23. Mai 2019 Studie · Web & Cloud · 13 Min. Lesezeit

Threat Modeling basierend auf Hardware

Archiviert. Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

Stand 2019-05-23
Quelle https://technology2.a-sit.at/veroeffentlichungen/threat-modeling-basierend-auf-hardware/

Threat Modeling ist ein etablierter Ansatz im Feld des Software Engineerings, um die potentiellen IT Sicherheitsschwachstellen eines Artefakts aufzuzeigen und bewusste Entscheidungen über den Umgang mit ihnen zu treffen. Während das auf die Software von IoT-Geräten anwendbar ist, fehlt es etablierten Frameworks wie STRIDE an expliziter Unterstützung für die Hardware Komponenten. Ziel dieser Arbeit ist es, Ansätze zu zeigen, wo im Design Prozess, basierend auf existierenden Unterlagen, ein solches Threat Modeling etabliert werden kann und wie die Kommunikation zum Software Engineering erfolgt. Dabei wird versucht die Kompatibilität zu bestehenden Frameworks aus dem Software Engineering Bereich zu maximieren.

1. Einführung

IoT Geräte machen immer wieder traurige Schlagzeilen mit Sicherheitslücken. Diese reichen von Fehlern, die in der Hardware begründet liegen, über die verwendete Software, sowohl selbstgeschrieben, als auch genutzter Software von Drittanbietern bis hin zu der Kommunikation mit Diensten in der Cloud. Die Fehler erscheinen aus Sicht von sicherheitsversierten Personen oftmals als trivial und wären leicht zu vermeiden, wenn im Design grundlegenden Sicherheitsanforderungen genüge geleistet würde.

Threat Modeling ist die Disziplin, welche als Ziel hat, systematisch Bedrohungen aufzuzeigen und zu bewerten, sodass auf diese in angemessener Art reagiert werden kann. Während dies im Software Engineering bereits weit verbreitet ist und es sehr ausgereifte Frameworks dafür gibt, war diese Art von Threat Modeling bis vor kurzem in Haushaltsgeräten nicht notwendig. Aus diesem Grund fehlt es an Ansätzen, die eine enge Kombination aus Software und Hardware vorsehen.

© A-SIT, Partner für Sicherheitsfragen Im Folgenden werden bestehende Ansätze aus beiden Bereichen aufgezählt, kurz Sicherheitsstandards und deren Relevanz diskutiert und die Differenzen beleuchtet. Abschließend wird versucht die Ansätze aus Soft- und Hardware näher aneinander heranzuführen.

2. Threat Modeling im Software Engineering

Threat Modeling ist ein etabliertes Konzept in der Software Entwicklung. Es geht auf sehr einfache und grundlegende Konzepte, wie Attack Trees, welche bereits im Jahr 1999 von Bruce Schneier vorgestellt wurden, zurück [1]. Daraus haben sich über die Jahre verschiedene, sehr umfangreiche Frameworks entwickelt.

Ein sehr bekanntes solches Framework ist STRIDE, welches von Microsoft entwickelt wurde und sich in den Security Development Lifecycle (SDL) eingliedert. [2] STRIDE ist eine Abkürzung und steht für:

- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege

Ziel des Frameworks ist es, allen Bestandteilen eines Softwareengineeringprojekts die passenden Schwachstellenklassifikationen zuzuordnen. Dazu werden Datenfluss Diagramme (DFD) genutzt. Diese sollten in einem Projekt bereits bestehen. Je nachdem, wie sorgfältig bei deren Erstellung vorgegangen wurde, müssen vorhandene Elemente nur noch in Primitive aufgelöst werden und um Vertrauensgrenzen erweitert werden. Anschließend werden alle Elemente durchnummeriert und nach Vorgaben des Frameworks beurteilt. Beispielsweise ist ein Datenfluss nach diesen Vorgaben anfällig für Tampering, Information Disclosure und Denial of Service.

