Sicherheitsforscher Sönke Huster über Lücken im WLAN-Stack des Linux-Kernels

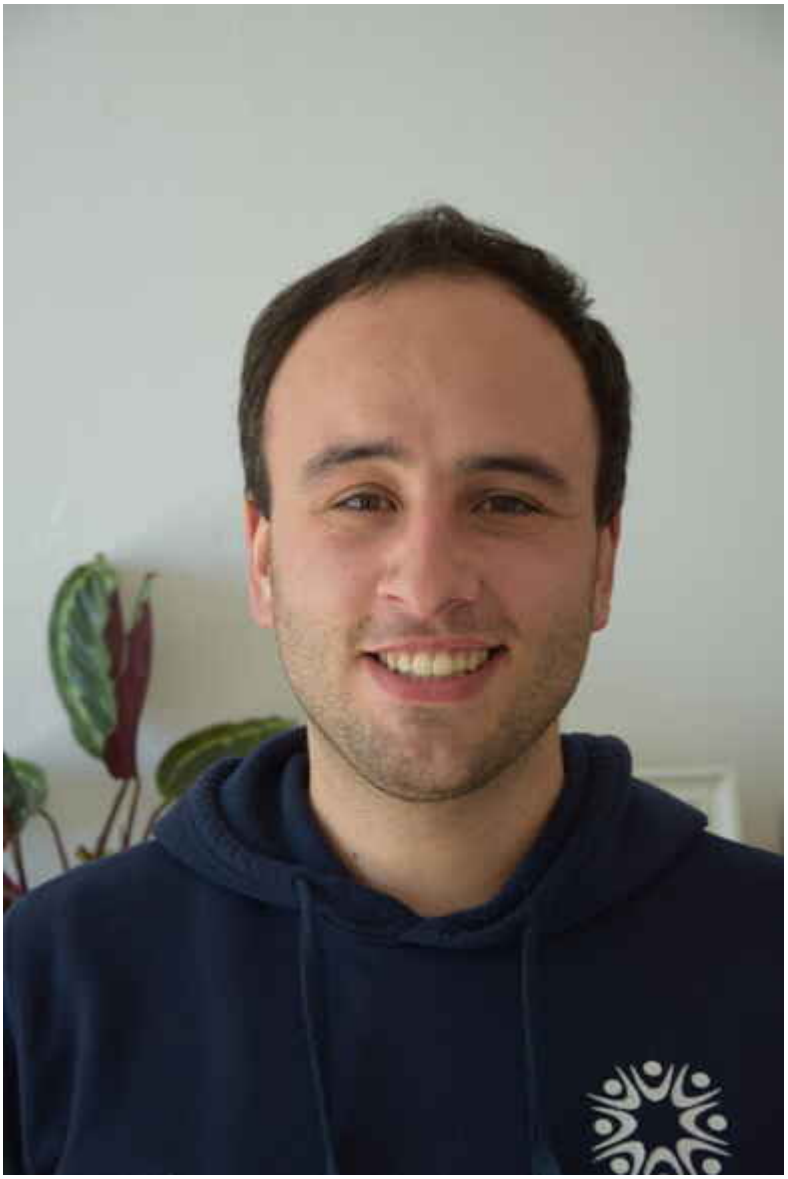
„Es reicht, wenn du dein WLAN anhast“

Über die Luft gehackt werden, nur weil das WLAN eingeschaltet ist? Im August hat Sönke Huster Sicherheitslücken im WLAN-Stack des Linux-Kernels gefunden, die einen solchen Angriff theoretisch ermöglicht hätten. Seine Entdeckung zeigt, wie wichtig es ist, Software ausführlich zu testen.
Über die Luft gehackt werden, nur weil das WLAN eingeschaltet ist? Im August hat Sönke Huster Sicherheitslücken im WLAN-Stack des Linux-Kernels gefunden, die einen solchen Angriff theoretisch ermöglicht hätten. Seine Entdeckung zeigt, wie

wichtig es ist, Software ausführlich zu testen.

