Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

A-1030 Wien, Seidlgasse 22 / 9
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

A-8010 Graz, Inffeldgasse 16a
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461
ZVR: 948166612
UID: ATU60778947

15010 DYNAMIC KEY USAGE POLICIES

Florian Reimair – florian.reimair@iaik.tugraz.at

Zusammenfassung: Immer mehr Daten und Ressourcen werden in die Cloud ausgelagert. Auch kryptografische Schlüssel profitieren mittlerweile von den Vorteilen der Cloud. Aktuell verwendete Authentifizierungsmethoden und Abwehrstrategien schaffen es aber oft nicht, Angriffe abzuwehren aber gleichzeitig dem rechtmäßigen Benutzer weiterhin das Service anzubieten. Vielmehr muss der Benutzer oft manuelle Schritte unternehmen, um das Service wieder nutzen zu können. Dadurch bleiben Denial-of-Service Angriffe gegen Benutzer wirksam.

Diese Arbeit stellt einen anderen Ansatz vor, wie solche Systeme geschützt werden können. Ziel ist es, automatisch auf eine Situationsänderung zu reagieren und dynamisch Authentifizierungsmethoden so zu ändern, dass Angriffe gestoppt werden können, der Benutzer trotz Angriff das Service nutzen kann und dem Benutzer möglichst immer die der Situation entsprechend einfachste Authentifizierungsmethode präsentiert wird.

Präsentiert wird eine Architektur die auf Basis von Strategien, Maßnahmen und Richtlinien die eben genannten Ziele erreichen kann ohne dass am Zielsystem große Änderungen notwendig werden. Zwei Prototypen demonstrieren zwei Anwendungsbeispiele aus der Realität.

Inhaltsverzeichnis

1. Einleitung

Immer mehr Daten und Ressourcen werden in die Cloud ausgelagert. Mittlerweile werden auch vermehrt kryptografische Primitive, also Schüsseldaten, in die Cloud ausgelagert [2][1]. Da Webservices an sich schon wertvolle Angriffsziele sind, steigern hochsensible Daten - kryptografische Schlüssel, die möglicherweise rechtsgültige Signaturen ermöglichen - die Attraktivität für Angriffe noch weiter. Gleichzeitig muss der Nutzer aber in der Lage sein, Verträge und/oder Zahlungen jederzeit und überall unterschreiben zu können. Das verlangt nach sehr hoher Verfügbarkeit des Cloud Schlüsseldienstes.

Übliche Schutzmechanismen haben gravierende Nachteile. Die Verwendung solcher kryptografischer Schüssel für kryptografische Operationen (signieren, entschlüsseln) wird durch gängige Authentifizierungsmethoden (PINs, Passworte, SMS TAN, ...) geregelt und so vor Missbrauch geschützt. Im Falle eines Angriffs kann es damit aber passieren, dass der Benutzer einem Denial-of-Service Angriff zum Opfer fällt. Grund dafür ist, dass er erst ein zweites unabhängiges Passwort nachschlagen muss (z. B. PIN und PUK-System von SIM-Karten), ihm für z.B. 30 min der Zugriff gesperrt wird, er erst über einen zweiten Kanal (z.B. eMail) eine Reaktivierung seines Zugangs vornehmen muss, oder er das Service gänzlich verliert und erst nach einem Neuerwerb wieder nutzen kann (z.B. eine als Bürgerkarte aktivierte e-card). All diese existierenden Abwehrstrategien ziehen für den Benutzer ein gewisses Maß an Totzeit und Mehraufwand nach sich.

Dieses Szenario lässt sich generalisieren. Nicht nur kryptografische Schlüssel sind betroffen, sondern jegliche Ressource, die ein Cloud- oder Webservice anbietet. Notizen, Kalenderdaten und eBanking und so weiter sind nur wenige Beispiele aus den potentiellen Anwendungsgebieten.

Diese Arbeit versucht nun andere Ansätze zu finden, um Angriffe abzuwehren aber gleichzeitig dem Benutzer die Ressource ohne Unterbrechung zur Verfügung zu stellen. Im Idealfall wird dem Benutzer dabei immer die für die aktuelle Situation (Angriff oder nicht) am wenigsten komplexe aber ausreichende Authentifizierungsmethode präsentiert. Diese Arbeit stellt dabei keinen Anspruch auf Vollständigkeit, sondern stellt viel mehr ein Sprungbrett dar. Die gefundenen Ansätze sind vielversprechend und können zukünftig weiter verfolgt werden.

Dafür widmet sich diese Arbeit den folgenden Themen. Zunächst wird eine Klassifizierung vorhandener Reaktionen und deren Generalisierung in drei Ebenen Richtlinie, Maßnahme und Strategie durchgeführt. Weiters wird kurz auf Varianten und Möglichkeiten eines Umgebungsmodells eingegangen, wie ein solches Modell einer Strategie als Informationsquelle dienen kann, wie ein solches Modell der Realität folgen kann und wie ein solches Modell Ereignisse erkennen und berichten kann. Auf Basis der Klassifizierung und Details zum Modell wird anschließend eine Architektur vorgestellt, die eine sauber gekapselte dynamische Steuerung von Richtlinien ermöglicht ohne dass das Zielsystem großen Änderungen unterworfen werden muss. Ergänzt wird die Architektur durch zwei Prototypen. Der erste Prototyp demonstriert, dass die Architektur verwendet werden kann, um je nach Quelle der Anfrage (lokales Netzwerk oder VPN) verschiedene Authentifizierungsmethoden zu präsentieren. Der zweite Prototyp demonstriert

eine Abwehr eines Brute-Force Angriffs auf ein Passwort. Eine kurze Zusammenfassung beschließt den Bericht.