Bemerkenswert ist es, dass STRIDE bereits exakte Vorgaben darüber macht, welche Bedrohungen auf welches Element des Datenflussdiagramms zutreffen. Dadurch ergeben sich unabhängig von der Person, die die Modellierung durchführt, immer die gleichen Klassifikationen von Bedrohungen. Auf diese Art wird der Diskurs über identifizierte Bedrohungen enorm erleichtert.

DFD Element	S	T	R	I	D	E
Externe Entität	X		X			
Datenfluss		X		X	X	
Datenspeicher		X	X	X	X	
Prozess	X	X	X	X	X	X

Die Zuordnung kann der Tabelle oben entnommen werden.

STRIDE überlässt es den AnwenderInnen, anschließend eine Beurteilung der Risiken vorzunehmen. Dies kann nach eigenen Wünschen gemacht werden, oder mit einem weiteren Framework des SDL, welches DREAD genannt wird, durchgeführt werden. Wir werden in dieser Arbeit nicht näher auf DREAD eingehen, da es bereits einen Schritt nach dem eigentlichen Threat Modeling stattfindet. Nach der Beurteilung ist es notwendig, sich dafür zu entscheiden, wie mit den potentiellen Bedrohungen umgegangen werden soll.

Während STRIDE sehr bekannt ist, besteht der Nachteil darin, dass es ein manueller Prozess ist. Microsoft stellt zwar ein Tool zur Unterstützung zu Verfügung, jedoch kann keine vollautomatisierte Kategorisierung durchgeführt werden. [3] Im Bereich des Software Engineering existieren noch viele weitere Threat Modeling Frameworks. Darunter CORAS [4], welches vor allem im Bereich des Consulting sinnvoll ist und LINDDUN [5], welches sich an die Privacy belangen der Anwender richtet.

3. Standards für die Sicherheit von IoT

Um Geräte im EU Raum auf den Markt bringen zu dürfen, bedarf es eines CE-Siegels. Unter diesem werden eine Reihe von Richtlinien zusammengefasst, welche zur Sicherheit von Produkten beitragen sollen. Sicherheit ist in diesem Fall mit dem Englischen Wort ‚safety', also der Personensicherheit von Geräten, gleichzusetzen. Wichtig ist, dass die Einschätzungen hier ausschließlich vom Hersteller vorgenommen werden, also kein externes Audit verlangt wird. Das CE-Siegel befasst sich insbesondere mit dem entsprechenden Schutz vor gefährlichen Substanzen, Strahlung u.ä. Details dazu sind in einem weiteren A-SIT Bericht zu finden. [6] Wesentlich umfangreicher, jedoch freiwillig, ist eine Zertifizierung nach Common Criteria für IT Produkte. Es handelt sich hierbei um ein Framework nach dem ISO Standard 15408. Basierend auf Sicherheitszielen (Security Targets/ST), welche – sofern für Produktgruppen anwendbar – aus Schutzprofilen (Protection Profiles/PP) entnommen werden, können Hersteller Aussagen über die Sicherheitsattribute ihrer Produkte machen, welche von unabhängigen Testlabors geprüft werden. Sicherheit ist in diesem Zusammenhang im IT Kontext zu verstehen. Ein Beispiel für eine Sicherheitsanforderung (Security Functional Requirement/SFR) ist die Beschreibung **wie** ein Benutzer bei der Ausführung einer bestimmten Rolle authentifiziert werden soll. Die Genauigkeit mit der eine Behauptung überprüft wurde, wird mit einem Evaluation Assurance Level (EAL) beziffert. Diese reichen von 1 bis 7 [7].

Während Common Criteria meist nicht verpflichtend ist, erfordert eine Zertifizierung ein hohes Maß an Sorgfalt bei der Implementierung von Sicherheitszielen um diese tatsächlich zertifizieren lassen zu können.

Politische Institutionen befassen sich ebenfalls mit der Sicherheit von IoT Geräten. In Großbritannien wurde der „Code of Practice for Consumer IoT Security" veröffentlicht. [8] Dieser beinhaltet einen Satz von 13 Regeln, um die derzeit größten Schwächen von IoT Geräten auszumerzen. Dazu zählt u.a. die Verwendung von Standardpasswörtern. Von der ENISA wurden die Baseline Security Recommendations for IoT veröffentlicht. Diese bieten ein sehr umfangreiches Nachschlagewerk zur Beurteilung von Risiken von IoT Geräten [9].

