DynFire - Verteilte Firewalls in heterogenen Netzwerken

Sören Berger
Universität Stuttgart
IZUS/TIK Netze und Kommunikationssysteme
Allmandring 30
70550 Stuttgart
soeren.berger@izus.uni-stuttgart.de

Alexander Vensmer
Universität Stuttgart
Institut für Kommunikationsnetze und Rechnersysteme
Pfaffenwaldring 47
70569 Stuttgart
alexander.vensmer@ikr.uni-stuttgart.de

Zusammenfassung

Dieser Beitrag präsentiert "DynFire", einen neuen Ansatz um Firewalls dynamisch und benutzerspezifisch zu konfigurieren. Durch DynFire kann in einem Netz kontrolliert werden, welche Nutzer zu welchen Diensten Verbindungen aufbauen dürfen. Dies wird durch neukonfigurieren der sich im Netz befindlichen Firewalls realisiert. Dabei ist grundlegende Annahme von DynFire, dass in einem vom Internet getrennten und streng kontrolliertem Netzwerk stets eine Zuordnung zwischen IP Adresse und Benutzerkennung existiert. Immer wenn sich ein Benutzer authentifiziert, werden die entsprechenden Firewalls freigeschaltet bzw. geschlossen. Dies wird mit einer zentralen Firewall-Manager Instanz und standardisierten Signalisierungsprotokollen realisiert.

1 Einleitung

Firewalls gelten als sicheres Mittel um IP-Netze voneinander abzuschotten und vor Eindringlingen zu schützen. Eine Grundannahme ist, dass sich Netzwerke in verschiedene Zonen mit unterschiedlichen Sicherheitsanforderungen einteilen lassen. Die Firewalls sind an entsprechenden Zonengrenzen platziert, um den Netzverkehr in die entsprechende Zone

entweder zuzulassen oder zu blockieren. Diese Sicherheitsregeln sind typischerweise fest in den Firewalls konfiguriert und lassen sich nur mit Aufwand und vor allem nicht dynamisch ändern.
Mit der Verbreitung von Smartphones, Tablet PCs und anderen mobilen Endgeräten sowie der Nutzung von VPNs über das Internet werden diese harten Sicherheitsgrenzen immer weiter aufgeweicht. Ein solches Endgerät bekommt bei der Einwahl in das Netz dynamisch eine neue IP-Adresse zugewiesen. Somit beinhalten die IP Pakete nicht genug Informationen, um es Firewalls zu ermöglichen benutzerspezifische Entscheidungen zu treffen.

Als Betreiber eines größeren Campusnetzwerkes stellen wir eine steigende Anzahl an Geräten fest, die nicht in die Kategorie der klassischen Endgeräte fallen, wie zum Beispiel Telefone oder Arbeitsplatz-Rechner. Geräte wie Messstationen oder die Gebäudeleittechnik werden immer häufiger in aktuelle Kommunikationsnetze integriert. Sie sind aber oft auf einem veralteten Sicherheitsstand (altes Betriebssystem, fehlender oder alter Virenscanner oder fest einprogrammierte Passwörter). Auf der anderen Seite sollen sie von außerhalb zugänglich sein, z.B. für Gebäudetechniker. Techniken wie beispielsweise VPN erlauben es einfach solchen externen Nutzern Zugriff auf diese Ressourcen zu gewähren, allerdings ermöglicht man so auch den Zugriff auf andere Netzteilnehmer. Der entsprechende Ansatz, eine Firewall vor dem entsprechende Gerät zu platzieren, erhöht zwar die Sicherheit des Systems, allerdings ist dieses Prinzip nicht flexibel genug, um Zugriffsberechtigungen auf der Basis von Nutzern zu erteilen. Lösungen, die Nutzern stets dieselbe IP-Adresse zu vergeben, können zu einer Adressknappheit führen und sind mit einem hohen Administrationsaufwand verbunden und stossen damit schnell an ihre Grenzen.

