# — Wirksame Sicherheitsmaßnahmen für IoT-Produkte

## Ein Ergebnis des Forschungsprojekts DEAL – Demonstration, Erklärung, Anleitung und Lehre zu Prinzipien der IT-Sicherheit.

Veröffentlichung: November 2020

Version 1.0

# — Inhaltsverzeichnis

# — Einleitung

Viele IT-Produkte erweisen sich nach ihrer Einführung als unsicher. Prominente Beispiele sind die „Furby“-Kinderspielzeuge, die wegen fehlender Zugriffsbeschränkung von beliebigen Personen ferngesteuert werden können, die WLAN-fähigen „Barbies“, die im Kinderzimmer Gespräche aufzeichnen können, ALDI-Webcams, die ihre Passwörter verraten, die Steuersoftware, über deren unsicheren Update-Mechanismus der WannaCry-Virus initial übertragen wurde und ConnectedDrive von BMW, über das beliebige Fahrzeuge ohne Schlüssel geöffnet werden konnten. Auch andere aufkommende Technologien, wie beispielsweise Solarstromspeicher für den Heimgebrauch, sind davor nicht gefeit.

Die Sicherheitslücken in diesen Beispielen entstehen nicht durch subtile Implementierungsfehler bei der Entwicklung der Software. Sie entstehen vielmehr durch grobe konzeptionelle Fehler beim Entwurf des Systems und durch unzureichende Umsetzung von Standardpraktiken der IT-Sicherheit. Konzeptionelle Schwachstellen sind jedoch für die Sicherheit eines Systems wesentlich bedrohlicher, da sie leichter auch von technisch weniger versierten Angreifern ausgenutzt werden können.

Diese Schwachstellen entstehen in der Regel nicht aufgrund von böser Absicht, sondern sind Ergebnis der hohen fachlichen Komplexität eines sicheren Systementwurfs und der mangelnden Erfahrung von Unternehmen in diesem Bereich. Durch die fortschreitende Digitalisierung und den steigenden Erwartungsdruck der Kunden werden insbesondere kleinere und mittlere Unternehmen immer mehr dazu gezwungen, IT-Funktionalität in Produkte zu integrieren, die bisher völlig ohne IT entwickelt wurden. Dementsprechend fehlt häufig die Expertise im Entwurf sicherer IT-Systeme und kann nicht kurzfristig aufgebaut werden. IT-Systeme sicher zu gestalten und zu betreiben, ist jedoch selbst für IT-Sicherheitsexperten eine Herausforderung.

Im Rahmen des Projekts „DEAL – Demonstration, Erklärung, Anleitung und Lehre zu Prinzipien der IT-Sicherheit“ (gefördert vom Ministerium für Wirtschaft, Arbeit und Wohnungsbau Baden-Württemberg) haben unsere Forscher des Kompetenzzentrums IT-Sicherheit am FZI Forschungszentrum Informatik eine Vielzahl unterschiedlicher Produkte, die IT einsetzen, untersucht. Dabei wurden verschiedene Fehler bei der Umsetzung von IT-Sicherheitsmaßnahmen identifiziert, die in der Praxis zu einer unsicheren Realisierung von zentralen Komponenten des Systems führen.

Bei den Untersuchungen zeigte sich, dass viele Produkte auf einem einfachen Systemmodell basieren: Ein Hersteller bringt ein Produkt, bzw. ein Endgerät, auf den Markt, das für den Gebrauch in den eigenen vier Wänden eines Endkunden gedacht ist. Der Endkunde bindet dieses Endgerät in sein privates Heimnetzwerk ein. Der Hersteller betreibt einen Server, der im Internet öffentlich erreichbar ist und über den einerseits erweiterte Funktionalitäten für das Endgerät bereitgestellt werden können (beispielsweise durch Cloud-Angebote) und der andererseits auch für die Verteilung von Updates und für den Fernwartungszugriff genutzt wird.

In diesem Whitepaper bieten wir konkrete Lösungsvorschläge zur sicheren Umsetzung von kritischen Systemkomponenten in dem beschriebenen Systemmodell. Die Vorschläge bieten dabei aus unserer Sicht jeweils den besten Trade-Off zwischen leichter und praxisnaher Umsetzung und hohem Sicherheitsniveau und sind unabhängig voneinander realisierbar.

# — Angreifermodelle und Schutzziele

## 2.1 Schutzziele

Um ein sicheres IT-System zu erhalten, kann die Gewährleistung von bestimmten Schutzzielen notwendig sein. Drei der bekanntesten Schutzziele sind Confidentiality, Integrity und Availability (CIA). Diese werden häufig durch Authentizität ergänzt. Die Schutzziele sollen im Folgenden genauer erläutert werden.

**Vertraulichkeit** (Confidentiality) ist dann gegeben, wenn die Kommunikation zwischen zwei Parteien so abläuft, dass ein Dritter den Inhalt der Kommunikation nicht in Erfahrung bringen kann. Dies ist insbesondere dann ein wichtiges Schutzziel, wenn es sich bei dem Inhalt um personenbezogene Daten oder Geschäftsgeheimnisse handelt. Hier kann eine Verletzung der Vertraulichkeit auch rechtliche Probleme nach sich ziehen. Mit Mitteln der Kryptographie (in diesem Fall der Verschlüsselung) lässt sich dieses Schutzziel gewährleisten. Mittels Schlüsselaustauschverfahren kann kryptographisches Schlüsselmaterial miteinander geteilt werden, sodass eine vertrauliche Kommunikation ermöglicht wird. Ob diese vertrauliche Kommunikation mit dem gewünschten Partner und nicht mit einem bösartigen Dritten passiert, muss jedoch separat geprüft werden.

Dies lässt sich durch das Schutzziel der **Authentizität** abbilden. Hier geht es darum, dass die Identität der Kommunikationsteilnehmer nachgewiesen wird. Dies lässt sich in der Regel durch eine Public-Key-Infrastruktur (PKI) gewährleisten. In den Fällen, bei denen eine PKI nicht, oder nur schwer, verwendet werden kann, weicht man auf andere Verfahren aus, um authentifiziert Schlüsselmaterial auszutauschen. Hier bietet sich ein Password Authenticated Key Exchange (PAKE) an. Authentizität ist insbesondere bei Softwareupdates wichtig. Gelingt es einem Angreifer, ein von ihm manipuliertes Update aufspielen zu lassen, so kann er die komplette Kontrolle über ein System erhalten. Eine Verletzung der Authentizität bei Updates ist ein häufiges Angriffsszenario. In diesem Whitepaper wird daher auf die korrekte Umsetzung von sicheren Softwareupdates besonderes Augenmerk gelegt.

Viele Maßnahmen sind wirkungslos, falls die dadurch geschützten Daten unbemerkt verändert werden können. Das Schutzziel der **Integrität** (Integrity) bildet diese Anforderung ab. Erreicht wird dies z.B. durch Message Authentication Codes (MACs) oder kryptographische Signaturverfahren.

Bei Produkten, die auf Infrastruktur des Herstellers zugreifen müssen, ist die **Verfügbarkeit** (Availability) der Server, die diese Infrastruktur bereitstellen, wichtig. Gelingt es einem Angreifer, die Infrastruktur zu blockieren, ist das Produkt nicht oder nur eingeschränkt nutzbar. Das kann beispielsweise durch den Versuch einer Überlastung der Server geschehen oder durch die Ausnutzung von Fehlern, die dafür sorgen, dass Server nicht mehr zuverlässig funktionieren. Auch Fehler beim Sicherheitsmanagement können die Verfügbarkeit gefährden, beispielsweise wenn TLS-Zertifikate nicht rechtzeitig erneuert werden.

Um die Sicherheitsanforderungen an ein IT-Systems präzise zu beschreiben, genügt es nicht, nur die gewünschten Schutzziele zu definieren. Vielmehr muss stets auch festgelegt werden, vor welchem Angreifer

man sich schützen will – das sogenannte Angreifermodell. Zu schwache Angreifermodelle sorgen dafür, dass das Sicherheitskonzept nicht geeignet ist, um in der Realität die Schutzziele zu erfüllen; zu starke Angreifermodelle können bedeuten, dass man Systeme als unsicher ansieht, die in der Praxis jedoch wenig Angriffsfläche bieten. Salopp gesagt: Nicht bei jedem System ist es sinnvoll, die am besten ausgestatteten Geheimdienste als Angreifer anzunehmen. Im Folgenden soll Hilfestellung für die Definition sinnvoller Angreifermodelle gegeben werden.

## 2.2 Angreifermodelle

Ein Angreifermodell beschreibt einen abstrakten Angreifer, der innerhalb einer bestimmten Umgebung agiert, sowie dessen Fähigkeiten und Eigenschaften. Ein Angreifermodell weist einem Angreifer in der Regel Ressourcen zu, dazu zählen beispielsweise Geld und technische Infrastruktur, und definiert technisches und domänenspezifisches Wissen und Fertigkeiten. Bei einer Sicherheitsanalyse wird anschließend ein System nach Schwachstellen aus dem Blickwinkel eines solchen Angreifers abgesucht. Ist die Durchführung eines bestimmten Angriffs für einen Angreifer aufgrund von eingeschränkten technischen Fähigkeiten nicht möglich, wird dieser Angriff in der Regel nicht berücksichtigt. Auf diese Art ermittelte Angriffspunkte können anschließend individuell nach ihrem Einfluss auf die Sicherheit des Gesamtsystems bewertet werden.

Angreifermodelle können nicht beantworten, welche Maßnahmen zur Begegnung einer identifizierten Bedrohung gewählt werden sollen. Es obliegt dem Anwender eines solchen Modells, die gefundenen Probleme und Risiken entsprechend ihrem Gefährdungspotenzial zu kategorisieren und Schutzmaßnahmen zu definieren. Wird die Systemarchitektur verändert, kann eine erneute Anwendung des Angreifermodells notwendig sein.