2. Strategien für dynamische Schlüsselverwendungsrichtlinien

In diesem Abschnitt werden Strategien diskutiert, die für die Umsetzung dynamischer Schüsselverwendungsrichtlinien geeignet sind. Zu Beginn werden grundsätzliche Maßnahmen diskutiert, die sich Strategien zu Nutze machen können. Im Anschluss werden Strategietypen diskutiert. Abschließend werden Varianten von Richtlinien diskutiert, die von Maßnahmen und damit von Strategien verwendet werden können.

2.1. Definitionen

In diesem Abschnitt werden drei logische Bausteine definiert, die für die Umsetzung von dynamischen Schlüsselverwendungsrichtlinien geeignet sind.

Richtlinie

Eine Richtlinie, im technischen Sprachgebrauch auch als Regel oder Policy bekannt, schlägt die Brücke zwischen einem Sachverhalt und einer Konsequenz. Eine Richtlinie beinhaltet Voraussetzungen und wird aktiv, wenn sie aufgrund ihrer Voraussetzung auf den aktuellen Sachverhalt, z.B. einem Zustand eines Systems, anwendbar ist. Die Richtlinie kann dann mit verschiedenen Konsequenzen reagieren. Die einfachsten Konsequenzen sind einfache ja/nein Entscheidungen. Beispielsweise kann eine Richtlinie verbieten, dass ein spezieller Schlüssel auf einem bestimmten IP-Bereich (z.B. VPN) verwendet werden darf. Andere Richtlinien können Obligationen einfordern. Beispielsweise kann ein Schlüssel nur dann verwendet werden, wenn eine von der Richtlinie vorgegebene Authentifizierung erfolgreich durchgeführt wurde. Richtlinien können so gestaltet werden, dass für die Verwendung von ein und dem selben Schlüssel aus verschiedenen IP-Bereichen (z.B. VPN oder lokaler Zugriff) verschiedene Authentifizierungsmethoden genüge getan werden muss.

Maßnahme

Eine Maßnahme beschreibt, durch welche Richtlinie(n) eine aktive Richtlinie ersetzt wird. Maßnahmen sind also Handlungsanweisungen, die verwendet werden können um auf Veränderungen eines Systems zu reagieren (z. B. wenn das System angegriffen wird).

Strategie

Eine Strategie ist eine Sammlung von Regeln, die Triggerereignisse entgegen nimmt, den Status des Systems evaluiert und entsprechend Maßnahmen aktiviert um z. B. eine laufenden Angriff zu stoppen. Dabei hat eine Strategie in der Regel eine Sammlung von Maßnahmen zur Verfügung die der Situation entsprechend eingesetzt werden können.

2.2. Maßnahmen

Diese Arbeit unterscheidet grob 5 verschiedene Arten von Maßnahmen die sich verschiedener Richtlinien bedienen. Von technologisch gleichen Richtlinien bis hin zu komplexeren Richtlinien können Zeit- und Zugriffszahlbeschränkungen kombiniert werden um eine geeignete Strategie zu

finden. Dieser Abschnitt diskutiert verschiedene Maßnahmen in Hinblick auf Aufwand und Effektivität.

Verwendung einer gleichen Richtlinie

Die einfachste Maßnahme, eine aktive Authentifizierungsrichtlinie zu ersetzen, ist eine andere, aber technologisch gleiche Richtlinie zu verwenden. So kann ohne großen Aufwand eine aktive Richtlinie aus dem Schussfeld genommen werden. Es sind weder große Änderungen auf der Server-Seite notwendig, noch muss die Anwendungs-Seite angepasst werden. Neben der damit aufwandseffizienten Umsetzung bietet eine gleichwertige Richtlinie zur Authentifizierung aber kein Mehr an Sicherheit. Es können potentiell die gleichen Schwachstellen ausgenutzt werden und damit ergibt sich entsprechend wenig Änderungsaufwand auf der Seite des Angreifers.

Verwendung einer gleichwertigen Richtlinie

Im Gegensatz zur Verwendung einer gleichwertigen Richtlinie wir mit einer technologisch anderen Richtlinie eine kurzfristige Erhöhung der Sicherheit erzielt. Da diese Erhöhung aber großteils auf dem Prinzip von Sicherheit durch Geheimhaltung beruht, wird das Sicherheitslevel nach einiger Zeit wieder auf das ursprüngliche Niveau sinken. Dabei ist schwer abzuschätzen, wie lange ein erhöhtes Sicherheitslevel gegeben ist. Der Aufwand für den zu schützenden Service ist wohl höher als im Fall der gleichwertigen Richtlinie, ist aber, vorausgesetzt das Service beherrscht bereits dynamische Richtlinien, überschaubar und einmalig. Auf der Seite des Angreifers fällt mehr Aufwand an da dieser möglicherweise eine Angriffsstrategie für die neue Richtlinie finden muss, diese neue Strategie implementieren und aktivieren muss, und möglicherweise seinen Angriff gegen die ursprüngliche Richtlinie abbrechen muss und damit Zeit verliert. Obwohl diese Maßnahme keine langfristige Verbesserung des Sicherheitslevels ermöglicht kann damit mit wenig Aufwand zum Beispiel auf laufende Angriffe reagiert werden. Wie effektiv diese Maßnahme im Detail ist hängt vom Typ der verwendeten Richtlinien ab.