Gezielt für IoT Produkte werden derzeit auch diverse Prüfsiegel von unabhängigen Zertifizierungsstellen wie dem TÜV erarbeitet. Diese befinden sich allerdings noch in einer frühen Phase und sind derzeit noch nicht am Markt verbreitet. Details zu diesen Siegeln können ebenso einem weiteren A-SIT Bericht entnommen werden [6].

4. Threat Modeling in der Hardware Entwicklung

Auch die Hardware Entwicklung stützt sich bereits auf digitale Artefakte. Die „Very High Speed Integrated Circuit Hardware Description Language“, kurz VHDL, wurde 1987 von der IEEE als Standard festgelegt. Sie erlaubt es, Hardware Elemente in Form einer einfachen Beschreibungssprache darzustellen. Dabei wird das gewünschte Verhalten der Schaltung dargestellt. Im Folgenden ein Beispiel für die Implementierung eines D-Flipflop in VHDL: [10]

```
ENTITY DFlipflop IS
PORT(D,Clk, nResetSync: IN Bit;
Q: OUT Bit);
END DFlipflop;
ARCHITECTURE Behav OF DFlipflop IS
BEGIN
PROCESS(Clk)
BEGIN
```

```
IF Clk'EVENT AND Clk = '1' THEN
IF nResetSync = '0' THEN
Q ⩽ '0';
ELSE
Q ⩽ D;
END IF;
END IF;
END PROCESS;
END Behav;
```

Während diese Sprache für die tatsächliche Implementierung der Hardware sehr wichtig ist, erlaubt sie es nur schwer, einen Überblick über die Architektur des Produkts zu erlangen.

Die Spezifikation des Produkts erlaubt hierfür einen geeigneteren Überblick. Sie beinhaltet verbaute Hardware, die verwendete Firmware und (eigens entwickelte) Software, welche dem Produkt die tatsächliche Funktionalität verleihen soll. Dadurch ist schnell ersichtlich, welche Schnittstellen vorhanden sein werden und somit auch, welche potentiellen Einfallstore existieren.

Generell bietet ein modellbasierter Ansatz beim System Engineering eine gute Grundlage, um ein Threat Modeling vorzunehmen, da in diesem Prozess diverse Artefakte angelegt werden, aus welchen sich später nützliche Informationen entnehmen lassen. Dazu zählen vor allem Datenfluss-, Sequenz- und Anwendungsfalldiagramme. [10] Intel hat im Jahr 2011 die Threat Assessment & Remediation Analysis (TARA) veröffentlicht. Es handelt sich hierbei um ein Framework, welches auf beliebige ‚cyber assets‘ anwendbar ist. Dazu zählen u.a. Geräte wie Switches oder VOIP Gateways, wie MITRE in einem Paper hierzu demonstriert. [11] Zu Beginn des Assessments werden ‚Tactics Techniques and Procedures‘ (TTP) potentieller Angreifer identifiziert. Intel verwendet hierfür die CAPEC Angriffsmuster [13], um relevante, mögliche Angriffe zu eruieren. Anschließend wird ein Assessment vorgenommen um den gefundenen Angriffsmustern eine Plausibilität zuzuordnen, bevor diese numerisch bewertet werden und in eine

Bedrohungsmatrix zusammengefasst werden. Anschließend werden Gegenmaßnahmen (Countermeasures/CM) in einer TTP/CM Tabelle herangezogen um entsprechende Vorkehrungen gegen die Bedrohungen zu finden. Die gewonnenen Erkenntnisse werden in umsetzungsfähige Vorschläge übertragen, sodass Entwickler gegen die potentiellen Probleme vorgehen können.

Während das Framework von Intel ausgereift erscheint, fehlt es im Vergleich zu STRIDE an der Reproduzierbarkeit. Es obliegt bei diesem Ansatz sehr stark den Fähigkeiten der Person, die das Threat Model erstellt, welche Schwächen und Schutzmaßnahmen ausgewählt werden.