Im Kontext der sicheren Systemupdates sowie der Fernwartung von IoT-Geräten, die in diesem Whitepaper einen besonderen Stellenwert haben, stellen wir zwei relevante Angreifermodelle vor. Für beide Modelle gilt, dass die daraus hervorgehenden Betrachtungen als Basis für weiterführende spezifische Modellierungen genutzt werden sollten. Eine vollumfassende Analyse sämtlicher Aspekte ist mit diesem Ansatz der Bedrohungsmodellierung jedoch nicht möglich und verzweigt sich schnell in komplizierte Modelle mit zahlreichen Annahmen. Sinnvoller ist es, sich auf einfache kleinere Modelle zu stützen und durch einen Abgleich von bekannten Sicherheitsvorfällen die Realitätsnähe eines betrachteten Modells zu gewährleisten.

### 2.2.1 Angreifermodell 1: Angreifer im Internet

Das erste Angreifermodell beschreibt einen klassischen Angreifer, der im Internet agiert. Ein solcher Angreifer ist in der Lage, Datenpakete ungefiltert aus dem Internet an eine Netzwerkschnittstelle eines Endgeräts zu schicken. Damit sind alle über das Netzwerk angebotenen Dienste im Internet exponiert und für den Angreifer erreichbar. Das hat Auswirkungen, sobald diese Dienste keine oder nur unzureichende

Sicherheitsmaßnahmen umsetzen, um die Systemintegrität des Endgeräts zu schützen, oder technische Schwachstellen aufweisen.

Dieser Angreifer hat weiterhin die Möglichkeit, ein baugleiches Endgerät für sich selbst zu erwerben und zu untersuchen. Er ist daher in der Lage, Daten seines eigenen Endgeräts aus dessen Speicher zu extrahieren. Des Weiteren kann er Verhaltenstests in einer kontrollierten Umgebung durchführen, um möglicherweise auffällige Fehlversuche zu kaschieren.

Beispielhaft wird in diesem Modell ein exponierter Konfigurationsdienst eines Endgeräts betrachtet, an dem sich ein legitimer Nutzer authentifizieren muss. Sollte das Endgerät durch den Hersteller nicht mit individuellen Zugangsdaten ausgestattet worden sein, ist es möglich, dass diese Daten bereits in öffentlich verfügbaren Passwortlisten aufgenommen wurden. Damit wird dem Angreifer ermöglicht, automatisierte Authentifikationsversuche durchzuführen und schlussendlich Zugriff auf diesen Dienst des Endgeräts zu erlangen. Alternativ kann der Angreifer auf einem baugleichen Endgerät versuchen, diese Zugangsdaten auszulesen.

Eine Erkenntnis aus diesen Betrachtungen ist somit, dass die Verwendung von identischen Zugangsdaten auf allen Endgeräten einer Baureihe keinen dauerhaft wirksamen Schutz bietet. Eine mögliche Gegenmaßnahme könnte darin bestehen, beim Herstellungsprozess sicherzustellen, dass jedes Endgerät mit individuellen Zugangsdaten provisioniert wird und diese dem Benutzer bei Auslieferung vertraulich mitgeteilt werden. Ebenso könnte bei erstmaliger Inbetriebnahme eines Endgeräts erzwungen werden, dass der Benutzer individuelle Zugangsdaten festlegen muss. Hier besteht allerdings die Gefahr, dass schlechte Zugangsdaten gewählt werden, die einem Angriff mit gängigen Passwortlisten nicht standhalten.

### 2.2.2 Angreifermodell 2: Angreifer im lokalen Netzwerk des Endkunden

In diesem Modell ist der Angreifer ein weiterer Teilnehmer im Heimnetzwerk des Endkunden. Es wird angenommen, dass mit dem Endgerät direkt und ohne technische Einschränkungen wie beispielsweise Firewalls kommuniziert werden kann. Der Angreifer ist dadurch auch in der Lage, den gesamten Datenfluss des Endgeräts zu kontrollieren, mitzulesen und gegebenenfalls zu manipulieren. Erreichen kann er dies beispielsweise über ARP-Cache Manipulationen der beteiligten Geräte, das Betreiben eines WLAN Access Points mit gleicher Kennung oder auch Manipulation der DNS Anfragen eines Geräts. Die Fähigkeiten des Angreifers aus dem ersten Modell werden also um die Fähigkeit, Man-in-the-Middle-Angriffe durchzuführen, erweitert.

Ein derartiger Angreifer ist realistischer, als zunächst zu vermuten ist: Prinzipiell kann jedes Endgerät im Netzwerk ein derartiger Angreifer sein. Kompromittierte Endgeräte können beispielsweise versehentlich eingebunden worden sein, etwa mit dem Ziel, den Datenverkehr des Geräts zu untersuchen. Es ist aber auch denkbar, dass bereits eingebundene Endgeräte nachträglich kompromittiert werden – beispielsweise

geschieht das häufiger durch automatisierte Angriffe auf schlecht abgesicherte Router. Ebenso ist es möglich, dass Schadsoftware in Form von nicht vertrauenswürdigen Apps für PCs, Smartphones oder IoT-Geräte unwissentlich durch den Endkunden installiert wird.

Beispielhaft wird mit diesem Angreifermodell wieder ein Dienst betrachtet, der auf einem Endgerät läuft. Dieser Dienst dient zum Abruf von Kontrolldaten, welche im laufenden Betrieb anfallen. Abgesehen von einer Authentifikation mit starken Passwörtern werden keine weiteren Sicherheitsmaßnahmen getroffen. Durch die Fähigkeit des Angreifers, in diesem Angreifermodell den gesamten Datenverkehr mitlesen und verändern zu können, kann er den Authentifikationsversuch beobachten und dabei Kenntnis über die verwendeten Zugangsdaten erlangen. Darüber hinaus ist der Angreifer in der Lage, die abgerufenen Betriebsdaten in Echtzeit zu verändern und somit dem Nutzer des Dienstes gefälschte Daten vorzulegen.

Anhand dieses Beispiels können jetzt wieder Gegenmaßnahmen definiert werden; dazu gehört die Wahrung der Vertraulichkeit und Integrität der verwenden Zugangsdaten sowie der abgerufenen Kontrolldaten. Diese beiden Ziele lassen sich im Allgemeinen durch den Einsatz von entsprechenden kryptographischen Maßnahmen erreichen. Konkrete Empfehlungen zum Einsatz von Kryptographie werden im Kapitel Weitere Probleme aus der Praxis und mögliche Lösungen beschrieben.

Ein Sicherheitskonzept mit wirksamen Schutzmaßnahmen gegen einen lokalen Angreifer schützt also auch gegen einen Angreifer im Internet. Angreifermodell 2 ist somit stärker als Angreifermodell 1. Gefundene Angriffe und identifizierte Schwachstellen in Systemen werden, wie in den obigen Modellen beschrieben, relevant, sobald das Endgerät in ein Netzwerk integriert wird, unabhängig davon, ob es sich um ein lokal beschränktes Netzwerk handelt oder ebenso ein Internetzugriff möglich ist.

# — Weitere Probleme aus der Praxis und mögliche Lösungen

## 3.1 Authentifizierung im Browser

Webdienste werden auf Endgeräten häufig dafür eingesetzt, die Konfiguration zu vereinfachen. Bevor Konfiguartionsoptionen geändert werden können, sollte sichergestellt werden, dass es sich wirklich um den Besitzer und nicht um einen Angreifer handelt, der Änderungen vornehmen möchte. Viele Geräte nutzen zu diesem Zweck ein Passwort, das beispielsweise im Benutzerhandbuch zu finden ist. Im Rahmen unserer Untersuchungen sind wir auf einen häufigen Fehler bei der Umsetzung von Passwortschutz gestoßen: die Prüfung des Passworts im Browser.

Da der Nutzer (und damit auch ein möglicher Angreifer) volle Kontrolle über den im Browser ausgeführten JavaScript-Code hat, kann er diese Überprüfung sehr einfach deaktivieren und sich ohne Passwort einloggen. Daher sollten Passwörter immer an den Webdienst übertragen und dort überprüft werden. Auch bei der Speicherung und Überprüfung von Passwörtern kann viel falsch gemacht werden.

Das Open Web Application Security Project (OWASP) stellt Ratgeber zu unterschiedlichen Themen der Websicherheit zur Verfügung, unter anderem auch zum sicheren Speichern von Passwörtern, und kann als Anlaufstelle für weitere Informationen dienen. Neben der passwortbasierten Authentifizierung existieren weitere Ansätze, wie beispielsweise der Standard WebAuthn, der eine Authentifizierung mit Public-Key-Kryptographie beschreibt und Sicherheitsprobleme durch schwache Passwörter und Phishing verhindert. Diese neuen Ansätze sind in der Praxis noch nicht weit verbreitet, werden in Zukunft aber voraussichtlich an Beliebtheit gewinnen.

## 3.2 TLS + Zertifikatsprüfung

Transport Layer Security (TLS) ist ein Protokoll, das eingesetzt wird, um transportierte Daten zu schützen. Bei korrekter Nutzung kann es Vertraulichkeit, Integrität und Authentizität gewährleisten. Das im Internet weit verbreitete Protokoll HTTP (Hypertext Transfer Protocol) wird durch die Absicherung mit TLS zu HTTPS. Um die genannten Schutzziele gewährleisten zu können, ist es erforderlich, die Authentizität der jeweiligen Gegenstelle zu prüfen. Die Authentisierung des Servers erfolgt in TLS durch den Einsatz von X.509-Zertifikaten. Beim Verbindungsaufbau übermittelt der Server ein digitales Zertifikat an den Client, indem seine Identität von einer vertrauenswürdigen Stelle (der sogenannten „Certification Authority“) bestätigt wird. Der Client muss dieses Zertifikat prüfen, bevor er mit dem Verbindungsaufbau fortfährt.

Werden Fehler bei der Zertifikatsprüfung gemacht, so können die oben genannten Schutzziele nicht gewährleistet werden. Es kann dann nicht ausgeschlossen werden, dass ein Angreifer die Daten abhört, manipuliert oder sich selbst als Server ausgibt. Die folgenden Empfehlungen sollten beachtet werden:

- Die Zertifikatsprüfung sollte keinesfalls deaktiviert werden. Erfahrung aus unseren Untersuchungen zeigt, dass eine zu Testzwecken deaktivierte Zertifikatsprüfung oft nicht wieder aktiviert wird.