Verwendung einer mächtigeren Richtlinie

Eine weitere Maßnahme, um eine Authentifizierungsrichtlinie aus der Schusslinie zu nehmen ist es, stattdessen eine Richtlinie zu installieren, die mehr Sicherheit bietet. Eine solche Richtlinie ist gezwungenermaßen technologisch anders und erfordert damit sowohl auf der Seite des Service, als auch auf der Seite des Angreifers Aufwand, um die Richtlinie zu implementieren und anzugreifen. Je nach verwendeter Richtlinie kann es durchaus auch zu höheren laufenden Kosten kommen. Auch für den Benutzer wird die Erfüllung der Richtlinie in vielen Fällen aufwändiger werden. Die Vergangenheit hat gezeigt, dass mehr Sicherheit oft nur durch komplexere Methoden realisiert werden kann [5]. Die Vergangenheit hat auch gezeigt, dass komplexere Abläufe von Benutzern weniger gerne verwendet werden [3, 4]. Somit ist die Verwendung von mächtigeren Richtlinien durchaus eine effektive Maßnahme, um laufende Angriffe zu stoppen bzw. Angriffe langfristig komplexer zu machen. Im Gegensatz zu den vorherigen Maßnahmen bleibt das erhöhte Sicherheitsniveau solange bestehen, solange die mächtigere Richtlinie aktiv bleibt.

Mehrere Richtlinien aneinanderreihen

Während zuvor diskutierte Maßnahmen aktive Richtlinien durch technologisch andere und/oder aufwändigere Richtlinien ersetzen wird auch gerne eine zeitliche Aneinanderreihung von

Richtlinien verwendet. Eine spezielle Art diese Maßnahme ist als Multi-Faktor-Authentifizierung bekannt. Das Ziel dieser Maßnahme ist es, die Chance des Verlustes von Authentifizierungsinformationen zu reduzieren. Multi-Faktor-Authentifizierung verwendet dazu die Faktoren Wissen, Besitz, und biometrische Eigenschaften des Benutzers. Dieses Konzept kann aber auch genereller angewandt werden z. B. Eingabe eines Passworts als erste Richtlinie und ein CAPCHA als zweite Richtlinie.

Hauptsächlich schützt Mehr-Faktor-Authentifizierung den Benutzer, bzw. die Information, die durch erfolgreiche Authentifizierung zugänglich gemacht wird. Selbst wenn eine Richtlinie erfolgreich angegriffen wurde, bleiben noch weitere Richtlinien, die der Angreifer überwinden muss. Ein laufender Angriff kann mit dieser Maßnahme alleine nicht gestoppt werden. Eine vorübergehende zusätzliche Richtlinie kann während eines Angriffs aber sehr wohl das Sicherheitsniveau anheben und dem legitimen Benutzer nach wie vor den Zugriff gewähren.

Numerische Änderungen

Eine weitere Art von Maßnahme ist die Änderung von numerischen Werten wie zum Beispiel der Gültigkeitsdauer von bereits erfolgreicher Authentifizierung oder Obergrenzen für nicht erfolgreiche Authentifizierungsversuche pro Zeitraum. Der Aufwand, um eine derart geänderte Richtlinie zu installieren ist für den Service-Anbieter gering, der Angreifer kann eine übernommene Authentifizierungssession weniger lange nutzen oder kann seinen Angriff weniger effizient ausführen und der Benutzer muss sich öfter wieder einloggen oder muss bei fehlgeschlagener Authentifizierung etwas länger warten bis er einen erneuten Authentifizierungsversuch starten kann.

Kombinationen

Wirkungsvolle und umsetzbare Maßnahmen können meist nur durch Kombination der genannten Maßnahmenarten zustande kommen.

2.3. Strategien

Grundsätzlich lassen sie zwei verschiedene Arten von Strategien unterscheiden. Zum Einen gibt es die klassische Möglichkeit, Strategien von Hand zu gestalten und zu implementieren (statische Strategien), zum Anderen können moderne Verfahren des Maschinellen Lernens zum Einsatz kommen (dynamische Strategien).

Statische Strategien

Statische Strategien sind konzeptionell einfacher umsetzbar, haben aber Nachteile in sich verändernden Systemen. Um eine effektive statische Strategie gestalten zu können muss neben der Anwendungsumgebung, der Anwendung selbst und der relevanten Prozesse auch der Angreifer gut bekannt sein. Mit diesen Informationen lassen sich effiziente und effektive Strategien entwickeln, die gut verstanden und folglich gut vorhersehbar sind, daher auch leicht evaluiert und getestet werden können und mit einmaligem Aufwand erstellt werden können. In der Praxis wird sich aber der Angreifer und eventuell auch die Anwendungsumgebung verändern. Um die Effektivität und Effizienz der Strategie aufrecht zu erhalten, muss diese also regelmäßig

nachgebessert werden. Ein Nachbessern ist allerdings in der Regel erst dann möglich, wenn die neue Umgebung bzw. der neue Angreifer bekannt sind. Man ist also immer einen Schritt hinten.

Dynamische Strategien

Dynamische Strategien sind in der Regel komplexer als statische Systeme. Damit sind sie in der Regel schwieriger umsetzbar, können aber den Vorteil haben, dass sie sich auf sich verändernde Systeme einstellen. Ähnlich wie bei statischen Strategien müssen vor dem Design einer dynamischen Strategie Informationen zu Anwendungsumgebung, Anwendung und Prozessen vorhanden sein. Allerdings kann der Angreifer in Grenzen genereller modelliert werden. Umgesetzt werden kann eine solche dynamische Strategie beispielsweise mit der Hilfe von Methoden des Maschinellen Lernens, die im laufenden Betrieb mitlernen. Solch ein System kann auf Veränderungen in der Umgebung und auch auf Änderung des Angreifers besser reagieren als statische Strategien. Der Nachteil daraus ist, dass nicht sichergestellt werden kann, dass ein neuer Angriff auch wirklich effektiv und effizient gekontert werden kann und das System an sich schwieriger zu evaluieren ist.

Kombinationen

In der Praxis wird wohl immer eine Kombination aus statischen und dynamischen Methoden sinnvoll sein. Damit können Vorteile aus beiden Welten übernommen werden und Nachteile abgeschwächt oder vermieden werden.

