kann so die Kompromittierung des Systems mit geringem Aufwand bestätigen. Abhilfe schafft hier die Auslagerung des Passwortes in einen nicht vom Dateisystem lesbaren Bereich des Datenträgers, beispielsweise den MBR oder den LUKS-Header.

7.2 Dynamik des Angriffs

Die Implementierung des Angriffs gilt bislang nur für eine spezielle Version eines Betriebssystems. Das Prinzip des Angriffs gegenüber ähnlichen, Linux-basierenden Betriebssystemen bleibt jedoch weitgehend identisch. Die dynamischen Faktoren bei diesem Angriff sind unter anderem:

- Die Position der Bootpartition
- Die Anzahl der Ramdisks
- Der Name der Ramdisk

Mittelfristig gilt es diese Faktoren in den Angriff einzubeziehen. Das automatisierte Auffinden der Boot-Partition kann mit geringem Aufwand implementiert werden. In der Regel haben Boot-Partitionen das Boot-Flag gesetzt. Diese Markierung kann sehr einfach über die Partitionstabelle identifiziert werden. Sollte bei keiner Partition das Boot-Flag gesetzt sein, wird das Einhängen aller verfügbaren Partitionen und das anschließende Suchen von Dateien, welche auf die Boot-Partition hindeuten, ebenfalls zum Erfolg führen.

Existieren auf einem System mehrere Kernel Versionen und somit demnach auch verschiedene Ramdisks, muss der Angreifer bzw. das Angriffsscript in Erfahrung bringen, welche Kombination aktuell zum Starten verwendet wird. Hierfür kann der Zeitstempel der Ramdisk herangezogen werden. Sollten die Ramdisks einen identischen oder unrealistischen Zeitstempel aufweisen, werden alle entdeckten Ramdisks manipuliert. Dies erhöht zwar ggf. den Aufwand und somit die Zeit, die für den Angriff benötigt wird, jedoch steigen die Erfolgsaussichten, die richtige Ramdisk zu manipulieren, stark.

Distributionen benutzen verschiedene Namensgebungen für Kernel und Ramdisk. Der Angreifer könnte über eine erweiterte Funktion im Angriffsscript nach Schlagwörtern, beispielsweise *initrd*, *iniramfs* oder *ramdisk* suchen. Weiter kann hier auch eine Art *Datenbank der Ramdisk-Namen* verwendet werden. Um ständig die neuesten Distributionen zu unterstützen, müsste das Script regelmäßig aktualisiert werden. Die Anzahl der unterstützten Distributionen kann mit den beschriebenen Methoden drastisch erhöht werden.

7.3 Reduktion des Aufwands für den Angreifer

In der Realität kann es vorkommen, dass ein Angreifer lediglich einmaligen physischen Zugriff zu einem System erlangen kann. Eine erfolgreiche Infizierung des Systems wäre folglich möglich, jedoch kann der Angreifer den Schlüssel anschließend nicht auslesen. Die Lösung für dieses Szenario ist der Abfluss des Passwortes über einmaligen physischen Zugriff.

Um dieses Ziel zu erreichen, kann über eine Erweiterung des Quellcodes ein zusätzliches Stück

Programmcode in das Entschlüsselungs-Script der Opfer-Ramdisk injiziert werden. Möglich wäre unter anderem, das Senden des Passwortes über eine HTTP-POST Anfrage an eine bestimmte, zuvor definierte Website. Das benötigte Programmstück kann bei jedem Systemstart beispielsweise in die Datei */etc/rc.local* des Systems geschrieben werden. Diese Datei ist üblicherweise für Benutzeraktionen, die während des Systemstarts ausgeführt werden sollen, reserviert. Eine Manipulation dieser Datei würde folglich kaum auffallen. Prüft der Benutzer nicht zusätzlich regelmäßig den Netzwerkverkehr, ist der kritische Informationsfluss kaum zu bemerken. Über die beschriebene Methode kann der Angreifer den Aufwand und das Risiko des Angriffs um die Hälfte reduzieren.

7.4 Persistenz durch Manipulation im System

Der geschilderte Angriff ist in der ausgearbeiteten Form nur für ein begrenztes Zeitfenster persistent. Wird der Kernel des Betriebssystems vom Benutzer aktualisiert, greifen die Automatismen zur Aktualisierung der Ramdisk. Ist dies der Fall, wurden die durch den Angreifer vorgenommenen Änderungen revidiert. Abhilfe schafft hier eine Manipulation der im Betriebssystem vorhandenen *initramfs-tools*. Nach dem erfolgreichen Angriff wird zusätzlich über die bereits manipulierte Ramdisk bei Systemstart die lokale Datei */usr/share/initramfs-tools/scripts/local-top/cryptroot* mit dem bekannten Schadcode infiziert. Sobald der Kernel eine Aktualisierung erfährt und somit die Ramdisk neu generiert wird, wird der Schadcode der neu generierten Ramdisk hinzugefügt. Eine Aktualisierung der *initramfs-tools* kommt wiederum nur sehr selten vor. Der Zeitraum der Infizierung des Systems wird somit deutlich verlängert.