Der Erkenntnis, dass Hard- und Software in IoT Geräten eine Einheit bilden, hat ARM in der Platform Security Architecture (PSA) Rechnung getragen. [13] Sie basiert auf der Annahme, dass Software Fehler enthält, welche auch Hardware kompromittieren können. Hierbei handelt es sich nicht um ‚versteckte Geheimnisse' (hidden Secrets) sondern um ‚echte' Fehler. Ziel ist es mittels sogenannter ‚Robustheitsregeln' ein sicheres Zusammenspiel zwischen Geräten und Clouds zu ermöglichen. Basis hierfür bildet die „Threat Model and Security Analysis" (TMSA). Hierbei wird zuerst die „Target Of Evaluation" (TOE) festgelegt. ARM hat hierfür ein Beispiel mit einer Sicherheitskamera veröffentlicht. Diese wird unterteilt in Hardware, Betriebssystem und Anwendungssoftware, welche auf dem Gerät läuft. Daraus werden Assets definiert. Diese reichen von der Firmware über Nutzerdaten bis hin zu Netzwerkbandbreite. Für diese werden Bedrohungen identifiziert und anschließend passende Sicherheitsziele. Mit der sogenannten Security Objectives Rationale werden dann Bedrohungen den Sicherheitszielen zugeordnet. Daraus ergeben sich schließlich die Sicherheitsanforderungen, welche im Produkt anzuwenden sind. [14] Auch wenn ein solcher, holistischer Ansatz speziell für IoT sehr wichtig ist, steckt er gerade erst in den Kinderschuhen. Die erste Veröffentlichung von ARM geschah im Oktober 2018.

5. Differenzen

Es zeigt sich für das IoT Umfeld, dass Hardware nichts ohne Software ist und vice versa. Auch wenn ein Threat Model für die selbst geschriebene Software noch so detailliert erarbeitet wurde, wird es nicht genügen, um ein realistisches Bild der Bedrohungslage zu vermitteln.

Das klassische Threat Modeling in der Software, beispielsweise mit Hilfe von STRIDE, nimmt implizit an, dass die Sicherheit der Hardware zu einem gewissen Grad gegeben ist, oder es nicht die Verantwortung der Software ist, auf diese zu achten. Auch wenn externe Entitäten und Prozesse existieren, ist es schwer, mit Hilfe dieser zu einer realistischen Einschätzung zu gelangen. Beispielsweise könnten bestimmte Elemente des Betriebssystems als Prozess dargestellt werden. Das hat jedoch zum einen das Problem, dass auf ein Betriebssystem in den üblichen Software Engineering Artefakten nicht eingegangen wird und ein Prozess ohnehin alle möglichen Bedrohungen zugeordnet bekommt.