- Die Zertifikatsprüfung sollte, wie alle sicherheitskritischen Operationen, durch etablierte Standardsoftware bzw. Standardbibliotheken erfolgen. Da es sich um einen komplexen Vorgang handelt, wird dringend davon abgeraten, eine eigene Implementierung anzufertigen.
- Standardbibliotheken sind davon abhängig, mit korrekten Parametern konfiguriert zu werden. Die Technische Richtlinie TR-02102-2 des BSI gibt konkrete Empfehlungen zur Parameterwahl bei TLS.

## 3.3 Kryptographische Funktionen selbst implementieren

Kryptographie ist ein etabliertes und grundlegendes Werkzeug für die Gewährleistung von IT-Sicherheit. Verfahren zur symmetrischen Verschlüsselung wie AES und Hashfunktionen wie SHA sind unverzichtbar für eine sichere Kommunikation über öffentliche Netzwerke. Die Sicherheit von kryptographischen Verfahren wird theoretisch auf der Basis von Modellen untersucht, welche von der Praxis stark abstrahieren. Zwischen Theorie und Praxis existiert eine große Lücke und somit eine Vielzahl an Bruchstellen, die häufig Seitenkanalangriffe ermöglichen. Kryptographie selbst zu implementieren bedeutet, dass man sich dieser Lücke bewusst ist und Maßnahmen ergreifen kann, um Seitenkanalangriffe zu verhindern. Implementierung von Kryptographie sollte daher den darauf spezialisierten Experten überlassen werden. Des Weiteren sollte man darauf achten, dass die verwendeten Kryptographiebibliotheken aus allgemein akzeptierten Quellen stammen und professionell implementiert wurden sowie auf dem aktuellen Stand gehalten werden. Beispiele für eine solche Bibliothek sind Bouncy Castle und NaCL. Der Stand der Technik der richtigen Implementierung oder Parametrisierung von kryptographischen Verfahren wird in internationalen Standards der IETF, der NIST oder in der nationalen technischen Richtlinie TR-02102 vom Bundesamt für Sicherheit in der Informationstechnik festgehalten. Die nationalen technischen Richtlinien sind aufgeteilt in mehrere Dokumente, die sich jeweils auf bestimmte Anwendungen beschränken. Sollte beispielsweise TLS in eine Anwendung integriert werden, ist es unumgänglich, sich mit der entsprechenden technischen Richtlinie TR-02102-2 vertraut zu machen. Bei Anpassungen der Parameter, wie z.B. Schlüssellängen, sollte man sich stets an den Empfehlungen der technischen Richtlinien orientieren.

## 3.4 Softwareupdates

Softwareupdates können aus verschiedenen Gründen notwendig sein. Selbst wenn Hersteller nicht beabsichtigen, über die Produktlaufzeit neue Features auszurollen, können Sicherheitsupdates erforderlich sein. Da es kaum möglich ist, fehlerfreie Software zu entwickeln, ist davon auszugehen, dass in jeder selbstentwickelten Software oder in den verwendeten Komponenten anderer Entwickler (z.B. Bibliotheken) Fehler enthalten sind. Werden diese bekannt und können von Angreifern ausgenutzt werden, so ist meist ein Softwareupdate erforderlich, um die Endgeräte und damit auch die Endkunden zu schützen. Daher sollte bereits bei der Produktentwicklung ein Mechanismus vorgesehen werden, über den Softwareupdates eingespielt werden können. Eine Möglichkeit, wie dies realisiert werden kann, ist im Abschnitt Updates beschrieben.

Da auch Software von Drittanbietern sicherheitskritische Fehler enthalten kann, sollten alle eingesetzten Softwarekomponenten stets beobachtet werden und es sollte regelmäßig geprüft werden, ob eine neue Version davon bereitsteht. Ist dies der Fall, so sollte die neue Version dahingehend geprüft werden, ob sie (sicherheits-)kritische Probleme behebt, die das eigene Produkt betreffen oder nicht. Ist dies wiederum der Fall, so sollte zunächst die Kompatibilität getestet werden und die neue Version bei bestandenem Test über den eigenen Updatemechanismus an die Endgeräte ausgeliefert werden.

Weiterhin ist zu beachtet, dass viele Sicherheitsmechanismen im Internet (wie z.B. HTTPS) auf einer funktionierenden globalen Zertifikatsinfrastruktur basieren. Dafür ist es erforderlich, dass Endgeräte stets mit gültigen Wurzelzertifikaten ausgestattet sind. Diese Wurzelzertifikate laufen allerdings nach einigen Jahren ab und müssen ersetzt werden. Die jeweils neuen Wurzelzertifikate müssen auf allen Endgeräten installiert werden, damit diese auch weiterhin in der Lage sind, gesicherte Verbindungen aufzubauen. Auch das von uns vorgestellte Konzept für sichere Updates basiert auf gültigen Wurzelzertifikaten. Sind diese einmal abgelaufen, so funktioniert der Updatemechanismus nicht mehr.

## 3.5 Standardpasswörter

Standardpasswörter (und Standardbenutzernamen) sind eine sehr einfache und verbreitete Methode zur initialen Konfiguration und Installation von neuen Geräten oder Programmen, welche einen beschränkten Zugriff auf bestimmte Funktionen anbieten. Hersteller erlauben damit eine einheitliche Möglichkeit zur Installation und einen einheitlich dokumentierten Weg, auf die eingerichteten Funktionen zuzugreifen. Standardpasswörter erleichtern allerdings auch Angriffe, da sie meistens nach kurzer Zeit im Internet oder in Wörterbuchlisten zu finden sind. Sind durch Standardpasswörter gesicherte Endgeräte oder Dienste im Netz auffindbar, können sie automatisiert von Angreifern übernommen und beispielsweise in ein Botnetzwerk integriert werden. Standardpasswörter bieten damit keine Sicherheit und sollten nicht als Sicherheitsmaßnahme eingesetzt werden. Sie sollten stattdessen im Allgemeinen als initiale Einrichtungsmöglichkeit für Testumgebungen betrachtet werden.

Selbst wenn Standardpasswörter vom Endkunden geändert werden können, geschieht dies häufig nicht. Eine Möglichkeit besteht darin, den Endkunden nach dem ersten Login zu einem Passwortwechsel zu zwingen. Dadurch wird sichergestellt, dass das unsichere Passwort nur für den ersten Login eingesetzt wird. Häufig ist auch zu beobachten, dass bei der Installation des Gerätes noch für viele weitere Dienste, die darauf aktiv sind, Standardpasswörter gesetzt werden. Der Endkunde hat jedoch keinen Überblick über aktivierte Schnittstellen des Endgeräts und kann diese Passwörter dementsprechend auch nicht ändern.

Für den Hersteller ist es in diesem Fall wichtig, auf die vollständige Dokumentation aller aktivierten Schnittstellen zu achten und den Endkunden bei der Einrichtung darüber zu informieren, dass hier ein Dienst mit einem Standardpasswort initialisiert wurde.

An dieser Stelle sei auch auf die Empfehlung des BSI für sichere Passwörter für Hersteller, Integratoren und Betreibende von „Embedded Devices“ verwiesen, die etwas ausführlicher auf dieses Thema eingeht. Die meisten Punkte der Empfehlung lassen sich auch für andere Dienste verallgemeinern.

## 3.6 Bluetooth

Protokolle zur Drahtloskommunikation werden häufig eingesetzt. Bluetooth und insbesondere die stromsparende Variante Bluetooth LE werden daher von immer mehr Geräten unterstützt. Sicherheitsforscher finden jedoch kontinuierlich neue Sicherheitsprobleme in Bluetooth und in Endgeräten, die Bluetooth einsetzen. Auch wir konnten im Rahmen unserer Untersuchungen die unsichere Verwendung von Bluetooth häufig beobachten. Ein häufiges Problem ist die Nutzung von veralteten Verfahren wie dem „Legacy Pairing“ und der Nutzung von neuen Verfahren, jedoch ohne aktivierte Sicherheitsmechanismen. Insbesondere bei Bluetooth LE ist zu beachten, dass es vier Sicherheitsstufen gibt, wobei die niedrigste Sicherheitsstufe keine Sicherheitsmechanismen bereitstellt. Hersteller sollten daher darauf achten, dass die verwendeten Bluetooth-Komponenten Unterstützung für die mit Bluetooth 4.2 eingeführten „Secure Connections“ bieten. Als Authentifizierungsverfahren sollte dann „Numeric Comparison“ oder „Passkey Entry“ verwendet werden und nicht das zwar einfachere aber auch unsichere „Just Works“. Mit dieser Kombination wird die höchste Sicherheitsstufe erreicht. Geräte mit sehr eingeschränkten Möglichkeiten zur Ein- und Ausgabe stellen eine Herausforderung dar, da die empfohlenen Authentifizierungsverfahren die Eingabe oder Anzeige eines Codes erfordern. Diesem Problem kann jedoch mit der sogenannten Out-of-Band-Authentifizierung begegnet werden, die die Authentifizierung über einen anderen Kanal, wie beispielsweise NFC, durchführt. Werden die von Bluetooth zur Verfügung gestellten Sicherungsmechanismen genutzt, so ist es nicht mehr notwendig, Verschlüsselung und Authentifizierung zusätzlich auf Anwendungsebene selbst umzusetzen, was wir leider auch häufig beobachten. Für weitere Informationen zur sicheren Verwendung von Bluetooth verweisen wir an dieser Stelle auf den Leitfaden der NIST.

# — Konzept für sichere Online-Updates

Softwareupdates können aus verschiedenen Gründen erforderlich sein: um neue Funktionen hinzuzufügen, bestehende Funktionen zu ändern oder um Sicherheitslücken zu schließen (siehe auch der Abschnitt 2.2.2 dazu). Wie im Abschnitt Schutzziele beschrieben, gibt es verschiedene Anforderungen an die Sicherheit von Softwareupdates, insbesondere aber sind die Integrität und Authentizität von Updates relevant. Um beides zu gewährleisten, muss das Endgerät in der Lage sein zu prüfen, ob das Update tatsächlich vom Hersteller stammt und ob es manipulationsfrei übertragen wurde. Beides kann durch den Einsatz von etablierten Mechanismen der IT-Sicherheit bzw. mit kryptographischen Mechanismen gewährleistet werden.