2.4. Richtlinien

Es kann prinzipiell drei verschiedene Arten von Richtlinien geben - statische, skalierbare und dynamische. Diese Typen werden nachfolgend diskutiert und mit einer kurzen Diskussion von speziell für dieses Projekt relevanten Richtlinien - Authentifizierungsmaßnahmen - abgerundet.

Statische Richtlinien

Statische Richtlinien werden einmalig per Hand gestaltet. Damit können diese konzeptbedingt nur auf sehr eingeschränkte Anwendungsfälle angewendet werden. Der klare Vorteil solcher Richtlinien ist ihre klare Definition und das damit einhergehende Verständnis der Richtlinie, ihrer Eigenschaften und damit ihrer Effektivität. In aktuellen Systemen finden sich fast ausschließlich solch statische Richtlinien. Beispielsweise muss ein Paar aus Benutzername und Passwort angegeben werden oder ein PIN-Code wird abgefragt.

Skalierbare Richtlinien

Die Sicherheit und Wirksamkeit von skalierbare Richtlinien kann beliebig angepasst werden. Damit können skalierbare Richtlinien als Vorstufe zu rein dynamischen Richtlinien gesehen werden. Skalierbare Richtlinien lassen sich genauso nur auf ein eingeschränktes Set an Anwendungsfällen gestalten und anwenden. Eine klare Definition von Verhalten und Reaktion und damit Effektivität und Sicherheit ist ebenfalls gegeben. In der Praxis finden sich keine klaren Beispiele für skalierbare Richtlinien. Einer der Gründe ist wohl dass sich bisher der Aufwand für solch eine skalierbare Richtlinie nicht lohnt und dem Endbenutzer gleich und jedes mal eine hochwertigere und für den Benutzer potentiell aufwändigere Richtlinie präsentiert wird. Ein einfaches Beispiel für solch eine skalierbare Richtlinie könnte ein Passwort variabler Länge sein. Im Normalfall werden

nur die ersten z.B. 6 Buchstaben des Passworts abgefragt, im Angriffsfall alle 14. Nach jedem fehlgeschlagenen Versuch wird ein Buchstabe mehr abgefragt.

Dynamische Richtlinien

Die Idee hinter dynamischen Richtlinien ist es, aus bestehenden statischen oder skalierbaren Richtlinien neue Richtlinien zu generieren, die für einen Anwendungsfall angebracht sind. Wieder liegt der Fokus darauf, dem Benutzer nur bei Bedarf (z. B. Angriff) komplexe und aufwändige Richtlinien zu präsentieren. Methodiken, um aus bestehenden Richtlinien vollautomatisch sinnvolle und effektive neue Richtlinien abzuleiten sind Gegenstand von Forschung und noch nicht praxistauglich.

3. Umgebungsmodell

Damit eine Strategie Entscheidungen treffen kann, braucht sie Informationen über die Umgebung. Die Umgebung wird meist in Modellen abgebildet liefert neben Informationen zum Zustand der Umgebung auch Benachrichtigungen bei Änderungen des Zustands. In diesem Abschnitt wird dieses Thema nur soweit behandelt, wie es zum Verständnis notwendig ist. Details kann der interessierte Leser in einschlägiger Literatur z.B. zu Angriffserkennungssystemen (IDS) und Intrusion Prevention Systemen (IPS) [6] finden.

In diesem Abschnitt werden zum Verständnis notwendige Details zu Sensorikmöglichkeiten gegeben, die dem Modell als Schnittstelle zur Realität dienen. Anschließend werden exemplarische Methoden beschrieben, wie ein Modell Ereignisse erkennen kann und schließlich wird kurz diskutiert, welche Zustände ein von einem Modell abgebildetes System haben kann.

3.1. Sensoren

Die Abbildung der Realität in einem Modell braucht Sensoren, die Veränderungen im realen System an das Modell weitergeben. Nur so kann eine dynamische Anpassung des Modells passieren. Hierzu gibt es aufwändige Mechanismen die aber meist in einfach verständliche Teile zerlegt werden können. In diesem Abschnitt werden nun grundlegende Mechanismen diskutiert, die für ein grundlegendes Verständnis notwendig sind.

Generell verwenden Sensoren physikalische Eigenschaften eines Systems und beobachten Frequenz von Ereignissen und deren Intensität. Für die Informationstechnologie lassen sich ähnliche Eigenschaften beobachten. Beispiele dafür sind Systemressourcen wie Netzwerkauslastung, Prozessor- und Speicherauslastung. Ereignisse werden meist mit der Hilfe von Protokollierungsmechanismen festgehalten und können daher genauso als Informationsquelle für Sensoren dienen.

Für den Anwendungsfall der Schlüsselnutzung können aber noch weitere Informationsquellen hinzugezogen werden, die sich als wertvolle Informationen für Strategien erweisen können. Beispiele dafür sind Informationen zum Client wie z.B. IP-Adressen (um VPN von lokalem Zugriff unterscheiden zu können), die Clientplattform (ein PC hat andere Eigenschaften als ein Mobiltelefon oder eine SmartWatch).

3.2. Ereignistrigger

Damit eine Strategie einen Handlungsbedarf sehen kann muss dieser von außen vorgeschlagen werden. Ob tatsächlich Handlungsbedarf besteht, entscheidet schlussendlich die Strategie selbst.
Die einfachste Variante eines Ereignistriggers ist ein Ereignis, das für sich selbst schon berichtet werden muss. Ein Beispiel dafür kann eine Schlüsselverwendungsanfrage sein. Andere übliche Ereignistrigger funktionieren auf Basis von Schwellwerten. Konkret wird ein Ereignis getriggert nachdem ein Wert des Modells einen Schwellwert überschreitet. Schwellwerte können dabei statisch vorgegeben sein (z.B. 15), eine Vorverarbeitung der Eingangswerte erfordern (ein Inkrement darf nicht größer als 15 sein) oder aber der Schwellwert folgt den Werten des Modells. Abgesehen davon gibt es komplexere Heuristiken die hier aber nicht diskutiert werden.