7.5 Umgehung von Integritätsprüfungen

Die in den Gegenmaßnahmen beschriebene Integritätsprüfung als Gegenmaßnahme des in dieser Arbeit beschriebenen Angriffs scheint auf den ersten Blick akzeptabel. Zu beachten ist jedoch, dass nach Entsperrung der Systempartition durch das *cryptroot-Script* die Daten des Betriebssystems kurzzeitig im Klartext vorliegen und zu diesem Zeitpunkt noch keinerlei Betriebssystem abhängige Werkzeuge in den Startvorgang eingreifen können. Es ist einem Angreifer in diesem Moment möglich, die Binärdateien bzw. die Funktionalität dieser Werkzeuge zu verändern oder außer Kraft zu setzen. Schutzmechanismen, die erst zur Laufzeit des Betriebssystems aktiv werden, sind als Maßnahme gegen den Angriff zu vernachlässigen.

Literatur

[BoCe01] D. P. Bovet, M. Cesati: Understanding the Linux Kernel. O Reilly and Associates, Inc., 1. Aufl. (2001).

[Chif09] P. Chifflier: Implementing the evil maid attack on Linux with Luks (2009). http://www.wzdftpd.net/blog/index.php?post/2009/10/28/44-implementing-the-evil-maid-attack-on-linux-with-luks

[Cryp09] Crypto-man: Hybrides Verschlüsselungsverfahren: Verschlüsselung eines Dokumente (2009).

[Dani13] E. T. Daniel Genkin, Adi Shamir: RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. http://www.tau.ac.il/ tromer/papers/acoustic-20131218.pdf (2013).

[Died09] O. Diedrich: Trendstudie Open Source. http://www.heise.de/open/artikel/Eingesetzte-Produkte-224518.html (2009).

[dist14] distributed.net: The Largest Computer on Earth. http://cowpie.distributed.net/rc5-proxyinfo.html (2014).

[Ecke04] C. Eckert: IT-Sicherheit – Konzepte, Verfahren, Protokolle. Oldenbourg Verlag München Wien, 3 Aufl. (2004).

[Etti06] R. Ettisberger: IT-Security and Hacking. Books on Demand GmbH, Norderstedt, 1 Aufl. (2006).

[Gelb13] M. Gelbmann: Debian/Ubuntu extend the dominance in the Linux web server market at the expense of Red Hat/CentOS. http://w3techs.com/blog/entry/debian_ubuntu_extend_the_dominance_in_the_linux_web_server_market_at_the_expense_of_red_hat_centos (2013).

[HoBu06] G. Hoglung, J. Butler: Rootkits. Addison-Wesley Verlag, 1 Aufl. (2006).

[Jone06] M. Jones: Inside the Linux boot process (2006). http://www.ibm.com/developerworks/library/l-linuxboot/index.html

[KrSi13] G. Kroah-Hartman, K. Sievers: Udev. http://sources.debian.net/src/udev/175-7.2/extra/udev.startup (2013).

[Lang11] P. Lange: Evil Maid (2011). https://www.os3.nl/2010-2011/students/pieter_lange/ot/evilmaid

[Lell13] J. Lell: Practical malleability attack against CBC-Encrypted LUKS partitions. http://www.jakoblell.com/blog/2013/12/22/practical-malleability-attack-against-cbc-encrypted-luks-partitions/ (2013).

[McNa15] R. McNamara: Introducing EvilAbigail. https://blog.gdssecurity.com/labs/2015/12/23/introducing-evilabigail.html (2015).

[Pete09] G. Pete: Substitutions-Permutations-Netzwerk(2009). http://de.wikipedia.org/wiki/Substitutions-Permutations-Netzwerk

[Proj13] cryptsetup/LUKS Projektseite: Frequently Asked Questions (2013). https://code.google.com/p/cryptsetup/wiki/FrequentlyAskedQuestions

[Royt13] M. Roytman: Data Fundamentalism. http://blog.risk.io/2013/04/data-fundamentalism/ (2013).

[Rutk09] J. Rutkowska: Evil Maid goes after TrueCrypt! (2009). http://theinvisiblethings.blogspot.de/2009/10/evil-maid-goes-after-truecrypt.html

[Schm07] K. Schmeh: Kryptographie – Verfahren, Protokolle, Infrastrukturen. dpunkt.Verlag GmbH, 3 Aufl. (2007).

[Schm12] J. Schmidt: Boot-Sicherung: Verschlüsselte Linux-Notebooks absichern (2012). http://www.heise.de/ct/inhalt/2012/03/146

[Viel07] R. Vieler: Professional Rootkits. Wiley Publishing, Inc., 1 Aufl. (2007).

[Woll13] P. Wollenweber: Sicherheit und Netze (2013). http://www.wollenweber-kl.de/13WS/CNAM