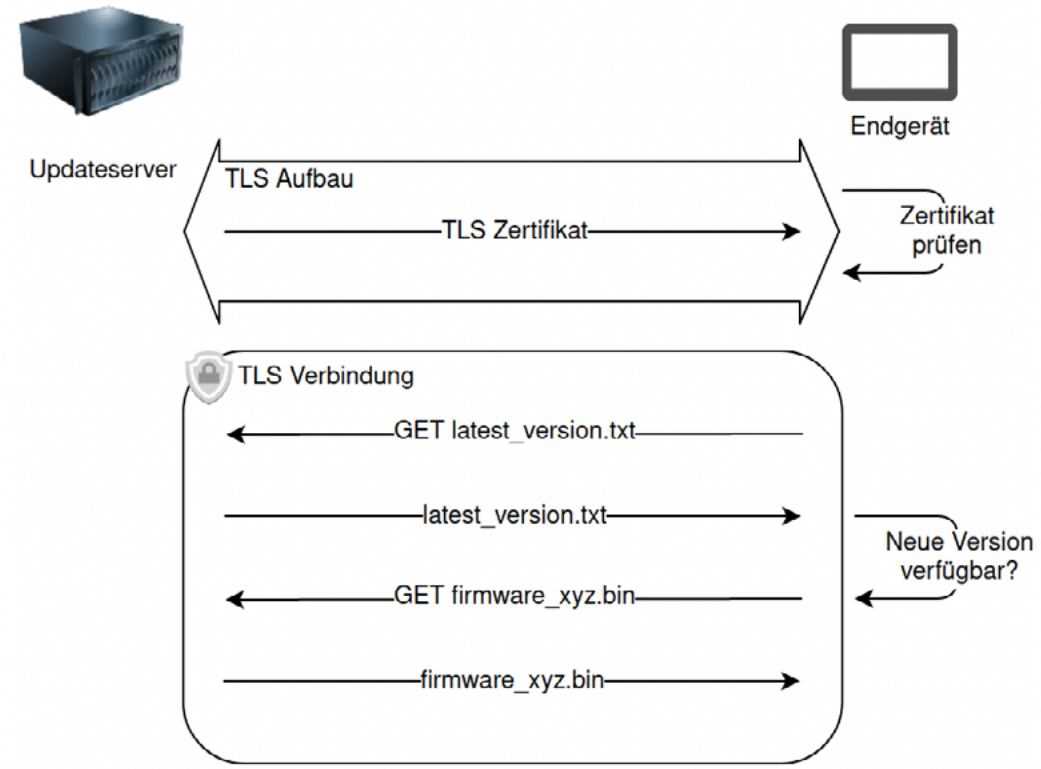
Das im Folgenden vorgestellte Konzept für eine sichere Implementierung eines Online-Updateprozesses ist ein Kompromiss zwischen möglichst einfacher Umsetzbarkeit und maximalem Schutzniveau und ist so gestaltet, dass es auch ohne umfangreiches Fachwissen umgesetzt werden kann.

## 4.1 Angriffe auf Online-Updates

Die Erfahrung aus der Praxis zeigt, dass Updateprozesse häufig ohne Schutzmechanismen für Integrität und Authentizität implementiert werden. Dadurch kann sich insbesondere ein Angreifer im lokalen Netzwerk (siehe Angreifer Typ 2) den Updateprozess zunutze machen, um das Endgerät anzugreifen. Einen solchen Angriff kann er z.B. durchführen, indem er sich als Updateserver ausgibt und dem Endgerät eine manipulierte Softwareversion ausliefert. Über die Manipulation kann der Angreifer z.B. eine Hintertür öffnen, um administrativen Zugriff auf das Gerät zu erhalten, oder anderen Schadcode einspielen. Alternativ kann der Angreifer auch ein legitimes Update während des Herunterladens manipulieren. Angriffe beider Arten können mit dem hier vorgestellten Konzept verhindert werden.

## 4.2 Ablauf

Der Updateprozess wird vom Endgerät initiiert, indem es eine TLS-geschützte HTTP-Verbindung (HTTPS) zu einem vom Hersteller bereitgestellten Webserver aufbaut. Bei diesem Aufbau muss das Endgerät insbesondere das vom Server ausgelieferte Zertifikat genau prüfen, um sicherzustellen, dass es sich beim Server tatsächlich um den Server des Herstellers handelt. Details dazu sind im Abschnitt TLS Zertifikatsprüfung beschrieben. Über diese HTTPS-Verbindung ruft das Endgerät eine zuvor definierte Textdatei ab. Bei dieser Anfrage können optional weitere Parameter, wie der Typ des Geräts oder die Versionsnummer der installierten Firmware, übergeben werden, z.B. als Dateipfad. Die abgerufene Textdatei enthält einen Eintrag mit der Versionsnummer der neuesten verfügbaren Software, sodass das Endgerät prüfen kann, ob eine neuere Softwareversion als die derzeit installierte existiert. Außerdem enthält die Textdatei einen Verweis auf den Ort, an dem die neueste Softwareversion abzurufen ist, sodass das Endgerät diese ggf. herunterladen kann. Auch dieses Herunterladen erfolgt über die geschützte HTTPS-Verbindung, sodass sichergestellt ist, dass das Softwareupdate tatsächlich vom Herstellerserver ausgeliefert wird und bei der Übertragung nicht manipuliert wurde.




## 4.3 Designentscheidungen

### 4.3.1 Initiieren des Vorgangs

Es muss davon ausgegangen werden, dass die meisten Endgeräte nicht vom Internet aus erreichbar sind. Sowohl in Privathaushalten als auch in Unternehmensnetzwerken werden meist Firewalls und andere Technologien betrieben, die den Zugriff auf Geräte im internen Netzwerk beschränken (z.B. NATs). So ist es den meisten Geräten möglich, Verbindungen ins Internet aufzubauen, ein Verbindungsaufbau aus dem Internet zu den Endgeräten ist aber meist nicht möglich. Daher muss das Endgerät die Verbindung zum Server des Herstellers aufbauen und nicht umgekehrt.

**Zusatzinformation**

Wenn der Updatevorgang nur durch das Endgerät selbst initiiert wird, kann insbesondere ein externer Angreifer (siehe Angreifermodell 1) kein Update initiieren oder gar eine manipulierte Softwareversion von außen auf das Gerät hochladen.

### 4.3.2 Automatische Updates

Aus Sicht der IT-Sicherheit sollten insbesondere Sicherheitsupdates so schnell wie möglich installiert werden. Um dies zu gewährleisten, sollte die Installation automatisiert erfolgen, da sich viele Nutzer nicht regelmäßig um Updates kümmern. Der automatischen Installation können Gründe der funktionalen Sicherheit (Safety) oder des Komforts entgegenstehen. Ein Kompromiss kann sein, die Installation automatisiert zu bestimmten Uhrzeiten durchzuführen oder wenn das Gerät gerade nicht benutzt wird. Dies muss im Einzelfall entschieden werden, wir raten aber davon ab, sich beim Auslösen der Installation auf den Nutzer zu verlassen. Das Endgerät sollte daher so konfiguriert sein, dass es periodisch beim Server prüft, ob neue Updates verfügbar sind. Bei der Wahl der Frequenz und des Zeitpunkts sollte beachtet werden, dass die Prüfung nicht auf allen ausgelieferten Endgeräten zum selben Zeitpunkt erfolgt. Andernfalls wird der Updateserver nur zu diesem Zeitpunkt stark beansprucht und kann, abhängig von der Zahl der Endgeräte, möglicherweise nicht alle Anfragen bedienen. Die Häufigkeit, mit der die Endgeräte auf neue Updates prüfen, sollte so gewählt sein, dass ein neues (Sicherheits-)Update möglichst zügig auf allen Endgeräten installiert wird.

## 4.4 Konfiguration des Servers

Der vom Hersteller bereitgestellte Webserver muss so konfiguriert sein, dass das Endgerät jederzeit zweifelsfrei feststellen kann, dass es sich tatsächlich um den korrekten Server handelt. Für den Einsatz von HTTPS wird ein TLS-Zertifikat benötigt. Dieses kann z.B. von Let's Encrypt oder einer anderen Zertifizierungsstelle ausgestellt sein. Hierbei ist zu beachten, dass das Zertifikat für die korrekte Domain ausgestellt und vor seinem Ablauf ausgetauscht wird. Weiterhin sollte die ausstellende Zertifizierungsstelle im CAA Eintrag des DNS hinterlegt sein.

Weiterhin sollte sichergestellt sein, dass nur diejenigen Miterarbeiter*innen Schreibberechtigung haben, die diese auch benötigen. Idealerweise wird der Webserver nicht zusätzlich für andere Zwecke genutzt, sondern exklusiv für das Verteilen der Updates eingesetzt. So können alle nicht benötigten Erweiterungen, wie z.B. PHP und andere Skriptsprachen, deaktiviert werden. Außerdem sollten regelmäßig Sicherheitsupdates eingespielt werden. Es kann hilfreich sein, den Server nicht selbst zu betreiben, sondern bei einem vertrauenswürdigen Drittanbieter anzumieten, um so den Wartungsaufwand auszulagern.

Um Fehlkonfigurationen eines Endgeräts sofort zu entdecken, wird empfohlen, den Server so zu konfigurieren, dass er ausschließlich TLS-gesicherte Verbindungen via HTTPS akzeptiert und HTTP-Verbindungen ablehnt. Weiterhin sollte mindestens TLS Version 1.2 eingesetzt werden.