Dieser Beitrag stellt DynFire vor, eine Architektur um dynamisch und benutzerbasiert Firewalls in einem heterogenen Netz zu konfigurieren. Neben der Erläuterung der verschiedenen Konzepte und Komponenten werden auch erste Erfahrungen über eine Erprobung im Betrieb und eine Leistungsbewertungen vorgestellt.

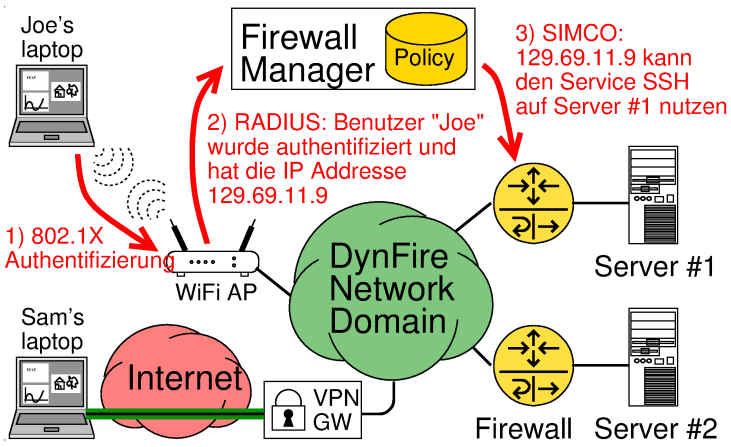

Abbildung 1: DynFire Szenario

2 Anforderungen

Die Hauptanforderung an DynFire ist es Zugriff auf Dienste im Netz nutzerbasiert freizuschalten. Dabei sollen Firewalls genutzt werden, ohne das es eine Unterstützung seitens der Endgeräte bedarf. Daher kann DynFire keine Lösung für das gesamte Internet sein, sondern ist als Absicherung von IT-Ressourcen einzelner Organisationen gedacht. Es wird angenommen, dass die jeweiligen Organisationen Firewalls im Einsatz haben, um ihr Netz gegen Gefahren aus dem Internet zu schützen. DynFire soll dabei in der Lage sein möglichst viele verschiedene Firewalls unterschiedlicher Hersteller zu unterstützen. DynFire stellt die Anforderung an das Netz, dass sich jeder Nutzer zunächst authentifiziert, bevor er Zugriff auf aus das betreffende Netz bekommt. Es wird angenommen, dass diese Authentifizierung nicht umgangen werden kann. Technisch kann dies durch den Einsatz von Protokollen wie zum Beispiel 802.1x/802.1ae (LAN oder WiFi) oder mittels eines zentralen VPN Konzentrators (Abbildung 1) realisiert werden. DynFire stellt nur sicher, dass autorisierte Nutzer die Möglichkeit haben, Kommunikationsverbindungen zu jeweiligen Ressourcen zu erstellen. Natürlich können und sollen zusätzlich anwendungsspezifische Schutzmaßnahmen verwendet werden um eine Ende-zu-Ende Sicherheit zu garantieren. Eine verteilte Administration mit der Möglichkeit zur Delegation der Netzverantwortung, stellt eine weitere Anforderung an DynFire dar. Administratoren, die Netzressourcen für Nutzer oder Gruppen freigeben wollen, erstellen allerdings keine Firewall-Regeln, sondern können durch ein DynFire-spezifisches Policy-Framework komfortabel abstrakte Policies formulieren. Diese Policies beinhalten Bedingungen, die an den Nutzer gestellt werden um Zugriff auf das Netzwerk zu erhalten. Die Bedingungen können verschiedenartiger Ausprägung sein. So ist es möglich eine bestimmte Gruppenzugehörigkeit zu fordern und auch den Zeitraum einzuschränken in der der Benutzer auf die Ressourcen zugreift. Auch ist es Administratoren möglich, Nutzern Attribute zuzuweisen, wie zum Beispiel die Authenifizierungsmethode. Wenn sich ein Nutzer anmeldet, werden diese abstrakten Policies von einem Firewall Manager in explizite Firewall-Regeln umgewandelt und auf die Firewalls mit Hilfe eines Provisioners übertragen. DynFire soll dabei auch mit komplexen Typologien umgehen und diese selber erfassen können. DynFire soll jederzeit berechnen können, welche und wie viel Firewalls umkonfiguriert werden müssen, damit ein Kommunikationskanal zwischen Nutzer und dem jeweiligen Dienst aufgebaut werden kann.