Von Kathrin Stoll

Sönke Huster ist wissenschaftlicher Mitarbeiter am Secure Mobile Networking Lab (SEEMOO) der TU Darmstadt. Im August 2022 hat er fünf Sicherheitslücken im WLAN-Stack des Linux-Kernels entdeckt. Mittlerweile gibt es Patches. Wir haben mit ihm über den Fund, seine Methodik und den Disclosure-Prozess gesprochen.



Der Sicherheitsforscher Sönke Huster hat fünf Sicherheitslücken im Linux-Kernel gefunden. Wie er das gemacht hat, verrät er im Gespräch mit c’t. *Josephine Franz*

c’t: Wie kommt man darauf, im Linux-Kernel nach Sicherheitslücken zu suchen?

Sönke Huster: Ich habe dieses Jahr meine Masterthesis über Bluetooth-Fuzzing unter Linux geschrieben. Die Idee kam von meiner Masterarbeitsbetreuerin Dr. Jiska Classen. Im Bluetooth-Stack habe ich dann auch ein paar kleine Sicherheitslücken gefunden. Dann wurde ich wissenschaftlicher Mitarbeiter am Secure Mobile Networking Lab von Prof. Matthias Hollick und es lag nahe, es auf WLAN auszuweiten. Aus Angreifersicht sind WLAN und Bluetooth super interessant und auch irgendwie ähnlich. Wenn ich dich hacken will, ist es ja viel cooler, ich kann das durch die Luft aus dem Raum nebenan machen, ohne dass ich dafür erst physisch auf deinen Rechner zugreifen können muss, um zum Beispiel einen USB-Stick einzustecken. Beide Protokolle sind dafür prädestiniert.

c’t: Du hast gleich fünf Lücken im Linux-Kernel gefunden. Wie bist du dabei vorgegangen?

Huster: Die Methode, die ich verwende, heißt Fuzzing. Sie wurde in den Achtzigerjahren von Barton Miller [Professor der Informatik in Madison, Wisconsin, Anm. d. Red.] entdeckt, der sich über eine Telefonleitung auf Holzmasten remote auf seinem Arbeitsrechner einloggte. Bei Gewitter wurde die Übertragung des Signals gestört und seine Eingaben kamen verzerrt an. Das führte dazu, dass Programme abstürzten oder sich anders verhielten als erwartet. So kam man dahinter, dass man zufällige Eingaben nutzen kann, um Bugs und Sicherheitslücken zu finden und das Fuzzing – auch Fuzz-Testing – war erfunden. Heute verwendet man dazu sogenannte Fuzzer. Das sind im Grunde Programme, die die Eingabeschnittstellen von Programmen, Betriebssystemen oder Netzwerken mit zufälligen Daten fluten.

Mit komplett zufälligen Eingaben arbeitet man heute aber nicht mehr. Man kann das Verfahren verfeinern und Eingaben benutzen, die nah an denen sind, die das Target – in diesem Fall eben Linux in meiner VM – erwartet. Um WLAN zu untersuchen, lasse ich den Fuzzer WLAN-Pakete mit kleinen Anomalien an das Linux-System in meiner virtuellen Maschine schicken, die er fortlaufend verändert. Dabei beobachtet und dokumentiert der

Fuzzer, welcher Code im Kernel zur Verarbeitung der mutierten WLAN-Pakete getriggert wird. Man könnte auch sagen: welchen Weg ein Paket bei der Verarbeitung nimmt. Immer, wenn bei der Verarbeitung eines Pakets Code abgedeckt wurde, der vorher noch nicht ausgeführt wurde, nimmt der Fuzzer dieses Paket in sein Eingabeset auf und nutzt es als Ausgangspunkt für neue Mutationen. Diese veränderten Pakete schickt er dann wieder an den Kernel. Das Ganze passiert ein paar Tausend Mal pro Sekunde. Das Ziel ist es, möglichst viel Code „zu covern“, also durch die mutierten Eingaben Teile des Kernel-Codes abzudecken, die der Fuzzer noch nicht kennt. Coverage-Guided Mutational Fuzzing lautet der Fachbegriff für diese Art von Fuzz-Testing.

c’t: Wenn das Target abstürzt, hat man einen Treffer gelandet?