## 4.5 Konfiguration des Endgeräts

Sowohl beim Prüfen, ob ein neues Update bereitsteht, als auch beim Herunterladen muss das Endgerät sicherstellen, dass es tatsächlich mit dem Server des Herstellers verbunden ist. Hierfür ist es erforderlich, das verwendete TLS-Zertifikat zu prüfen (siehe Abschnitt TLS Zertifikatsprüfung). Eine vollständige Zertifikatsprüfung beinhaltet auch das Prüfen der zeitlichen Gültigkeit des Zertifikats. Damit diese korrekt arbeiten kann, darf die Systemzeit des Endgeräts nicht zu stark abweichen. Es ist daher empfehlenswert, eine automatische Zeitsynchronisierung einzurichten, sodass die Endgeräte z.B. bei jedem Neustart ihre Systemzeit mit einem Zeitserver aus dem Internet synchronisieren.

Beim Herunterladen der Updates muss Folgendes beachtet werden:

- Das Endgerät darf sich ausschließlich per gesicherter Verbindung (HTTPS) verbinden.
- TLS sollte in Version 1.2 oder höher genutzt werden.
- Die Zertifikatsprüfung darf nicht deaktiviert werden.

Unter Linux kann dafür z.B. curl eingesetzt werden. Um sicherzustellen, dass alle Sicherheitsmechanismen korrekt umgesetzt werden, sind zusätzliche Parameter erforderlich:

```
$ curl --proto -all,=https --tlsv1.2 -o update.bin https://hersteller-\
server.de/updates/update.bin
```

Codeblock 1: cURL Parameter

"--proto -all,=https" limitiert die Protokolle, die für den Abruf eingesetzt werden können, auf HTTPS. Ungesicherte Verbindungen via HTTP sind so nicht möglich.
"–tlsv1.2" schränkt die TLS Versionen ein, nur TLS ab Version 1.2 wird akzeptiert.
"-o update.bin" gibt an, wohin die heruntergeladene Datei gespeichert werden soll.

**Zusatzinformation**

Das Nutzen einer TLS-gesicherten Verbindung und eine sorgfältige Prüfung des Serverzertifikats schützen vor Man-in-the-Middle-Angriffen. Derartige Angriffe lassen sich im lokalen Netzwerk einfach ausführen und sind daher für den Angreifer aus Angreifermodell 2 sehr relevant. Ohne eine Überprüfung des Serverzertifikats könnte sich ein Angreifer gegenüber dem Endgerät als Updateserver ausgeben, ein modifiziertes Update ausliefern und sich so Zugang zum Endgerät verschaffen.

## 4.6 Nachteile dieses Konzepts

Das hier vorgestellte Konzept bietet einen Kompromiss zwischen Sicherheit und Einfachheit der Umsetzung. Es kann daher nicht gegen alle möglichen Angreifer schützen und muss bei Bedarf erweitert werden. Insbesondere gegen Angriffe, bei denen ein Angreifer die Domain des Herstellers übernimmt oder Zugriff auf den Updateserver erhält, kann dieses Konzept nicht schützen (siehe unten). Durch den Einsatz von digitalen Signaturen können auch solche Angriffe erschwert werden. Der korrekte Umgang mit digitalen Signaturen ist allerdings komplex und fehleranfällig, weswegen an dieser Stelle nicht weiter darauf eingegangen wird.

### 4.6.1 Domain des Updateservers

Die Domain, unter welcher der vom Hersteller bereitgestellte Updateserver erreichbar ist, ist fest auf den Endgeräten konfiguriert. Sollte der Hersteller die Herrschaft über die Domain verlieren, so kann er keine weiteren Updates mehr verteilen, und ein potenzieller Angreifer, der die Domain unter seine Kontrolle bringt, kann gefälschte Updates mit Schadcode an die Endgeräte ausliefern.

### 4.6.2 Zugriff auf den Updateserver

Jede Person mit Schreibzugriff auf den Updateserver kann dort Dateien platzieren, welche von den Endgeräten heruntergeladen und installiert werden. Der Hersteller muss daher zwingend darauf achten, dass nur vertrauenswürdige Personen Zugriff auf den Server erhalten. Das betrifft auch die Auswahl eines eventuellen Dienstleisters, falls der Server nicht vom Hersteller selbst betrieben wird. Insbesondere sollte beachtet werden, dass Mitarbeiter*innen, die das Unternehmen des Herstellers verlassen, der Zugriff auf den Server entzogen wird.

# — Fernwartung mit SSH

In diesem Kapitel wird beschrieben, wie Wartungs- und Reparaturarbeiten von Endgeräten aus der Ferne, unter Berücksichtigung von IT-Sicherheit, konkret umgesetzt werden können. Voraussetzung dafür ist ein Fernwartungsserver, den der Hersteller betreibt und der über das Internet erreichbar ist.

Die Möglichkeit zur Fernwartung stellt mehrere Anforderungen an eine technische Lösung. Eine Herausforderung besteht darin, dass der Hersteller eine Verbindung zum Endgerät aufbauen können muss, auch wenn dieses an einem Heimanschluss betrieben wird, bei dem Verbindungsanfragen von außen üblicherweise blockiert werden. Zudem sollte sichergestellt werden, dass die Fernwartungsschnittstelle nur vom Hersteller selbst und nicht etwa von Angreifern genutzt werden kann. Da der Zugriff auf das Gerät über das Internet erfolgt, muss zudem die Datenübertragung abgesichert werden. Letztlich sollten keine Geheimnisse auf dem Endgerät gespeichert werden, die bei Kompromittierung den Zugriff auf weitere Endgeräte des Herstellers ermöglichen.

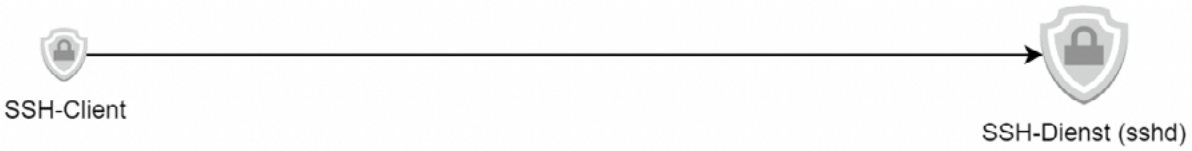
Es existieren verschiedene technische Lösungen für dieses Problem. Mit einem Virtual Private Network (VPN) kann beispielsweise eine gesicherte Verbindung vom Endgerät zum Netzwerk des Herstellers aufgebaut werden. Für die Steuerung des Geräts können Protokolle wie Remote Desktop (RDP) und Secure Shell (SSH) verwendet werden. Die Einrichtung einer derartigen Lösung ist jedoch alles andere als einfach, da die Sicherheit maßgeblich von vielen unscheinbaren Einstellungen abhängt. Wir stellen daher einen konkreten Umsetzungsvorschlag vor.

## 5.1 Probleme mit Fernwartung in der Praxis

Im Rahmen verschiedener Sicherheitsuntersuchungen sind wir auf einige unsicher umgesetzte Fernwartungslösungen gestoßen. Zu den von uns entdeckten Problemen zählten unter anderem im lokalen Netzwerk erreichbare Fernwartungsdienste, schwache Passwörter und Geheimnisse, die von einem Endgerät extrahiert werden können und somit Zugriff auf andere Endgeräte des Herstellers oder die Infrastruktur des Herstellers ermöglichen. Teilweise hatte das zur Folge, dass ein Angreifer alle Geräte eines Herstellers unter seine Kontrolle bringen konnte. Diese und weitere Schwachstellen wurden beim Entwurf der im Folgenden vorgestellten Fernwartungslösung berücksichtigt und entsprechende Maßnahmen zur Absicherung getroffen.

## 5.2 Secure Shell (SSH)

Das Protokoll SSH ermöglicht den Aufbau einer gesicherten Verbindung zu einem entfernten Gerät. Über diese Verbindung können dann Kommandozeilenbefehle ausgeführt werden. Im Wesentlichen besteht SSH aus zwei Komponenten: Dem Client-Programm ssh sowie dem Dienst sshd, welcher auf dem Gerät, auf dem man sich einloggen möchte, läuft. Dieser Dienst kann beispielsweise auf einem Fernwartungsserver, aber auch auf den Endgeräten laufen.




Über die gesicherte Verbindung können nicht nur Befehle ausgeführt, sondern auch Daten übertragen werden. Mit SSH kann dadurch eine ähnliche Funktionalität wie mit einem VPN umgesetzt werden. Der Vorteil ist, dass man sich den zusätzlichen Einrichtungs- und Wartungsaufwand eines VPNs spart, sofern SSH sowieso für den Login benötigt wird.

### 5.2.1 Authentisierung

#### 5.2.1.1 Client

Beim Aufbau einer Verbindung zwischen einem Client und einem Dienst findet eine gegenseitige Authentisierung statt. Benutzer können sich nicht nur mit einem Passwort authentisieren, sondern auch ein Authentisierungsverfahren auf Basis von Public-Key-Kryptographie verwenden. Wir empfehlen grundsätzlich die Verwendung eines Public-Key-Verfahrens. Der öffentliche Schlüssel des Benutzers wird dafür auf dem Server in die Datei *~/.ssh/authorized_keys* eingetragen. Zugriff auf den Dienst wird nur den Clients mit einem zum öffentlichen Schlüssel passenden privaten Schlüssel gewährt. Ein Erraten des geheimen Schlüssels ist, im Gegensatz zu Passwörtern, nicht möglich.

#### 5.2.1.2 Server

Der Server wird ausschließlich auf Basis von sogenannten Host-Keys mit Public-Key-Kryptographie authentifiziert. Die Authentisierung des Servers ist wichtig, da sonst Clients, die standardmäßig deaktivierte Funktionen wie beispielsweise AgentForwarding nutzen, angegriffen werden können. Der Host-Key des Servers wird beim ersten Verbindungsaufbau an den Client übertragen und der Nutzer des Clients gewarnt, falls sich der Schlüssel in einem nachfolgenden Verbindungsaufbau ändert, was ein Hinweis für einen Angriff sein kann. Eine noch höhere Sicherheit wird dadurch erreicht, den Host-Key im Voraus auf den Client zu übertragen. Dadurch wird ausgeschlossen, dass das Endgerät beim ersten Verbindungsaufbau angreifbar ist.