3 Architektur & Implementierung

3.1 Firewall Manager

Der Firewall-Manager ist die zentrale Komponente in DynFire. Er verwaltet eingeloggte Benutzer, die Netztopologie, die definierten Regeln und das Verteilen der Firewall-Regeln auf die Firewalls. Die Funktionalität des Firewall-Managers ist auf die vier Module Policy Framework, Policy Controller, Topology Awareness und Firewall Provisioner aufgeteilt, und wird anhand dieser noch detailliert beschrieben. Die benötigten Daten, werden in vier verschiedenen Datenbanken gehalten und synchronisiert:

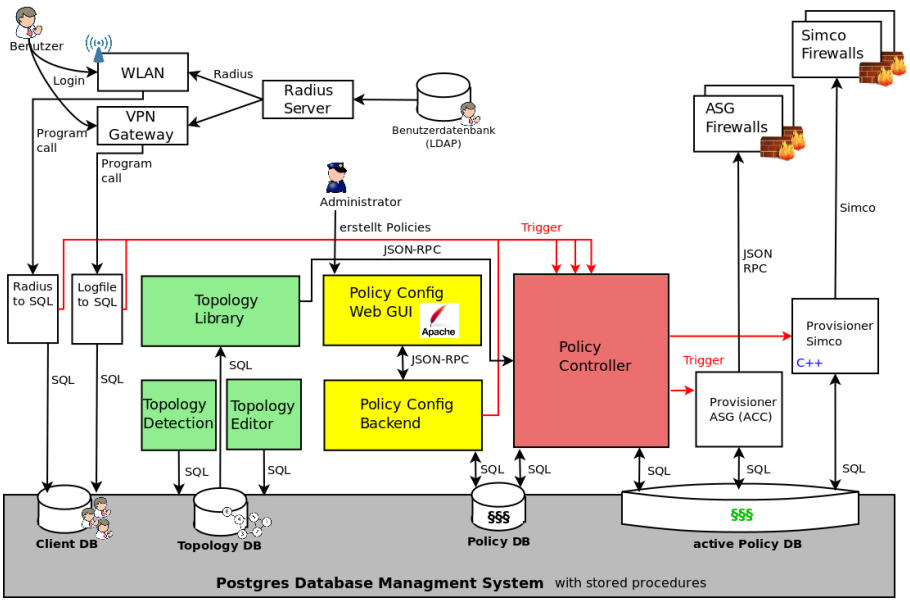

Abbildung 2: DynFire Architektur

client DB Diese Datenbank beinhaltet Benutzer die aktuell eingeloggt sind. Nutzer die sich anmelden, werden mit so genannten Loginscripten durch das VPN Gateway oder WLAN Authentifizierungssystem in die Datenbank eingetragen. Es lassen sich außer 802.1x noch andere Authentifizierungsmethoden realisieren. Die Minimalanforderung ist, dass diese den Benutzernamen und die zugewiesene IP Adresse des Benutzers bereitstellen.

Policy DB Die Policy DB beinhaltet alle abstrakten Policies, die von Administratoren in das System eingetragen werden. Diese stellt eine statische Sicht auf den kompletten Regelsatz dar und beinhaltet welcher Nutzer auf welche Ressourcen unter welchen Bedingungen zugreifen dürfte, falls er sich anmelden würde. Das zugrunde liegende Datenbankmodell wird im Unterkapitel “Policy Framework” vorgestellt.

activePolicy DB Die Datenbank mit dem Namen activePolicy DB beinhaltet alle aktuell auf Firewall zu verteilenden und verteilten Firewall-Regeln. Sie stellt eine dynamische Sicht der Daten des Systems dar und gibt Information über den aktuellen Zustand des Systems inklusive der Firewalls. Sie dient zur Synchronisation zwischen dem Policy Controller und den Firewall Provisionern. Dafür wird ein Soll und Ist Zustand der Firewalls gespeichert. Der Sollzustand wird vom Policy Controller gesetzt. Die entsprechenden Firewall Provisioner stellen diesen Zustand durch die Eintragung oder die entsprechende Entfernung der Regeln her.