Huster: Genau. Ein Absturz oder anderes unerwartetes Verhalten, zum Beispiel, wenn es sich aufhängt, sind eigentlich immer ein Hinweis auf einen Bug oder eine Schwachstelle. Die Eingaben, die so etwas bewirken, speichert der Fuzzer separat ab, sodass ich den Crash reproduzieren kann. Bei einer der fünf Lücken, die ich gefunden habe, war es zum Beispiel so, dass ein kaputtes Paket – oder eine Reihe von Paketen – eine sogenannte Linked List korrumpierte und quasi das letzte Paket in der Liste wieder auf das erste gezeigt hat. Bei der Verarbeitung wusste das Betriebssystem nie, wann die Liste zu Ende ist und hat sich schließlich aufgehängt, weil es aus dieser Schleife nicht rauskam.

c’t: Das klingt nach einem ärgerlichen Bug, aber nicht nach einem, den ein Angreifer für eine Remote Code Execution nutzen könnte.

Huster: Nein. Es wäre schwierig, eine Möglichkeit zu finden, das auszunutzen. Die Endlosschleife führt dazu, dass das Betriebssystem sich aufhängt und das wars. Aber eine andere der Lücken ermöglicht es einem Angreifer, den Speicher zu überschreiben, sodass er theoretisch Code aus der Ferne

ausführen könnte. Der Kernel reserviert Speicher für die Ausführung von Programmen und Prozessen. Wenn jetzt beispielsweise 128 Byte an einer Stelle im Speicher für einen bestimmten Vorgang vorgesehen sind, dann darf man da eigentlich auch nicht mehr als diese 128 Byte reinschreiben. Bestimmte Eingaben des Fuzzers haben Fehler in der Paketverarbeitung aufgedeckt, die dazu führen, dass man mehr als die vorgesehene Länge in einen für einen Vorgang reservierten Teil des Speichers schreiben kann – ein sogenannter Buffer Overflow.

c't: Das wäre bereits ausreichend, damit ein Angreifer einen Rechner aus der Ferne übernehmen könnte?

Huster: Theoretisch. Es war möglich, als Angreifer 256 Byte kontrolliert in den Speicherbereich zu schreiben, der auf den zugewiesenen folgte. Für eine RCE müsste man zusätzlich herausfinden, wo im Speicher die kaputten WLAN-Pakete, die diesen Fehler im Kernel-Code triggern, überhaupt verarbeitet werden. Das ist aber gar nicht so einfach, weil es Mechanismen gibt, die dafür sorgen, dass der Kernel immer an unterschiedlichen Stellen im Speicher ausgeführt wird. Kernel Address Space Layout Randomization nennt sich das. Aber es wäre denkbar, dass sich noch weitere Sicherheitslücken finden, die einem das verraten.

c't: Ist das eine Hypothese oder hast du das auch erfolgreich prüfen können?

Huster: Nein. Das übersteigt meine Fähigkeiten. Es ist schon eher eine Hypothese. Aber eine, die sehr wahrscheinlich zutrifft. Es gibt verschiedene Arten von Sicherheitslücken und eine Lücke von diesem Typ bietet sich – in diesem konkreten Fall eben in Kombination mit weiteren – theoretisch dafür an.

Aus Angreifersicht das Spannende an den Sicherheitslücken ist, dass man überhaupt keine Nutzerinteraktion braucht. Du musst dich nicht aus Versehen mit einem Hotspot verbinden, den der

Hacker kontrolliert, damit er dir böse WLAN-Pakete schicken kann. Es reicht, wenn du dein WLAN anhast und dein Gerät nach Netzwerken in der Umgebung sucht. Im Hintergrund passiert das relativ häufig zur Standortbestimmung. Es ist nicht wie bei einem Phishing-Versuch, bei dem der Angreifer das Opfer erst dazu bringen muss, auf einen Button zu klicken und Login-Daten einzugeben. Genau das macht solche Lücken potenziell so kritisch. Linux-Nutzer gibt es nicht so viele, aber drei der Lücken betreffen Android, und Android-Nutzer gibt es eine ganze Menge. Am Smartphone haben die meisten Nutzer ihr WLAN in der Regel an.

c’t: Ist der Fuzzer eine Eigenentwicklung des Secure Mobile Networking Labs?