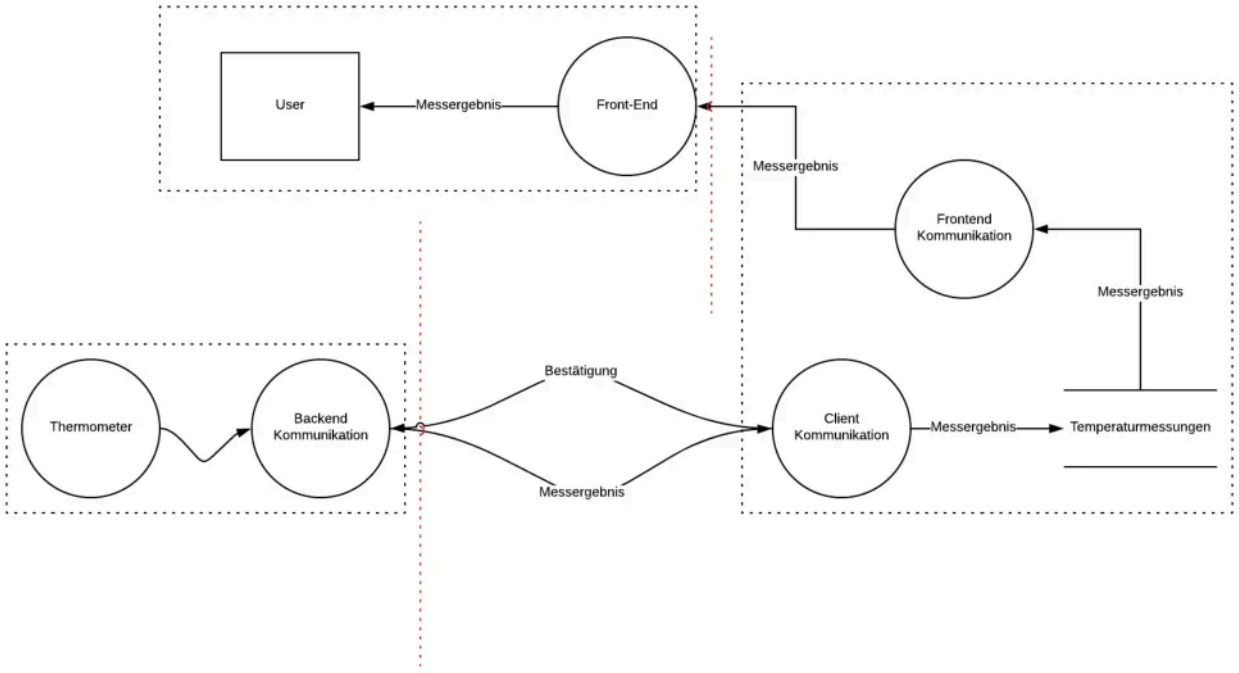
Von der Hardware Seite kommend, erscheint TARA von Intel relativ ausgereift. Die Verwendung von vordefinierten Angriffsmustern erleichtert es, bei der Modellierung über mögliche, unerwartete Szenarien nachzudenken und entsprechend zu behandeln. Nachteil ist, dass es hierbei keine exakte Zuweisung von Hard- und Softwarekomponenten zu diesen Angriffsmustern gibt, wodurch bei fehlendem Einblick in das gesamte Hardwaredesign Schwachstellen übersehen werden können. Das könnte beispielsweise der Fall sein, wenn Debugging-Schnittstellen nicht von den finalen Produkten entfernt werden, welche aber eigentlich nicht als Funktion vorgesehen sind.

Die Artefakte aus dem Hardwaredesign stellen sich ebenfalls als disjunkt zu jenen aus dem Software Engineering dar. Außer bei Verwendung eines modellbasierten Ansatzes wird die eigentliche Funktionalität der Software bzw. Firmware nicht näher erwähnt. Es geht ausschließlich darum, dass sie vorhanden ist und auf das Gerät gespielt wird. Dabei ist die Spezifikation des Produkts noch am ehesten hilfreich für das Threat Modeling, allerdings bietet sie nur eine Auflistung der Teile ohne auf das Zusammenspiel dieser einzugehen.