3.3. Zustände

Modelle sind in der Regel zu komplex um einfache Aussagen über den Zustand eines Systems zu treffen. Es ist daher üblich, eine Sammlung von Werten und Informationen in diskreten Zuständen zusammenzufassen.
Ein System wird als *nominal* bezeichnet, wenn sich Informationen und Werte im zulässigen Rahmen bewegen. Sobald Ereignistrigger ausgelöst werden, befindet sich das System in einem vom Nominalstatus abweichenden Status. Das System wird beispielsweise *angegriffen*. Dabei kann unterschieden werden, wie schwerwiegend ein Angriff ist. Unterschieden werden kann hier mit Informationen zur Art der Attacke, der Regelmäßigkeit, von welcher Quelle geht der Angriff aus (SmartPhone, PC, Botnetz, ...), der Wahrscheinlichkeit, dass der Angriff glückt oder auch nach den Auswirkung, die ein erfolgreicher Angriff hätte (z.B. das System ist danach 5 Minuten nicht erreichbar oder hochsensible Daten sind nun öffentlich zugänglich). Demnach kann den Status *Angriff* feiner unterteilt werden.
Systemzustände bilden damit potentiell komplexe Zusammenhänge in einem System auf vergleichsweise einfache und überschaubare Werte ab. Diese Werte können nun einer Strategie als Informationsquelle dienen. Wie granular diese Abbildung erfolgt liegt im der Entscheidung der Systemarchitekten.

4. Vorgeschlagene Architektur

Es gibt prinzipiell zwei Möglichkeiten, wie an die Situation angepasste Schlüsselverwendungs-richtlinien implementiert werden können. Die erste Möglichkeit ist trivial: Die Richtlinien werden dabei so gestaltet, dass sie sämtliche Informationen des Modells bereits berücksichtigen. Der Vorteil besteht darin, dass keine aufwändigen Heuristiken, Strategien und Maßnahmen erstellt werden müssen. Der Nachteil ist, dass die Richtlinien statisch sind und bei jeder Anpassung der Richtlinien wieder das gesamte System in Betracht gezogen werden muss. Die zweite Variante verändert die Quelle der Richtlinien so, dass diese jeweils genau für die vorherrschende Situation und Umgebung passen. Vorteile sind neben der besseren Kapselung von Funktionalität auch eine weit überschaubarere Menge an aktiven Richtlinien. Als Nachteil kann die Funktionalität gesehen werden, die benötigt wird, um Richtlinien, Maßnahmen und Strategien bereitzustellen. Da diese

Bausteine aber wiederverwendet werden können, fällt dieser Nachteil nicht so sehr ins Gewicht. Aufgrund der eben genannten Eigenschaften folgt unser Ansatz der zweiten Variante.

Abbildung 1↓ zeigt unseren Ansatz um an die Situation angepasste Schlüsselverwendungsrichtlinien zu implementieren. In einer allgemeineren Betrachtung kann damit jede beliebige Ressource so geschützt werden, dass die Komplexität der Authentifizierung für den Benutzer den situationsabhängig wenigst-mögliche Aufwand erzeugt.

Im Detail wird ein Benutzer (*User*) immer eine Art von Schnittstelle, eine *API*, benutzen, die ihm eine *Ressource* zur Verfügung stellt. Die Art der *Ressource* ist dabei völlig unerheblich. Beispielsweise können neben einfachen Dateizugriffen auch kryptografische Schlüsseldaten sowie ganze Services geschützt werden. Beschützt wird eine *Ressource* durch das *Policy Enforcement*.

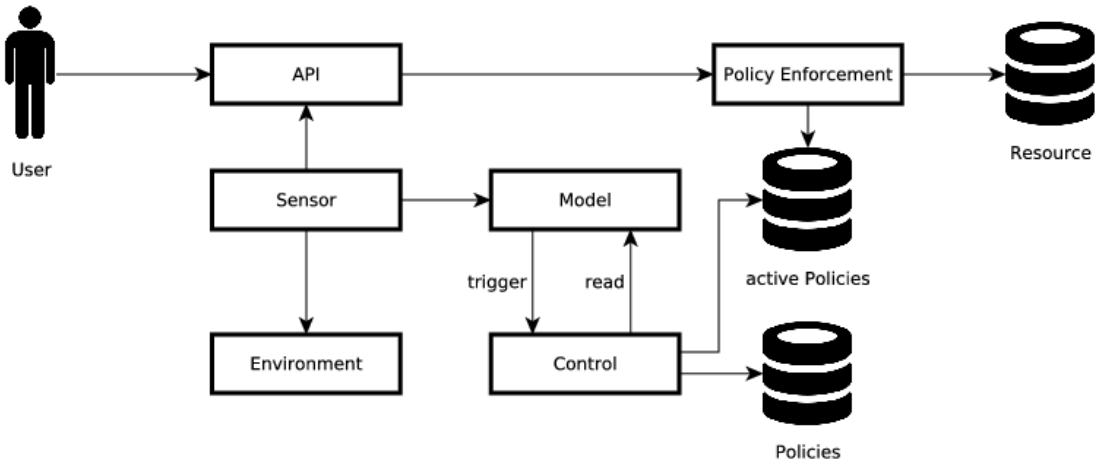

Abbildung 1 Architektur

Das *Policy Enforcement* entscheidet ob eine Anfrage rechtmäßig ist, ob weitere Informationen (Authentifizierungsdaten) benötigt werden oder ob die Anfrage abgewiesen wird. Die Entscheidung trifft das *Policy Enforcement* auf Basis von *Policies (Richtlinien).* Klassische Systeme verwenden oft ein von Hand gepflegtes Set von *Richtlinien*.