Huster: Ja. Wir nutzen Komponenten aus LibAFL. Das ist eine Bibliothek, die ein sehr gutes Grundgerüst mitbringt, aber die Architektur unseres Fuzzers unterscheidet sich stark von der bestehender Fuzzer.

c’t: Kannst du sicher sein, dass es außer den fünf Lücken nicht noch weitere gibt?

Huster: Ich denke, man kann auf jeden Fall sagen, dass WLAN unter Linux durch unsere Arbeit ein bisschen sicherer geworden ist. Wir waren an Stellen im Kernel, wo meines Wissens nach noch nicht so viel gefuzzt wurde. Momentan gucken wir uns noch weitere Teile an und bisher haben wir nichts weiter gefunden. Aber hundertprozentige Sicherheit, dass es nicht noch mehr Bugs und Sicherheitslücken gibt, wird man nie haben. Es kann immer unvorhergesehene Eingaben geben, die einen Bug oder eine Sicherheitslücke offenlegen. Ein Angreifer kann sie genauso gut finden wie wir. Genau deshalb ist Fuzz-Testing so wichtig.

c’t: Seit Oktober gibt es Patches. Wie und an wen hast du die Sicherheitslücken gemeldet?

Huster: Es gibt gefühlt 1000 Anlaufstellen für Linux-Sicherheitssachen, zum Beispiel eine Mailing-Liste aller

Hersteller irgendwelcher Linux-Distributionen. Dort hätte ich das melden können. Parallel hätte ich dann noch die Kernel-Leute informieren müssen. Ich hab mich entschieden, den Prozess an einen Hersteller abzugeben und habe mich an SUSE gewandt. Die SUSE-Leute haben Johannes Berg von Intel ins Boot geholt. Er ist der Maintainer des WLAN-Stacks unter Linux. Für mich war es superspannend, mit ihm in so einem engen Austausch zu stehen, während er die Patches für die beiden Sicherheitslücken, die ich initial an SUSE gemeldet hatte, geschrieben hat.

Er hat mir die Patches dann geschickt und ich habe meinen Fuzzer darauf angesetzt. So sind wir auf die drei weiteren Sicherheitslücken – und insgesamt noch ein paar weitere kleinere Bugs – gestoßen. Das Ganze hat ein paar Wochen gedauert. Als alle Patches fertig waren, hat SUSE alle anderen Hersteller im Geheimen informiert und man hat einen Zeitpunkt festgelegt, zu dem man die Öffentlichkeit über die Lücken informiert. Die Hersteller hatten bis dahin über eine Woche Zeit, entsprechende Updates rauszubringen. Überrascht hat mich, dass manche Hersteller ihre Updates erst mehrere Tage nach der Bekanntgabe der Lücken verteilt haben.

c’t: C gilt als relativ unsichere Programmiersprache. Künftig soll es möglich sein, Kernel-Komponenten stattdessen in Rust zu schreiben. Hätte das deine Sicherheitslücken verhindert?

Huster: Sehr wahrscheinlich wären diese Lücken nicht aufgetreten, hätte man die Module in Rust geschrieben. Gerade die Geschichte, dass man Speicher überschreiben kann. Der Rust-Compiler hätte verhindert, dass die Kernel-Entwickler diesen Fehler überhaupt einbauen. Aber es gibt natürlich auch Fehler, die durch keine Programmiersprache der Welt verhindert werden.

c’t: Gibt es etwas, was du Admins und Anwendern raten würdest?

Huster: Sicherheitsupdates immer schnell einzuspielen. Wie

gesagt, bis alle größeren Distributionen die Updates verteilt haben, hat es nach Veröffentlichung noch ein paar Tage gedauert. Gerade bei Android dauert es oft länger. Es kann einfach sein, dass die betreffende Sicherheitslücke schon eine Weile öffentlich ist, bis man als Nutzer ein Sicherheitsupdate bekommt. Deshalb sollte man Updates möglichst sofort installieren. Auch wenn es nervt. Aber dann holt man sich in der Zwischenzeit halt mal einen Kaffee. *(kst@ct.de)*

Weitere Infos: ct.de/yvwk

Microsoft schaltet Linux-Bootloader ab

Ausgebootet

Microsoft schaltet Linux-Bootloader ab

Dank Secure Boot starten Rechner nur noch Betriebssysteme, deren Bootloader von Microsoft signiert wurden. So sollen Rootkits und Bootviren keine Chance haben. Nun hat Microsoft etliche dieser Signaturen per Windows Update zurückgezogen und Linuxe so faktisch lahmgelegt. Wir erklären, wie Sie Ihr Linux trotz Microsofts Boot-Monopol wieder flott bekommen.