6. Integration

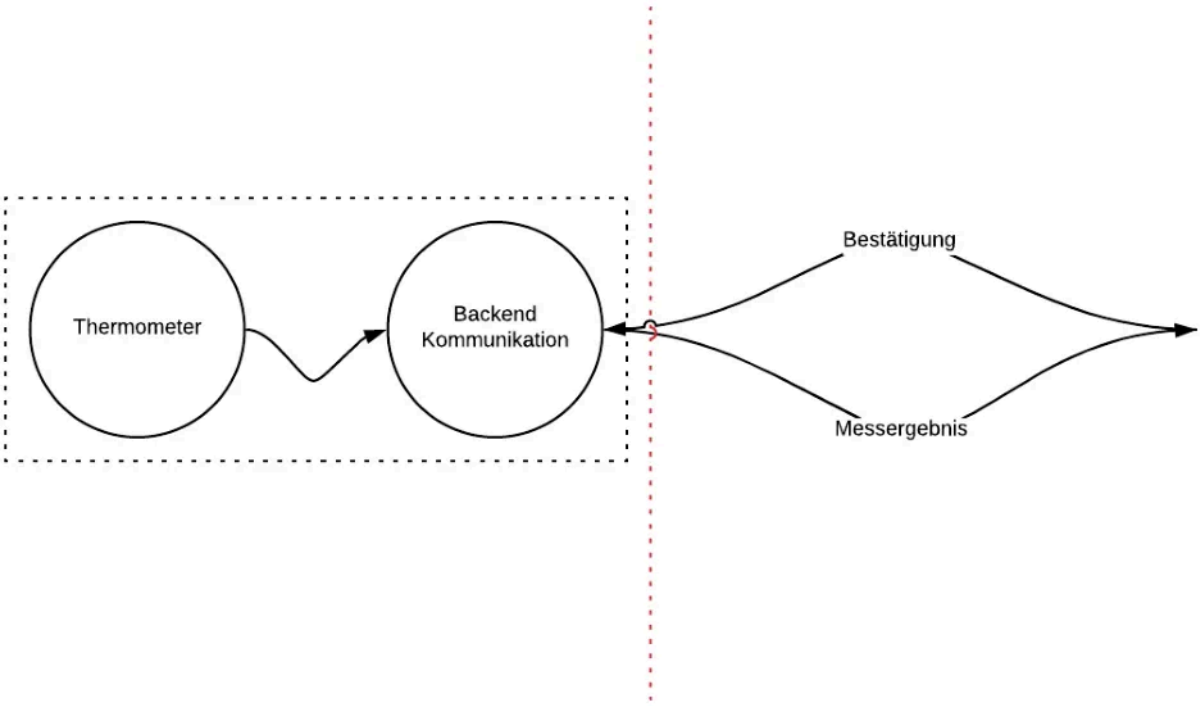
In diesem Abschnitt bieten wir Ansätze, wie es möglich ist, die Domänen Hardware- und SoftwareModellierung mit dem Fokus auf Threat Modeling anzunähern. Wir fangen hierfür mit der Software Seite an.

Als Beispiel wollen wir ein einfaches Datenfluss-Diagramm verwenden. Es handelt sich dabei darum, wie der Datenaustausch zwischen einem Smart Thermometer und einer Front-End Smartphone App aussehen kann. Das Hardware Thermometer liefert die Temperaturmessergebnisse an eine Backendkommunikationskomponente. Diese übermittelt die Daten an einen Server, der sie in einer Datenbank speichert um sie bei Bedarf einer Smartphone App zu Verfügung zu stellen.



Die erste Version des Diagramms ist ausreichend um aus Software Engineering Sicht einen Überblick über die Datenflüsse zu erlangen. Die Trust Boundaries, welche als rot-strichlierte Linien eingezeichnet sind, geben Aufschluss darüber, wann die Daten über ein unsicheres Netzwerk übertragen werden. Im Folgenden wollen wir uns ausschließlich auf das IoT Gerät links unten im Diagramm befassen.

Das IoT Gerät besteht aus Software-Sicht aus zwei großen Prozessen, der Software um Messungen mit dem Thermometer zu machen und dem Softwareteil, der die Messergebnisse an den Server weiterleitet.



Im üblichen Softwareengineering Kontext ist diese Darstellung ausreichend, um ein Threat Model beispielsweise mit STRIDE anzufertigen, wenn die beiden Prozesse noch entsprechend aufgelöst werden. Im IoT Kontext sollte trotzdem ständig die Umwelt beachtet werden. Vor allem, wenn man weiter mit STRIDE arbeiten möchte, lohnt es sich, sie als eigenständigen komplexen Prozess zu betrachten, der noch weiter aufgelöst werden muss. Ebenso ist es bei Kommunikation nach außen besonders wichtig, auf das Betriebssystem und sogar die Hardware zu achten. Auch diese beiden Elemente können als Prozesse dargestellt werden.