In der vorgeschlagenen Architektur übernimmt ein weiterer Block, *Control*, die Kontrolle über die *aktiven Richtlinien*. *Control* kann je nach Systemstatus, Strategie und Maßnahme aus einem Set von *Richtlinien* gezielt *Richtlinien* aktivieren und deaktivieren. Der Systemstatus wird dabei von einem Modell, dem *Model*, abgebildet. Das Modell kann *Control* von Ereignissen in Kenntnis setzen und bietet *Control* ein Informationsinterface an durch das *Control* den Systemstatus abfragen kann um damit Strategien und Maßnahmen zu wählen. Eine Zahl von *Sensoren* versorgen das Modell mit Informationen aus der realen Umgebung (dem *Environment*), der *API* und anderen Informationsquellen. Mit diesen Informationen kann das Modell der Realität folgen.

Mit dieser Architektur wird es möglich, unabhängig von der Art der Ressource und auch unabhängig vom Policy Enforcement eine Ressource adäquat zu schützen ohne dabei den Benutzer unnötig komplexe Authentifizierungsmethoden zu präsentieren.

5. Konkrete Umsetzungen

Im Zuge dieser Arbeit wurden beide in Abschnitt 4↑ angesprochenen Methoden implementiert. Beide Prototypen sind sehr einfach gehalten, um einen klaren Blick auf die Idee und das Konzept zu ermöglichen.

Beide Prototypen implementieren dynamische Schlüsselverwendungsrichtlinien bei Verwendung von Skytrust [2, 1] für kryptografische Operationen. Im Detail will der Benutzer eine kryptografische Operation z.B. „signieren“ ausführen mit einem Schlüssel, der auf einem Skytrust-Service liegt. Nur nach erfolgreicher Authentifizierung am Skytrust-Service wird die angeforderte Operation ausgeführt. Welche Authentifizierung benötigt wird hängt primär vom ausgewählten kryptografischen Schlüssel ab. Die Ziel ist es nun, von einem Benutzer entsprechend der Situation eine ausreichende aber einfachst-mögliche Authentifizierung zur Freischaltung einer kryptografischen Operation zu verlangen.

Folgend werden die beiden Prototypen im Detail beschrieben. Der erste Prototyp demonstriert eine automatische Unterscheidung zwischen Zugriff aus dem lokalen Intranet und dem Zugriff über ein VPN. Der zweite Prototyp demonstriert eine Reaktion auf einen Brute-Force Angriff auf ein Passwort.

5.1. Lokaler Zugriff/VPN

Dieser Prototyp demonstriert eine automatische Unterscheidung zwischen einem Zugriff auf den Skytrust-Service aus dem lokalen Intranet und dem Zugriff über VPN. Ziel ist es, dem Benutzer für die verschiedenen Zugriffswege (lokal, remote) unterschiedliche Authentifizierungsanforderungen zu präsentieren. Konkret soll zu Demonstrationszwecken der Zugriff aus dem lokalen Intranet mit einem einfachen Passwort geschützt werden, der Zugriff über VPN soll mit einer Zwei-Faktor-Authentifzierung erfolgen.

Die Entscheidung woher die Anfrage kommt, wird anhand der Quell-IP-Adresse getroffen. Eine entsprechende Konfiguration auf Netzwerkebene wird vorausgesetzt. Mit dieser einfachen Unterscheidung können potentiell noch weitere Anwendungsfälle abgedeckt werden wie zum Beispiel Zugriff per Kabel oder Zugriff per WLAN. Dieser Prototyp beschränkt sich aber auf zwei Optionen.

Umgesetzt wurde dieser Prototyp mit einer vereinfachten Architektur (siehe Abbildung 2↓).

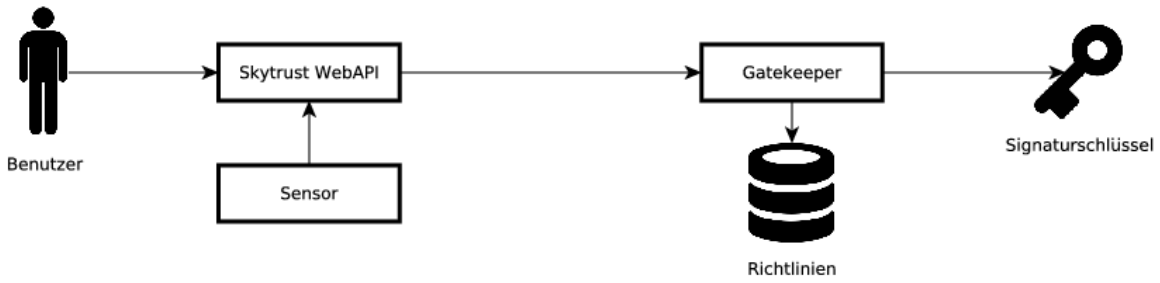

Abbildung 2 Architektur des ersten Prototypen

Der *Benutzer* verwendet die *Skytrust WebAPI* um seine Signaturerstellungsanfrage abzusetzen. Ein *Sensor* extrahiert aus der Anfrage die Quell-IP und schreibt die Daten in den Skytrust