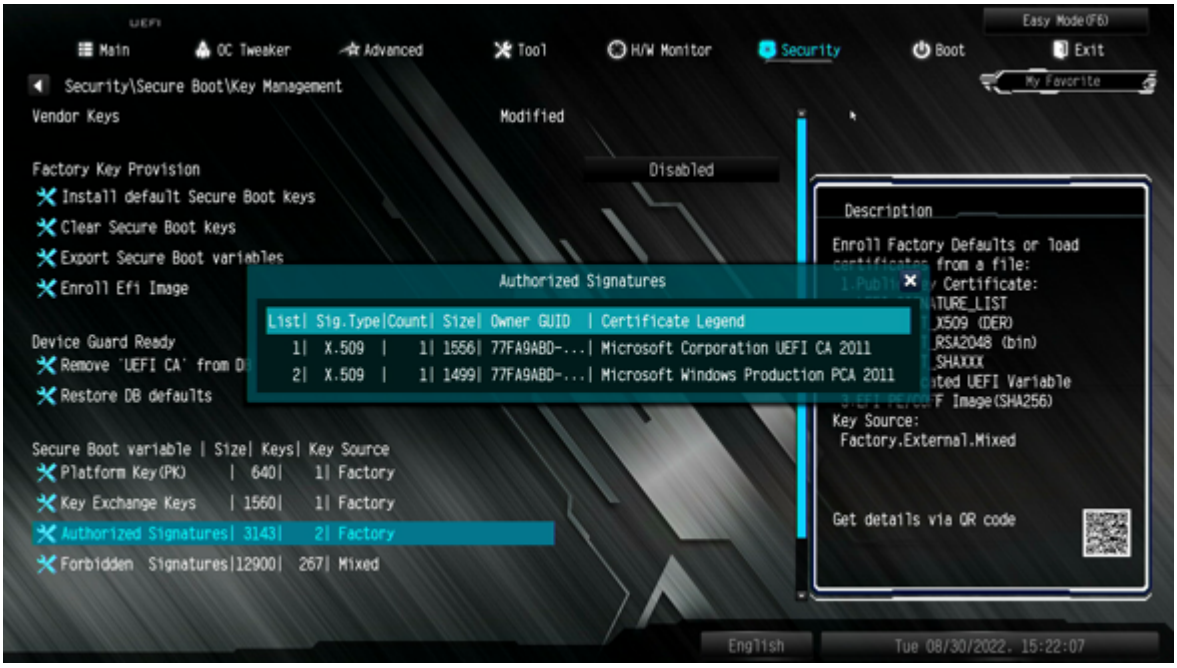
Von Mirko Dölle

Es erinnert an den Betriebssystemkrieg „Microsoft gegen Linux“ zur Jahrtausendwende: Mit dem Windows-Sicherheitsupdate vom 9.

August hat Microsoft über 100 Linux-Bootloader auf die schwarze Liste gesetzt. Seither verhindert UEFI Secure Boot, dass etliche Linux-Distributionen booten. So konnten wir bei Redaktionsschluss das noch immer aktuelle Ubuntu 20.04 LTS und auch Manjaro Linux nicht mehr installieren – außer, man schaltet Secure Boot im BIOS-Setup ab. Auch Live-Linux-Systeme wie etwa Desinfec’t booteten nicht mehr auf PCs, bei denen zuvor das Windows-Update automatisch eingespielt wurde.