Ab diesem Punkt wird es schwerer allgemeingültige Aussagen zu treffen. Jedenfalls sollte dieses Diagramm einen etwas umfassenderen Überblick geben und helfen, Annahmen aus dem klassischen Softwareengineering, wie beispielsweise, dass ein gepatchtes Betriebssystem gegeben ist, zu überdenken. Dazu ist es notwendig, die ‚Prozesse' für die Umgebung, Hardware und Betriebssystem aufzulösen und genau zu betrachten.

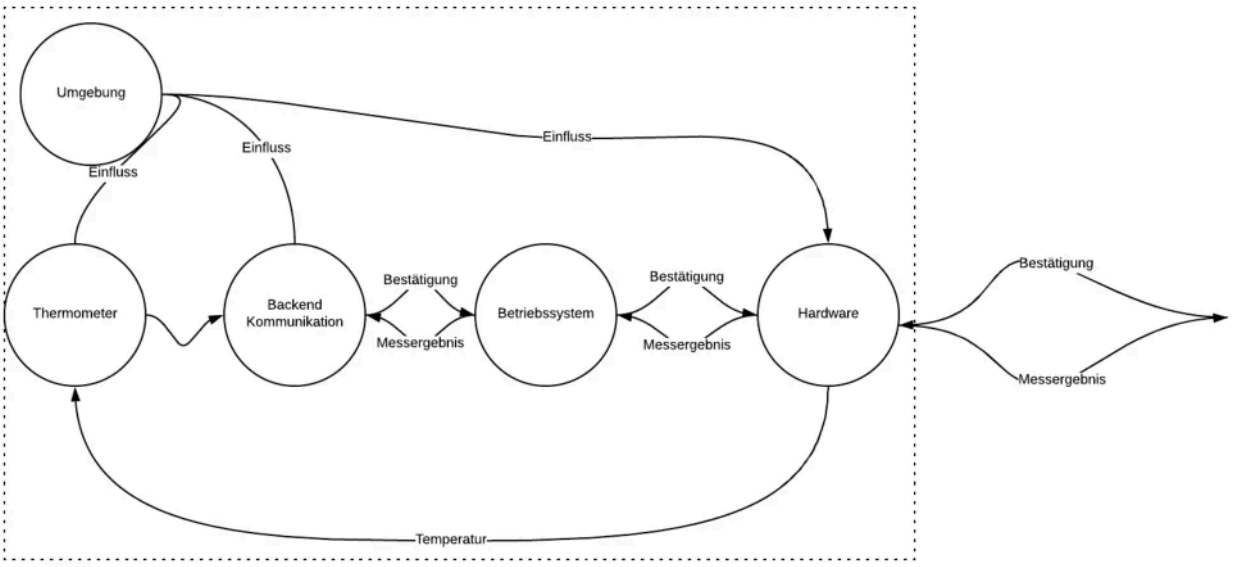
Hinsichtlich der Umgebung müssen Einflüsse bedacht werden, wie die mutwillige Entfernung des Geräts, leere Akkus, oder Umwelteinflüsse wie Wasser. Bei den Überlegungen über mögliche Bedrohungen kann die vormals erwähnte CAPEC Datenbank helfen. [13] Auf Hardwareebene gibt es Spezialitäten, die für einen Softwareengineer unerwartet kommen können. Zum Beispiel hat der Raspberry Pi 3 Model B+ zwar einen Gigabit Ethernet Adapter. Dieser ist allerdings über den USB 2.0 Bus angebunden, weswegen de facto nur 330Mbps Durchsatz erreicht werden können. [16] Während das vermutlich nicht sicherheitskritisch ist, illustriert es sehr gut, welche unerwarteten Designs auf Hardwareebene vorliegen können. Die Spezifikation des Geräts kann hierfür nützliche Inputs liefern.