Protokollheader. Das gesamte Protokollpaket wird nun an den *Gatekeeper* weitergegeben. Der *Gatekeeper* ist im Skytrust-System jene Komponente, die entscheidet, ob eine Anfrage rechtens ist und schützt damit die Ressource (in diesem Fall den *Signaturschlüssel*). Der *Gatekeeper* fordert nun auf Basis der *Richtlinien* eine Authentifizierung vom *Benutzer*. Nach erfolgreicher Authentifizierung wird das Protokollpaket an die Signaturerstellungseinheit weitergegeben, dort mit Hilfe des *Signaturschlüssels* bearbeitet und der *Benutzer* erhält das Ergebnis zurück.
Um diesen einfachen Anwendungsfall abzubilden werden lediglich Richtlinien verwendet und auf Maßnahmen und Strategien verzichtet. Die Richtlinien sind so konstruiert, dass sie bereits alle nötigen Informationen beinhalten um auf unterschiedliche IP-Bereiche zu passen oder eben nicht. Für eine Ressource (den Signaturschlüssel) und zwei IP-Bereiche reichen also zwei Richtlinien - eine Richtlinie spricht an wenn der Zugriff aus dem lokalen Intranet erfolgt und fordert wenn aktiv ein Passwort, die zweite Richtlinie spricht an wenn der Zugriff über VPN erfolgt und fordert wenn aktiv eine Zwei-Faktor-Authentifizierung. Hier können beliebig viele IP-Bereiche ergänzt werden und so die Granularität von Bereich und Authentifizierungsmethoden verfeinert werden.
Problematisch wird dieser Ansatz sobald mehr als eine Ressource, also mehr als ein einziger Signaturschlüssel, für eine Vielzahl von verschiedenen IP-Bereichen geschützt werden soll. Wo die Wartung von zwei Policies noch überschaubar ist, ist die manuelle Wartung von Richtlinien für mittelständische Unternehmen, wo jeder Mitarbeiter über einen Signaturschlüssel und einen Verschlüsselungsschlüssel verfügt, weit komplexer und durch die Ähnlichkeit der Richtlinien auch fehleranfälliger. Entsprechend hoch ist der Aufwand auch, wenn eine Änderung der Richtlinien durchgeführt werden muss.

5.2. Brute-Force Attacke auf ein Passwort

Dieser Prototyp demonstriert die automatische Reaktion auf eine Brute-Force Attacke auf ein Passwort, das einen von einem Skytrust-Service gehaltenen Signaturschlüssel schützt. Ziel ist es dem Benutzer nur dann komplexere und aufwendigere Authentifizierungsanforderungen zu präsentieren, wenn das System angegriffen wird. Im Nominalstatus werden z.B. nur die ersten 5 Zeichen des Passworts benötigt. Nach jedem fehlgeschlagenen Authentifizierungsversuch soll die Länge des benötigten Passworts vergrößert werden. Wenn die volle Länge des Passworts bereits erreicht ist soll stattdessen eine Zwei-Faktor-Authentifizierung eingesetzt werden.
Per Definition von oben wird das System als angegriffen betrachtet, sobald eine Authentifizierungversuch fehlschlägt. Mit dieser einfachen Heuristik lässt sich das System gut demonstrieren. Es können aber natürlich auch mächtigere Heuristiken eingesetzt werden wie sie beispielsweise in Intrusion-Detection-Systemen (IPS) verwendet werden, oder es lassen sich Intrusion-Detection-Systeme direkt als Angriffssensor verwenden.
Die konkrete Umsetzung folgt der in Abschnitt 4↑ vorgestellten Architektur (siehe Abbildung 3↓).

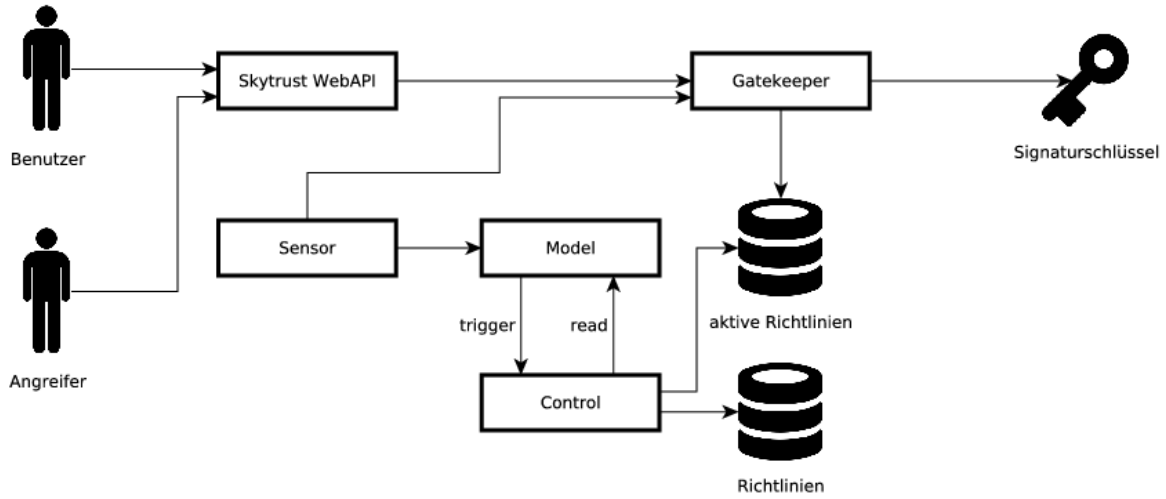

Abbildung 3 Architektur des zweiten Prototypen

Auch hier verwendet der *Benutzer* die *Skytrust WebAPI* um den gewünschten Signaturerstellungsbefehl abzusetzen. Allerdings verwendet auch der *Angreifer* dieselbe API und versucht unerlaubterweise mit dem selben *Signaturschlüssel* Daten zu signieren. Die Anfragen werden in beiden Fällen an den *Gatekeeper* weitergegeben. Der *Gatekeeper* gibt entsprechend der in diesem Fall einen *aktiven Richtlinie* die Authentifizierungsanforderung nach den ersten 5 Stellen des Passworts aus. Der *Angreifer* kann diese nicht auf Anhieb erraten und beantwortet die Authentifizierungsanfrage falsch. Ein *Sensor* meldet den fehlgeschlagenen Authentifizierungsversuch an das *Modell* und der *Controller* ersetzt die *aktive Richtlinie* mit einer, die die ersten 6 Stellen des Passworts abfragt. Diese Spiel wiederholt sich solange, bis schließlich die vergleichsweise sehr aufwändige Zwei-Faktor-Authentifizierung aktiv wird oder aber eine Authentifizierungsanfrage korrekt beantwortet wird und damit das System in den Nominalstatus zurückgesetzt wird.