Das Update unter Windows wieder zurückzunehmen löst das Problem nicht, weil es den Inhalt des Flash-Speichers auf dem Mainboard ändert, der auch den UEFI-BIOS-Code speichert. Kurzerhand Secure Boot abzuschalten kann das Problem bei manchen Computern sogar vergrößern und zu einem vollständigen Datenverlust unter Windows führen, sodass man am Ende ganz ohne funktionierendes Betriebssystem dasteht.

Laut Beschreibung von Microsoft dient das Update KB5012170 dazu, die Secure-Boot-Plattform zu stärken: UEFI Secure Boot überprüft vor jedem Start, ob der Bootloader des Betriebssystems korrekt mit dem Schlüssel einer vertrauenswürdigen Stelle signiert wurde. In der Praxis ist ab Werk nur ein solcher Schlüssel hinterlegt – nämlich der von Microsoft, damit das üblicherweise vorinstallierte Windows anstandslos bootet. Für Linux-Distributionen war Secure Boot anfangs ein rotes Tuch, schließlich waren der Linux-Bootloader Grub und die Linux-Kernel der Distributionen nicht von Microsoft signiert. Man musste Secure Boot erst ausschalten, um Linux installieren und benutzen zu können.



Praktisch alle Mainboard- und PC-Hersteller tragen im UEFI-BIOS als vertrauenswürdige Schlüssel nur die von Microsoft ein. Damit startet der Rechner ausschließlich von Microsoft signierte Bootloader, solange Secure Boot aktiviert bleibt.

Von Microsofts Gnaden

Die Lösung liefert der freie EFI-Loader Shim, der nur dazu gedacht ist, nach Überprüfung der Signatur den Linux-Bootloader Grub nachzuladen. Der Clou: Linux-Distributoren können ihren eigenen Signaturschlüssel in Shim hinterlegen und somit sicherstellen, dass etwa der in Ubuntu enthaltene Shim lediglich einen von Ubuntu signierten Grub und dieser wiederum einen von Ubuntu signierten Linux-Kernel nachlädt. Die distributionsspezifische Version von Shim lassen sich alle Linux-Distributoren bis heute von Microsoft für Secure Boot signieren, sodass Linux genau wie Windows anstandslos auf PCs mit aktiviertem Secure Boot starten sollte.

Allerdings wurden in den letzten zwei Jahren mehrere Bugs in Grub bekannt, mit denen Angreifer trotz Secure Boot unsignierten Code nachladen können. Microsoft hat darauf mit dem Update von 9. August reagiert und die Signaturen für die von den bekannten Sicherheitslücken betroffenen Linux-Bootloader zurückgezogen. Dazu wurde per Windows Update eine

zusätzliche Revocation List mit den nun gesperrten Bootloader-Signaturen („Forbidden Signatures“, dbx) in den Flash-Speicher des UEFI-BIOS eingespielt.

Die Folge ist, dass sich zum Beispiel das noch bis 2025 gepflegte Ubuntu 20.04 LTS nicht mehr vom USB-Stick booten lässt – weder für Reparaturarbeiten noch für eine Neuinstallation. Ähnlich verhält es sich mit anderen Linux-Distributionen mit Langzeitunterstützung: Ältere Installationsmedien funktionieren nicht mehr. In den letzten Monaten aktualisierte Linux-Installationen auf Festplatte sollten hingegen einen neuen Bootloader erhalten haben, der nicht auf Microsofts Blacklist steht. Damit gibt es dann keine Bootprobleme. Auch der Installationsdatenträger von Ubuntu 22.04 LTS enthält einen neueren Bootloader mit noch gültiger Signatur und lässt sich deshalb uneingeschränkt verwenden.

BitAusLocker

Um eine lokale Linux-Installation trotz eines gesperrten Bootloaders starten zu können, damit man sie auf den aktuellen Stand bringen kann, müsste man Secure Boot im BIOS-Setup des Rechners deaktivieren. Doch das ist nicht ohne Risiken und Nebenwirkungen: Schaltet man Secure Boot später wieder ein und bootet Windows, wird man bei verschlüsselter Systempartition unter Umständen zur Eingabe des BitLocker-Wiederherstellungsschlüssels (Recovery Key) aufgefordert. Warum Windows nach Aus- und Wiedereinschalten von Secure Boot manchmal nach dem Wiederherstellungsschlüssel fragt, haben wir noch nicht herausfinden können, kennen das Phänomen aber sowohl aus eigener Praxis als auch von Leserzuschriften.