## 5.3 Umsetzung der Fernwartung

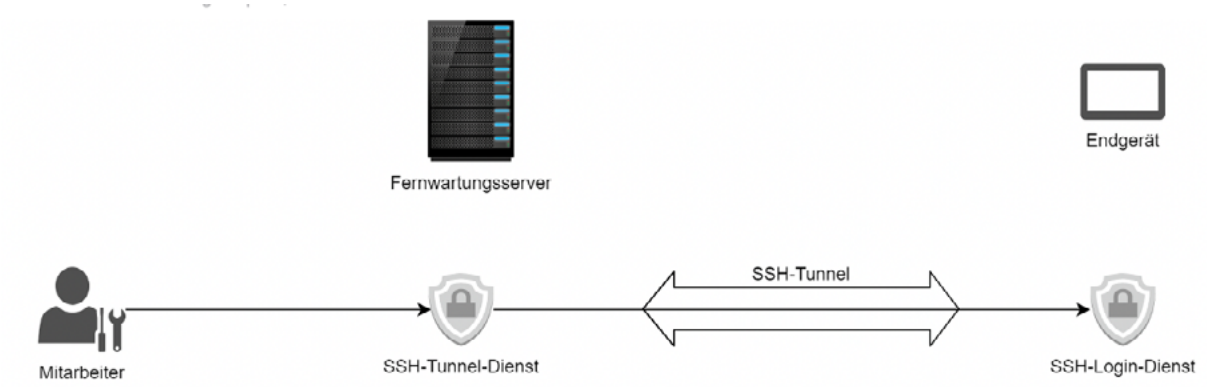
Unsere Architektur sieht einen zentralen Fernwartungsserver vor. Mitarbeiter*innen verbinden sich auf diesen Server per SSH und können von diesem Server wiederum per SSH auf die zu wartenden Endgeräte zugreifen. Endgeräte werden allerdings häufig an Heimanschlüssen und internen Netzwerken betrieben, weshalb Verbindungsversuche von außen in der Regel blockiert werden. Mitarbeiter*innen können sich demnach nicht direkt mit dem Gerät verbinden. Die Verbindung muss durch das Endgerät initiiert werden.

Damit sich Mitarbeiter*innen über den Fernwartungsserver mit dem Endgerät verbinden können, wird eine SSH-Funktionalität namens *Reverse Forwarding* angewendet. Hierfür initiiert das Endgerät eine Verbindung aus dem Netzwerk des Endkunden heraus zum Fernwartungsserver, woraufhin wiederum ein Tunnel vom Fernwartungsserver zum Endgerät aufgebaut wird. Um den Tunnel aufbauen zu können, muss ein SSH-Dienst (im Folgenden SSH-Tunnel-Dienst genannt) auf dem Fernwartungsserver betrieben werden.

Der so eingerichtete Tunnel kann dann für den Login auf dem Endgerät verwendet werden. Für den Login auf dem Endgerät muss ein SSH-Dienst (im Folgenden SSH-Login-Dienst genannt) auf dem Endgerät eingerichtet werden.

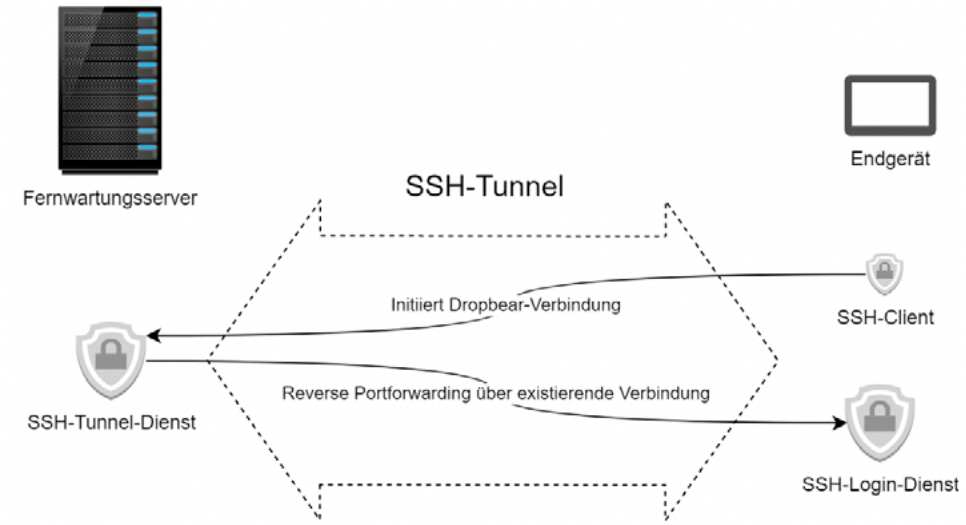
Zur Umsetzung haben wir uns auf dem Fernwartungsserver für die Nutzung von OpenSSH entschieden. Da Endgeräte oft nur sehr eingeschränkte Ressourcen besitzen, schlagen wir dort jedoch den Einsatz der leichtgewichtigen Alternative Dropbear SSH vor.

In den folgenden Abschnitten beschreiben wir die Einrichtung der Fernwartungslösung für einen Fernwartungsserver („remoteserver“ genannt) mit einem Endgerät („device“ genannt) und geben dabei konkrete Konfigurationsempfehlungen. Weitere Endgeräte können analog hinzugefügt werden.



## 5.3.1 Einrichtung des SSH-Tunnels

Als Erstes richten wir den SSH-Tunnel zwischen Fernwartungsserver und Endgerät ein. Über diesen Tunnel findet später der eigentliche Login auf dem Endgerät von außen durch die Firewall statt.




### 5.3.1.1 Einrichtung des Fernwartungsservers

Zunächst legen wir einen neuen Benutzer *tunnel* an, welcher die öffentlichen Schlüssel der Endgeräte verwaltet, die für die Authentifizierung genutzt werden. Der Benutzer wird mit der Login-Shell */bin/false* angelegt, damit er sich nicht einloggen kann. Die Force-Command-Option in der folgenden Konfigurationsdatei hat den gleichen Effekt. Um Sicherheitsproblemen durch spätere Änderungen vorzubeugen, empfehlen wir, beide Optionen zu setzen. Es ist ausreichend, einen Benutzer für mehrere Endgeräte anzulegen, da mehrere Schlüssel hinterlegt und auch selektiv gesperrt werden können. Die öffentlichen Schlüssel der Endgeräte werden, wie eingangs erwähnt, in der Datei *authorized_keys* verwaltet. Hier fügen wir später für jedes Endgerät einen Schlüssel hinzu.

```
user@remoteserver:~$ sudo useradd -m -s /bin/false tunnel
user@remoteserver:~$ su - tunnel
tunnel@remoteserver:~$ mkdir ~/.ssh
tunnel@remoteserver:~$ chmod 700 ~/.ssh
tunnel@remoteserver:~$ touch ~/.ssh/authorized_keys
tunnel@remoteserver:~$ chmod 600 ~/.ssh/authorized_keys
```

Codeblock 2: Anlegen des Tunnel-Benutzers

Als Nächstes erstellen wir die Konfigurationsdatei */etc/ssh/sshd_tunnel_config*. Wir überschreiben absichtlich nicht die Standardkonfigurationsdatei, da wir davon ausgehen, dass bereits für die Administration des Servers ein SSH-Dienst auf dem Server läuft. Die Konfiguration sorgt dafür, dass die Endgeräte lediglich einen eingeschränkten Tunnel zum Server aufbauen und insbesondere keine Befehle auf dem Server ausführen können. Wir haben den Port 443 gewählt, da dieser auch von HTTPS verwendet wird und daher von Firewalls selten blockiert wird.

```
# Dienst auf Port 443 starten, um Blockierung durch Firewall zu vermeiden
Port 443

# Passwort-Authentifizierung deaktivieren
PasswordAuthentication no
# Challenge-Response-Authentifizierung deaktivieren
ChallengeResponseAuthentication no
# Weiterleiten von SSH-Agent deaktivieren
AllowAgentForwarding no
# Weiterleiten von Unix-Domain-Sockets deaktivieren
AllowStreamLocalForwarding no
# Weiterleiten von grafischen Ausgaben deaktivieren
X11Forwarding no
# Erstellen von VPN-ähnlichem Tunnel deaktivieren
PermitTunnel no

# Ausschließlich Reverse Forwarding und kein Local Forwarding erlauben
AllowTcpForwarding remote
# Alle Ziele für Local Forwarding verbieten
PermitOpen none
# Reverse Forwarding von Ports außerhalb von Loopback-Schnittstelle erlauben
GatewayPorts clientspecified
# Reverse Forwarding nur zum speziell angelegten Netzwerkinterface erlauben
PermitListen 10.8.0.1:*

# Shell-Zugang verbieten
ForceCommand echo 'Login disabled.'
# Zugang zum Dienst ausschließlich für Nutzer tunnel erlauben
AllowUsers tunnel
```

Codeblock 3: SSH-Konfiguration des Tunnel-Dienstes

**Zusatzinformation**

Das Einschränken der Konfiguration des SSH-Tunneldienstes schützt sowohl vor einem Angreifer, der aus dem Internet auf den Fernwartungsserver zugreift, als auch vor einem Angreifer, der physischen Zugang zu einem Endgerät des Herstellers hat und Schlüsselmaterial extrahieren kann (Angreifermodell 1). Passwortangriffe werden durch das Deaktivieren der Passwortauthentifizierung verhindert. Ein Angreifer mit physischem Zugang zu einem Gerät kann mit dem Schlüsselmaterial eine Verbindung zum Fernwartungsserver aufbauen und sich authentisieren. Die Konfiguration stellt in diesem Fall sicher, dass der Angreifer dort aber keine Befehle ausführen darf und den Fernwartungsserver auch nicht dafür missbrauchen kann, weitere Verbindungen aufzubauen. Die einzige Funktionalität, die er nutzen kann, ist Reverse Forwarding. Damit erhält er jedoch weder Zugriff auf den Fernwartungsserver noch auf andere Endgeräte.