Ein großer Vorteil dieser Implementierung ist, dass weder der Gatekeeper geändert werden muss, noch dass sämtliche Informationen zum Angriff und dessen Detektion in den Richtlinien enthalten sein müssen und damit gewartet werden müssen. Die Strategie ist in diesem Fall recht einfach: es wird solange eine technologisch gleichwertige aber minimal andere Richtlinie als Maßnahme eingesetzt bis die Maximallänge des Passworts erreicht ist. Dann wir auf eine mächtigere Richtlinie eskaliert. Die Strategie wird vom Controller umgesetzt, der seine Entscheidungsgrundlagen aus dem Modell bezieht. Damit bekommt das Skytrust-System an sich von Strategien und Maßnahmen nichts mit sondern agiert wie gehabt auf Basis von Richtlinien (die von der Strategie vorausgewählt werden). Damit wird dem Benutzer nur dann eine komplexere Authentifizierung vorgeschrieben, wenn diese auch notwendig ist.

6. Zusammenfassung

Ziel des Projekts war es Möglichkeiten für dynamische Verwendungsrichtlinien für Ressourcen zu erkunden. Im Speziellen sollten dynamische Ansätze gefunden werden, wie laufende Angriffe gestoppt werden können ohne gleichzeitig dem rechtmäßigen Nutzer den Zugriff zu verwehren.

Die Arbeit beschäftigt sich zunächst mit einer Generalisierung eines solchen Anwendungsfalls. Dabei werden als konzeptionelle Komponenten Controller, Modelle und Sensoren zu den Basiskomponenten eines Systems identifiziert. Der Controller bedient sich dreierlei Ebenen, auf denen er auf das System Einfluss nehmen kann. Die niedrigste Ebene bilden die Richtlinien selbst, die für einen sehr konkretes Szenario Anwendung finden. Weiters werden Maßnahmen identifiziert, die mit Hilfe von Richtlinien in der Lage sein können, auf Systemänderungen (z.B. Angriff) zu reagieren. Und schließlich werden Strategien identifiziert, die Maßnahmen willkürlich einsetzen können, um unerwünschten Systemänderungen entgegen zu wirken. Systeminformationen bekommt der Controller vom Modell, welches wiederum mit Hilfe von Sensoren der Realität folgt.

In diesem Dokument werden auch zwei Prototypen vorgestellt, die das System an praktischen Anwendungsfällen erproben. Zum Einen wird je nach Herkunft der Anfrage (lokales Intranet oder VPN) eine andere Richtlinie aktiv und fordert der Herkunft entsprechende Authentifizierung, zum Anderen wird demonstriert, wie das System einen Brute-Force Angriff auf ein Passwort entgegenwirken kann ohne dass das Basissystem verändert werden muss. Beide Prototypen unterstützen so den vorgeschlagenen Ansatz.

Diese Arbeit dient mehr als Sprungbrett für zukünftige Tätigkeiten. Im Rahmen dieses Projekts sollte nur ein Überblick über Varianten und Möglichkeiten gewonnen werden, wie denn das oben genannte Ziel erreicht werden kann. Die Prototypen haben bewiesen, dass in der Praxis Anwendungsfälle existieren und das Thema auch wissenschaftlich interessant und relevant ist. Der Sourcecode der Prototypen ist auf github einsehbar[1].

Alles in allem ergibt sich so ein System, das flexibel und leicht für ähnliche Anwendungsfälle anpassbar ist. Durch die klare Kapselung der Funktionalität muss an Zielsystemen sehr wenig geändert werden um die Vorteile der dynamischen Verwendungsrichtlinien zu nutzen.

7. References

[1] Florian Reimair, Peter Teufl, Thomas Zefferer. CrySIL: Bringing Crypto to the Modern User. In *Lecture Notes in Business Information Processing* . Springer, in press.

[2] Florian Reimair, Peter Teufl, Thomas Zefferer. WebCrySIL - Web Cryptographic Service Interoperability Layer. *Proceedings of Web Information Systems and Technologies*:35-44, 2015.

[3] Thanasis Petsas, Giorgos Tsirantonakis, Elias Athanasopoulos, Sotiris Ioannidis. Two-factor Authentication: Is the World Ready?: Quantifying 2FA Adoption. Proceedings of the Eighth European Workshop on System Security:4, 2015. URL https://dx.doi.org/10.1145/2751323.2751327.

[1] https://github.com/IAIK/CrySIL

[4] Catherine S. Weir, Gary Douglas, Tim Richardson, Mervyn Jack. Usable Security: User Preferences for Authentication Methods in eBanking and the Effects of Experience. Interact. Comput., 22(3):153—164, 2010. URL https://dx.doi.org/10.1016/j.intcom.2009.10.001.

[5] William E. Burr, Donna F. Dodson, Elaine M. Newton, Ray A. Perlner, W. Timothy Polk, Sarbari Gupta, Ebad A. Nabbus. Electronic Authentication Guideline. 2013. URL https://dx.doi.org/10.6028/NIST.SP.800-63-2.

[6] Karen Scarfone, Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). 2007. URL https://dx.doi.org/10.6028/NIST.SP.800-63-2.