Sollten Sie unseren Rat aus [1] ignoriert haben und für die Anmeldung bei Windows ein Microsoft-Konto verwenden, hat Windows Ihren BitLocker-Schlüssel automatisch an Microsoft „zur sicheren Aufbewahrung“ geschickt – dann können Sie (und potenziell auch staatliche Stellen in den USA und anderswo) ihn in Ihrem Online-Profil abrufen und damit Ihre Windows-

Partition entschlüsseln. Wer hingegen nur lokale Benutzerkonten verwendet, sollte den Recovery Key zuvor wie in [2] beschrieben abgerufen und notiert haben. Andernfalls, und falls Sie in Windows Home nur ein lokales Benutzerkonto angelegt haben, kommen Sie nicht mehr an Ihre Daten heran. Deshalb sollten insbesondere Windows-Home-Anwender die Geräteverschlüsselung in den Sicherheitseinstellungen von Windows deaktivieren und warten, bis die Entschlüsselung abgeschlossen ist, bevor Sie etwas an den Secure-Boot-Einstellungen Ihres Rechners verändern.

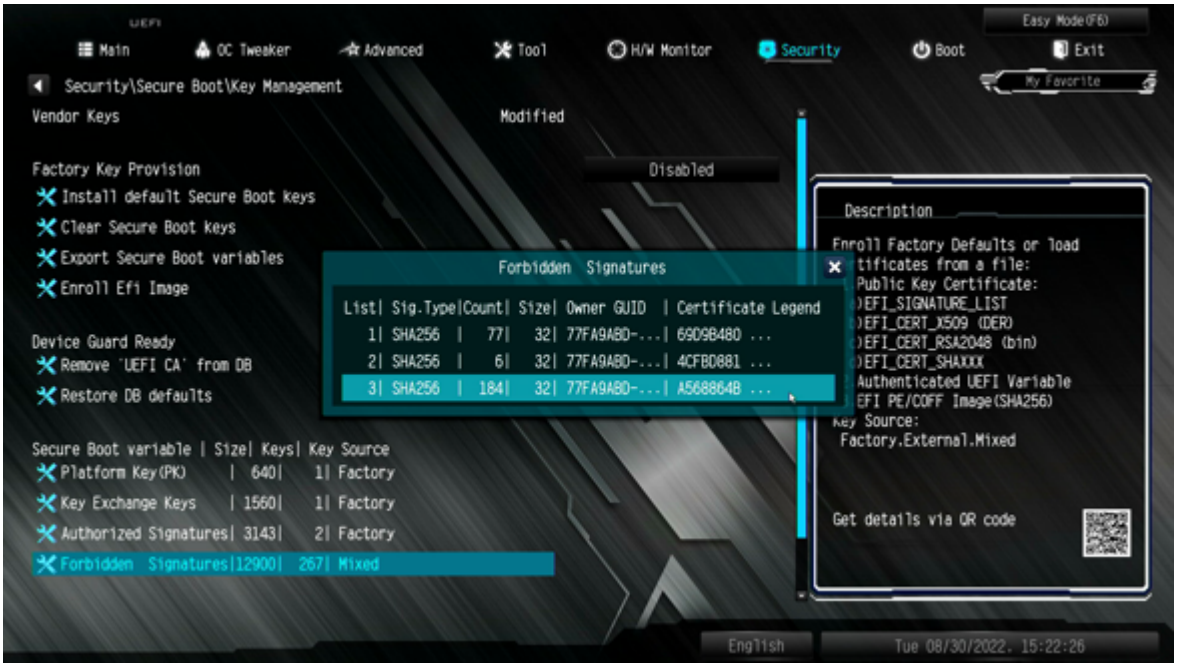
Zurück auf Los

Um Secure Boot nicht ausschalten und Probleme mit Windows riskieren zu müssen, können Sie die Schlüsselverwaltung von Secure Boot in Ihrem BIOS-Setup verwenden, etwa um ein nun gesperrtes Linux für ein Update zu booten. Dazu sollten Sie zunächst über den Update-Verlauf unter Windows das Update KB5012170 entfernen, damit Sie es bei Bedarf zu einem späteren Zeitpunkt wieder einspielen können. Anschließend booten Sie neu und begeben sich ins BIOS-Setup Ihres Rechners, um dort die Änderungen des Windows-Updates wieder rückgängig zu machen.