Der ‚Prozess' Betriebssystem umfasst in diesem Zusammenhang sowohl das eigentliche Betriebssystem, als auch andere Dienste, z.B. ein SSH Daemon, die auf dem Betriebssystem laufen, sowie Einstellungen und Designentschlüsse. Gerade bei IoT ist es notwendig ganzheitlich über die Härtung des Systems

nachzudenken und beim Design auch über effektive und effiziente Möglichkeiten zur Verteilung von Patches nachzudenken. Jüngst gab es aus dem Bereich der Container ein Beispiel dafür, dass auch wenn man sich auf Standardsoftware, die gut getestet sein sollte, verlässt, unerwartete Lücken auftreten können. So wurden einige Images auf Alpine-Linux Basis mit NULL als Passwort für den root Benutzer verteilt. [17]

7. Zusammenfassung

Dieser Bericht bietet einen Einblick in die Welt des Threat Modeling aus der Softwareengineering an Hand von STRIDE.



Wir betrachten zwei Ansätze aus der Hardwareentwicklung, TARA und PSA, und gehen auf die Differenzen zu dem Threat Modeling, wie es aus dem Softwareengineering bekannt ist, ein.

Abschließend bieten wir einen Ansatz, wie die Domänen Hardware und Software in einem gemeinsamen Modell, geeignet für IoT, integriert werden können. Darauf aufbauend können Entwickler sich mit den Bedrohungen auf umfassendere Art befassen.

Referenzen

[1] B. Schneier, „Academic: Attack Trees," [Online]. Available: https://www.schneier.com/academic/archives/1999/12/attack_trees.html. [Zugriff am 07 05 2019].

[2] Microsoft, „Microsoft Security Development Lifecycle," [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl/. [Zugriff am 07 05 2019].

[3] Microsoft, „Download Microsoft Threat Modeling Tool 2016," [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=49168. [Zugriff am 2019 05 07].

[4] K. Stølen, „The CORAS Method," [Online]. Available: http://coras.sourceforge.net/. [Zugriff am 07 05 2019].

[5] D. R. Group, „LINDDUN privacy threat modeling,“ [Online]. Available: https://linddun.org/. [Zugriff am 07 05 2019].

[6] A-SIT, „Beurteilung von Sicherheitsstandards für IoT-Haushaltsgeräte,“ [Online]. Available: https://technology.a-sit.at/veroeffentlichungen/beurteilung-von-sicherheitsstandards-fuer-iot-haushaltsgeraete/. [Zugriff am 07 05 2019].

[7] C. Criteria, „New CC Portal,“ [Online]. Available: https://www.commoncriteriaportal.org/. [Zugriff am 07 05 2019].

[8] Gov.uk, „Secure by Design,“ [Online]. Available: https://www.gov.uk/government/collections/secureby-design. [Zugriff am 07 05 2019].

[9] ENISA, „Baseline Security Recommendations for IoT,“ [Online]. Available: https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot. [Zugriff am 07 05 2019].

[10] Wikipedia, „Very High Speed Integrated Circuit Hardware Description Language,“ [Online]. Available:

https://de.wikipedia.org/wiki/Very_High_Speed_Integrated_Circuit_Hardware_Description_Language.

[Zugriff am 07 05 2019].

[11] Wikipedia, „Systems engineering,“ [Online]. Available:

https://en.wikipedia.org/wiki/Systems_engineering#Using_models. [Zugriff am 07 05 2019].

[12] MITRE, „Threat Assessment & Remediation Analysis (TARA),“ [Online]. Available:

https://www.mitre.org/sites/default/files/pdf/11_4982.pdf. [Zugriff am 07 05 2019].

[13] MITRE, „CAPEC - Common Attach Pattern Enumeration and Classification,“ [Online]. Available:

https://capec.mitre.org/data/index.html. [Zugriff am 07 05 2019].

[14] ARM, „Platform Security Architecture,“ [Online]. Available:

https://developer.arm.com/architectures/security-architectures/platform-security-architecture. [Zugriff am 07 05 2019].

[15] ARM, „Arm Threat Model: Network Camera,“ [Online]. Available: https://pages.arm.com/tmsa resources-collective-network-camera.html. [Zugriff am 07 05 2019].

[16] R. P. Foundation, „Raspberry Pi 3 Model B+,“ [Online]. Available:

https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Zugriff am 10 05 2019].

[17] CVE, „CVE-2019-5021,“ [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE 2019-5021. [Zugriff am 10 05 2019].