Wir fügen anschließend eine virtuelle Netzwerkschnittstelle für das Reverse Forwarding hinzu. Das hat den Vorteil, dass die weitergeleiteten Ports, die Zugriff auf den lokalen Login-Dienst des Endgeräts ermöglichen, von der Loopback-Schnittstelle, auf der weitere lokale Dienste laufen, getrennt werden können. Die IP-Adresse 10.8.0.1 wurde beliebig gewählt, sollte jedoch aus einem privaten IP-Adressbereich stammen. Die Konfiguration der Schnittstelle muss bei einem Neustart erneut angelegt werden. Zur dauerhaften Nutzung sollte man das Netzwerkverwaltungstool der jeweiligen Distribution nutzen, um die Schnittstelle automatisiert anzulegen.

```
user@remoteserver:~$ sudo ip tuntap add mode tun dev tun0
# Erstellen einer virtuellen Netzwerkschnittstelle
user@remoteserver:~$ sudo ip addr add 10.8.0.1 dev tun0
# Festlegen einer IP-Adresse für die Schnittstelle
user@remoteserver:~$ sudo ip link set dev tun0 up
# Aktivieren der Schnittstelle
```

Codeblock 4: Hinzufügen einer Netzwerkschnittstelle für Port Forwardings

Der Dienst kann mit dem folgenden Befehl gestartet werden.

```
user@remoteserver:~$ sudo /usr/sbin/sshd -f /etc/ssh/sshd_tunnel_\
config
```

Codeblock 5: Starten des SSH-Dienstes

#### 5.3.1.2 Schlüsselmaterial des Fernwartungsservers auf dem Endgerät hinterlegen

Für die Einrichtung des Clients auf dem Endgerät benötigen wir noch die Host-Keys des Dienstes auf dem Fernwartungsserver. Diese Schlüssel erlauben dem Client zu überprüfen, dass er mit dem richtigen Server verbunden ist und kein Angreifer die Kommunikation abfängt. Die Schlüssel können auf dem Server mit dem folgenden Befehl ausgelesen werden.

```
user@remoteserver:~$ cat /etc/ssh/ssh_host_*.pub
ecdsa-sha2-nistp256 AA... root@remoteserver
ssh-ed25519 AA... root@remoteserver
ssh-rsa AA... root@remoteserver
```

Codeblock 6: Auslesen der Host-Keys des Tunnel-Diensts

Auf dem Endgerät speichern wir die Host-Keys des Fernwartungsservers in der Datei ~/.ssh/known_hosts. Den Ordner ~/.ssh legen wir wie oben beim Fernwartungsserver beschrieben an und setzen die richtigen Berechtigungen. Der Platzhalter „HOSTNAME“ in den folgenden Befehlen muss durch den vollständigen Hostnamen des Fernwartungsservers ersetzt werden.

```
user@device:~$ echo "HOSTNAME ecdsa-sha2-nistp256 AA... root@\
remoteserver" >> ~/.ssh/ssh_known_hosts
user@device:~$ echo "HOSTNAME ssh-ed25519 AA... root@remoteserver" >>\
~/.ssh/ssh_known_hosts
user@device:~$ echo "HOSTNAME ssh-rsa AA... root@remoteserver" >>\
~/.ssh/ssh_known_hosts
```

Codeblock 7: Speichern der Host-Keys auf dem Endgerät

**Zusatzinformation**

Das Übertragen der Host-Keys des Fernwartungsservers auf das Endgerät schützt vor Man-in-the-Middle-Angriffen. Derartige Angriffe lassen sich im lokalen Netzwerk einfach ausführen und sind daher für den Angreifer aus Angreifermodell 2 sehr relevant. Ohne eine Überprüfung der Host-Keys könnte sich ein Angreifer gegenüber einem Endgerät als Fernwartungsserver ausgeben und dadurch Zugang zum lokalen SSH-Login-Dienst verschaffen. Aufgrund unserer Architektur würde das alleine noch nicht reichen, um sich als Angreifer Zugang zum Endgerät selbst zu verschaffen. Dazu ist zusätzlich ein geeigneter privater Schlüssel eines*r Mitarbeiters*in notwendig.

#### 5.3.1.3 Schlüsselmaterial des Endgeräts auf dem Fernwartungsserver hinterlegen

Als Nächstes erstellen wir auf dem Endgerät ein Schlüsselpaar für die Authentisierung am Fernwartungsserver. Wir verwenden das Kryptosystem RSA mit einer Schlüssellänge von 3072 Bit. Das BSI empfiehlt in ihrer Technischen Richtlinie TR-02102-4 eine Schlüssellänge von mindestens 3000 Bit ab dem Jahr 2023.

```
user@device:~$ dropbearkey -t rsa -s 3072 -f .ssh/id_dropbear
Generating 3072 bit rsa key, this may take a while...
Public key portion is:
ssh-rsa AA...
Fingerprint: sha1!! a9...
```

Codeblock 8: Erstellen eines Schlüsselpaars

Den öffentlichen Schlüssel des Schlüsselpaars müssen wir auf dem Fernwartungsserver hinzufügen.

```
user@remoteserver:~$ su - tunnel
tunnel@remoteserver:~$ echo "ssh-rsa AA.." >> ~/.ssh/authorized_keys
```

Codeblock 9: Anlegen des SSH-Benutzers

Letztlich kann der Verbindungsaufbau mit dem folgenden Befehl auf dem Endgerät getestet werden. Der Port 3000 wurde beliebig gewählt, sollte aber für jedes Endgerät individuell sein.

```
user@device:~$ dbclient -p 443 -l tunnel -N -R \
10.8.0.1:3000:localhost:2222 -i .ssh/id_dropbear HOSTNAME
```

Codeblock 10: Aufbauen des SSH-Tunnels auf einem Endgerät mit dbclient

Wir haben nun einen SSH-Tunnel zwischen Endgerät und Fernwartungsserver aufgebaut, über welchen eine Verbindung durch die Firewall auf das Endgerät möglich ist.

## 5.3.2 Einrichtung des Login-Diensts

Nun konfigurieren wir unser eigentliches Ziel, den Login vom Fernwartungsserver auf dem Endgerät.

### 5.3.2.1 Schlüsselmaterial des Endgeräts auf dem Fernwartungsserver hinterlegen

Für den Login müssen wir ebenfalls Schlüsselmaterial zwischen Fernwartungsserver und Endgerät austauschen. Da der Login vom Fernwartungsserver auf dem Endgerät stattfindet soll, müssen diesmal die Host-Keys des Endgeräts auf dem Server hinterlegt werden. Da Dropbear nur die privaten Schlüssel der Host-Keys speichert, benötigen wir das Programm *dropbearkey*, um die zugehörigen öffentlichen Schlüssel auszugeben.

```
$ sudo dropbearkey -y -f /etc/dropbear/dropbear_ecdsa_host_key
Public key portion is:
ecdsa-sha2-nistp256 AA..
$ sudo dropbearkey -y -f /etc/dropbear/dropbear_rsa_host_key
Public key portion is:
ssh-rsa AA..
$ sudo dropbearkey -y -f /etc/dropbear/dropbear_dss_host_key
Public key portion is:
ssh-dss AA..
```

Codeblock 11: Ausgabe der Host-Keys des Login-Diensts

Vor dem ersten Login fügen wir diese Host-Keys manuell auf dem Fernwartungsserver hinzu. Dadurch werden Man-in-the-Middle-Angriffe beim Login verhindert. Da mehrere Benutzer auf das Endgerät zugreifen können, speichern wir die Host-Keys in der systemweiten Datei */etc/ssh/ssh_known_hosts*. Man beachte, dass wir hier einen eindeutigen Namen („CLIENT1“) für das Endgerät wählen. Dieser Name muss nicht mit dem Hostnamen des Endgeräts übereinstimmen und dient lediglich dazu, die Host-Keys mehrerer Endgeräte unterscheiden zu können.

```
root@remoteserver:~$ echo "CLIENT1" ecdsa-sha2-nistp256 AA... root@\
remoteserver" >> /etc/ssh/ssh_known_hosts
root@remoteserver:~$ echo "CLIENT1" ssh-ed25519 AA... root@\
remoteserver" >> /etc/ssh/ssh_known_hosts
root@remoteserver:~$ echo "CLIENT1" ssh-rsa AA... root@remoteserver" \
>> /etc/ssh/ssh_known_hosts
```

Codeblock 12: Hinzufügen der Host-Keys des Login-Diensts auf dem Fernwartungsserver

#### 5.3.2.2 Schlüsselmaterial des Fernwartungsservers auf dem Endgerät hinterlegen

Jetzt fehlt lediglich noch die Einrichtung des SSH-Clients, um sich vom Fernwartungsserver auf das Endgerät zu verbinden. Für den Login auf dem Endgerät erstellen wir ein Schlüsselpaar auf dem Fernwartungsserver. Dabei wird man aufgefordert, ein Passwort zu wählen, das vor jeder Benutzung des Schlüssels eingegeben werden muss. Die Vorzüge der Public-Key-Authentifizierung können jedoch auch ohne Passwort genutzt werden. Dazu muss man bei der Abfrage eines Passworts Enter drücken. Zweck des Passworts ist es, einem Angreifer die Nutzung eines entwendeten Schlüssels zu erschweren.

```
user@remoteserver:~$ ssh-keygen -b 3072 -t rsa
user@remoteserver:~$ cat .ssh/id_rsa.pub
ssh-rsa AA... user@remoteserver
```

Codeblock 13: Starten des SSH-Diensts

Den öffentlichen Schlüssel fügen wir auf dem Endgerät zur Datei authorized_keys hinzu und erlauben somit den Login mit dem zugehörigen privaten Schlüssel.

```
user@device:~$ mkdir ~/.ssh
user@device:~$ chmod 700 ~/.ssh
user@device:~$ touch ~/.ssh/authorized_keys
user@device:~$ chmod 600 ~/.ssh/authorized_keys
user@device:~$ echo "ssh-rsa AA.." >> ~/.ssh/authorized_keys
```

Codeblock 14: Hinzufügen eines öffentlichen Schlüssels für den Login

#### 5.3.2.3 Konfiguration des Endgeräts

Zunächst konfigurieren wir den Login-Dienst auf dem Endgerät. Dazu ändern wir die Konfigurationsdatei von Dropbear unter */etc/default/dropbear*. Wir schränken damit den Zugriff auf das Endgerät ein und deaktivieren den Login mit Passwort.

```
# Enable Dropbear
NO_START=0
# Only listen on localhost
DROPBEAR_PORT="localhost:2222"
# Disable password login
DROPBEAR_EXTRA_ARGS="-s"
```

Codeblock 15: Änderungen in der Dropbear-Konfigurationsdatei

Als Nächstes kann der Dienst gestartet werden.

```
user@device:~$ sudo systemctl start dropbear
```

Codeblock 16: Starten des SSH-Servers auf dem Endgerät

**Zusatzinformation**

Die Dropbear-Konfiguration schützt vor Angreifern, die sich im lokalen Netzwerk befinden (Angreifermodell 2). Die Passwortauthentifizierung wird mit dem Parameter „-s“ deaktiviert und dadurch Angriffe auf schwache Passwörter verhindert. Der Zugang zum SSH-Login-Dienst wird zudem auf die Loopback-Schnittstelle begrenzt und ist somit nicht über das Netzwerk erreichbar. Ein Angreifer müsste sich daher zunächst Zugang zum Fernwartungsserver verschaffen, um überhaupt die Möglichkeit zu haben, auf das Endgerät zuzugreifen.