Unter welchem Menüpunkt Sie die Schlüsselverwaltung für Secure Boot finden, ist nicht standardisiert. Manchmal ist sie in den Boot-Einstellungen versteckt, anderswo in den erweiterten Einstellungen oder Sie müssen erst in die Experten-Ansicht wechseln. Beim Asrock DeskMini versteckte sich der Menüpunkt „Secure Boot“ unter „Security“ im „Advanced Mode“ und wir mussten den „Secure Boot Mode“ erst auf „Custom“ umstellen, bevor wir das „Key Management“ aufrufen konnten.

Dort fanden wir unter „Forbidden Signatures“ (DBX) drei aktuell installierte Revocation Lists mit 77, 6 und 184 Einträgen. Die längste wurde im Rahmen des Updates KB5012170 eingespielt. Indem Sie auf „Delete“ klicken und nicht gleich sämtliche Einträge löschen lassen, gelangen Sie zur Auswahl

der drei Listen und können gezielt die letzte entfernen. Anschließend speichern Sie die Änderungen im BIOS-Setup und können Linux wieder uneingeschränkt booten.



Mit dem Windows-Update vom 9. August wurde eine Liste zurückgezogener Bootloader-Signaturen mit insgesamt 184 Einträgen eingespielt. Indem Sie in der Schlüsselverwaltung des BIOS-Setup diese letzte Blacklist löschen, booten zuvor lahmgelegte Linux-Systeme wieder.

Microsofts Boot-Monopol

Da praktisch alle Hardware-Hersteller nur Microsofts Signaturschlüssel im UEFI-BIOS hinterlegen und Secure Boot seit Jahren auf allen PCs standardmäßig aktiviert ist, hat sich ein neues De-Facto-Monopol für den Software-Konzern aus Redmond ergeben. Mit dem Sicherheitsupdate vom 9. August hat Microsoft unfreiwillig demonstriert, welche Macht es dadurch besitzt: Letztlich entscheidet Microsoft, welche Betriebssysteme auf den Rechnern dieser Welt booten. Die Situation spitzt sich dadurch weiter zu, dass es den Herstellern inzwischen freigestellt ist, ob sie überhaupt noch eine Abschaltmöglichkeit für Secure Boot vorsehen.

Auf der Sicherheitskonferenz DefCon 30 kritisierten Jesse Michael und Mickey Shkatovur Microsofts Praxis, fremde

Bootloader zu signieren: So gelang es ihnen mühelos, eine Signatur für den LOL-Bootloader („Laughing Out Loud“, „laut lachen“) zu bekommen, der beliebige, auch unsignierte Software nachlädt. Das Fazit der Sicherheitsexperten: Secure Boot sei für Angreifer keine Hürde, mache aber Linux-Anwendern das Leben unnötig schwer.

Microsofts Boot-Monopol auf Rechnern praktisch aller Hersteller beweist, dass der freie Markt hier überfordert ist, wenn selbst Größen wie Red Hat, Suse oder Canonical von den Hardwareherstellern ignoriert werden. Deshalb ist es an der Zeit, dass der Gesetzgeber handelt: Man könnte etwa per EU-Einfuhrrichtlinie vorschreiben, dass Schlüssel entsprechend zertifizierter anderer Betriebssystemhersteller gleichberechtigt neben dem von Microsoft installiert werden müssen. So würden Anwender zumindest in Europa die Boot-Hoheit über ihre eigenen Rechner zurückgewinnen. *(mid@ct.de)*

1. Literatur
2. Axel Vahldiek, Zurück in die Kiste!, Windows ohne Microsoft-Konto nutzen, c’t 13/2021, S. 28
3. Jan Schüßler, Wohl oder übel, Keine Angst mehr vor Windows-Updates, c’t 8/2022, S. 148