Ein Login auf dem Endgerät kann mit dem folgenden Kommando ausgeführt werden. Die Portnummer hängt davon ab, welche Portnummer beim Start des Dropbear-Clients angegeben wurde.

```
user@remoteserver:~$ ssh device -p 3000
```

Codeblock 18: Login auf dem Endgerät

### 5.3.3 Einrichtung des Fernwartungsservers für den Login von Mitarbeiter*innen

Wir haben nun erfolgreich die notwendige Infrastruktur aufgebaut, um aus der Ferne auf Endgeräte zuzugreifen. Nun fehlt noch die Möglichkeit für Mitarbeiter*innen, per SSH auf den Fernwartungsserver zuzugreifen. Hier empfehlen wir, einen getrennten SSH-Dienst zu betreiben. Die Konfiguration des Dienstes benötigt keinen großen Aufwand. Wichtig ist auch hier, ausschließlich Public-Key-Authentifizierung zu verwenden und nur Nutzern einer eigens dafür angelegten Gruppe („employees“) den Zugriff zu erlauben. Insbesondere dürfen dem Benutzer *tunnel* keine Berechtigungen gegeben werden, da ein kompromittiertes Endgerät sonst Befehle auf dem Fernwartungsserver ausführen kann. Diese Trennung der Dienste hat den Vorteil, dass der Dienst für die Mitarbeiter*innen ausschließlich vom lokalen Netzwerk aus verfügbar gemacht werden kann, wohingegen der SSH-Tunnel-Dienst vom Internet aus erreichbar sein muss.

```
# Standard-Port für SSH
Port 22
# Passwort-Authentifizierung deaktivieren
PasswordAuthentication no
# Challenge-Response-Authentifizierung deaktivieren
ChallengeResponseAuthentication no
# Zugang zum Dienst ausschließlich für Mitarbeiter*innen
AllowGroups employees
```

Codeblock 19: SSH-Konfiguration für den Login von Mitarbeiter*innen

Um einen Mitarbeiter*innen hinzuzufügen, genügt es, einen Benutzer mit dem Befehl *useradd* anzulegen, diesen der Gruppe *employees* hinzuzufügen und den öffentlichen Schlüssel des*der Mitarbeiters*in in *~/.ssh/authorized_keys* einzutragen. Im Gegensatz zu den Endgeräten erstellen wir für jede*n Mitarbeiter*in einen eigenen Benutzer. Das hat den Vorteil, dass spezifische Zugriffsrechte für Endgeräte vergeben werden können. Zu beachten ist, dass man beim Erstellen des Benutzers für eine*n Mitarbeiter*in eine Login-Shell, wie beispielsweise */bin/bash*, angibt. Andernfalls kann sich der*die Mitarbeiter*in nicht einloggen. Zuvor muss der*die Mitarbeiter*in ein Schlüsselpaar auf seinem Rechner erstellen. Das Erstellen des Schlüsselpaars, des Ordners *~/.ssh* und das Hinzufügen des öffentlichen Schlüssels kann analog zur Einrichtung des Nutzers für den Login auf dem Endgerät, wie im vorangegangenen Abschnitt beschrieben, durchgeführt werden.

```
user@remoteserver:~$ sudo useradd -m -s /bin/bash employee
user@remoteserver:~$ sudo usermod -a -G employees employee
```

Codeblock 20: Hinzufügen eines Benutzers mit Login-Shell und Hinzufügen des Nutzers zu einer Gruppe

#### 5.3.3.1 Provisionierung

Für den dauerhaften Einsatz der Lösung muss sich der Hersteller entscheiden, ob die Fernwartung standardmäßig aktiv oder vom Nutzer manuell freigegeben werden soll. Die Umsetzung der manuellen Freigabe, beispielsweise durch einen Knopfdruck auf dem Endgerät, ist spezifisch für das jeweilige Gerät und wird hier nicht genauer behandelt. Um die Fernwartung automatisch zu starten, müssen wir dafür sorgen, dass der Dropbear-Client auf dem Endgerät automatisch neu gestartet wird, wenn die Verbindung abbricht. Dazu kann das Tool *autossh* verwendet werden. Eine Alternative kann auch ein Provisionierungsdienst sein, der regelmäßig vom Endgerät abgerufen wird, beispielsweise über HTTPS und lediglich bei Bedarf den Aufbau des SSH-Tunnels anfordert. Dies hat den Vorteil, dass weniger SSH-Verbindungen gleichzeitig aufrechterhalten werden müssen. Die Anzahl der Verbindungen wird relevant, wenn der Hersteller eine große Anzahl von Endgeräten betreibt. Über einen derartigen Dienst kann auch die Portnummer für den SSH-Tunnel verteilt und dadurch die Portnummern wiederverwendet werden.

```
user@device:~$ AUTOSSH_PATH=dbclient autossh -M 10000 -p 443 -l \
tunnel -N -R 10.8.0.1:3000:localhost:2222 -i .ssh/id_dropbear HOSTNAME
```

Codeblock 21: Automatisches Neustarten des SSH-Tunnels mit autossh

#### 5.3.3.2 Hinzufügen und Sperren von Endgeräten

Weitere Endgeräte können analog hinzugefügt werden, indem man ein weiteres Mal der Anleitung folgt. Ebenso ist es möglich, bösartige Endgeräte, die beispielsweise Angriffe auf den Fernwartungsserver durchführen, zu sperren. Dazu muss lediglich der entsprechende öffentliche Schlüssel aus der Datei */home/tunnel/.ssh/authorized_keys* auf dem Fernwartungsserver entfernt werden. Zu beachten ist, dass sich die Änderung erst auf neue Verbindungen auswirkt und bestehende Verbindungen gegebenenfalls manuell getrennt werden müssen.

#### 5.3.3.3 Schlüsselverwaltung mit mehreren Endgeräten

Mit der obigen Anleitung können ohne großen Aufwand weitere Endgeräte hinzugefügt werden. Eine wichtige Frage, die sich dabei noch stellt, ist ob das gleiche SSH-Schlüsselpaar für den Login auf allen Endgeräten zum Einsatz kommt. Dies birgt die Gefahr, dass einem Angreifer bei Kompromittierung des Schlüssels der Zugriff auf alle Endgeräte ermöglicht wird. Aufgrund der Architektur der Fernwartungslösung benötigt der Angreifer aber zusätzlich Zugang zum Fernwartungsserver, um sich auf den Endgeräten einloggen zu können. Das Abhandenkommen eines SSH-Schlüssels alleine reicht also nicht aus. Die Kompromittierung eines Mitarbeiter*innenzugangs auf dem Fernwartungsserver ist jedoch ebenfalls ein realistisches Szenario. Dies würde bei einem geteilten Schlüssel bedeuten, dass der Angreifer alle Endgeräte kompromittieren kann. Sofern die Mitarbeiter*innen nur Zugriff auf einzelne Endgeräte benötigen, ist es daher von Vorteil, wenn für jedes Endgerät ein eigener Schlüssel für den Login erstellt wird und dieser nur dem*der Mitarbeiter*in zur Verfügung gestellt wird, die ihn benötigen.

# — Zusammenfassung

Im Rahmen des Projektes „DEAL – Demonstration, Erklärung, Anleitung und Lehre zu Prinzipien der IT-Sicherheit“ haben wir eine Reihe von unterschiedlichen verbraucherorientierten IT-Produkten auf Sicherheitsschwachstellen untersucht. Dabei konnten wir feststellen, dass viele Produkte Schwachstellen enthalten, die in der Regel nicht durch subtile Implementierungsfehler ausgelöst werden, sondern durch die mangelhafte Umsetzung von Standardpraktiken der IT-Sicherheit. Eine häufig auftretende Fehlannahme in der Konzeption der IT-Sicherheitsarchitektur ist, dass Bedrohungen nicht auch aus dem lokalen Netzwerk stammen können.

In diesem Whitepaper haben wir, basierend auf relevanten und realistischen Angreifermodellen, konkrete technische Realisierungsoptionen für zwei Teilsysteme beschrieben, die nach unserer Erfahrung in der Praxis häufig unsicher umgesetzt werden: Updates und Fernwartung. Darüber hinaus haben wir Hinweise zur sicheren Umsetzung für eine Reihe an weiteren, in unseren Untersuchungen regelmäßig aufgetretenen Problemfeldern gegeben.

# — Impressum


## Herausgeber

FZI Forschungszentrum Informatik
Haid-und-Neu-Str. 10–14
76131 Karlsruhe
Tel: +49 721 9654-0
Fax: +49 721 9654-909
www.fzi.de

Stiftung des bürgerlichen Rechts
Stiftung Az: 14-0563.1
ISSN 0930-3014

Vorstand
Prof. Dr. Andreas Oberweis
Jan Wiesenberger
Prof. Dr.-Ing. J. Marius Zöllner
Vorsitzender des Kuratoriums:
Ministerialdirigent Günther Leßnerkraus

Gestaltung, Layout und Satz
Communications (COM), FZI