Topology DB Die Netztopologie wird in der Topology DB gehalten. Das Netz wird als Graph abgebildet. Dabei stellen die Knoten, Router und Accessnetze dar und werden durch Kanten, welches dann die korrespondierenden Links sind, miteinander verbunden. An den Links können dann die Firewalls spezifiziert werden. Es ist zum einen möglich die Topologie von Hand einzupflegen oder die Netztopologie direkt aus der Netzinfrastruktur bzw. von einem Netzwerk Management System zu importieren.

Die Gesamtarchitektur von DynFire ist in Abbildung 2 dargestellt. Die bereits genannten Module werden durch Trigger synchronisiert und die Module überprüfen darauf Änderungen in den entsprechenden Datenbanken.

3.1.1 Policy Framework

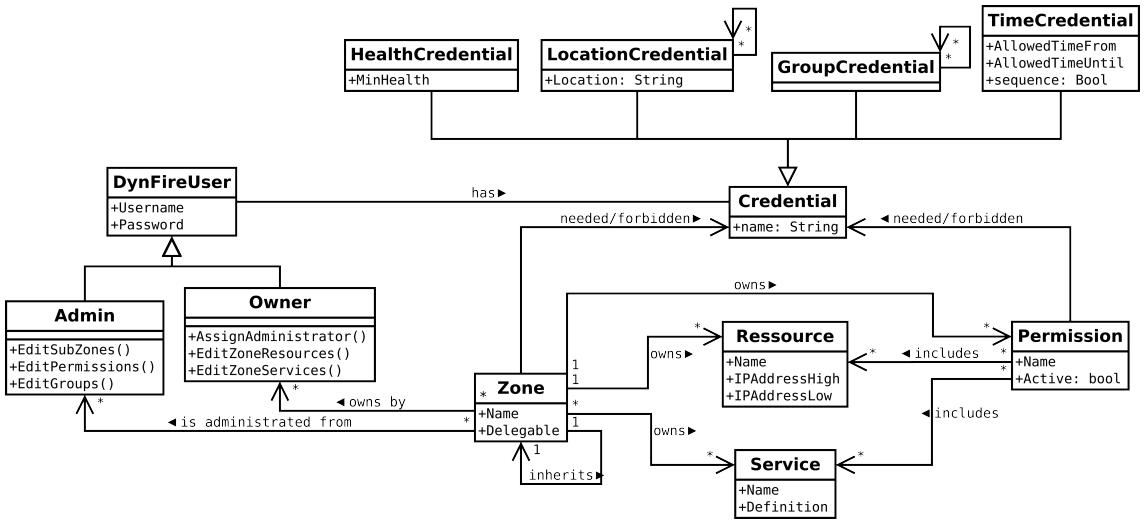

Abbildung 3: DynFire Policy Framework

Die Aufgabe des Policy Frameworks ist die Verwaltung des kompletten DynFire Regelsatzes. Dabei gibt es die Möglichkeit die Verantwortung durch Delegation auf eine Vielzahl von Administratoren aufzuteilen. Der Regelsatz besteht aus Policies, zusammengesetzt aus Permissions, Credentials, Ressourcen und dem jeweiligen Services. Dabei realisiert diese Zusammensetzung die Anforderung nach welcher Administratoren formulieren können sollen, welche Dienste ein Nutzer unter welchen Bedingungen nutzen kann. Diese Policies können durch einen Algorithmus zu Firewall-Regeln im 5-tuple Format (Protokoll, Quelladresse, Quellport, Zieladresse, Zielport) konvertiert werden.

Eine vereinfachte Version der Datenstruktur des Policy Frameworks ist in Abbildung 3 dargestellt.

Services Ein Service entspricht im einfachsten Falle einem Tupel aus Quell- und Zielports unabhängig von den entsprechenden Quell- und Zieladressen. Es können auch Mengen von Ports ein. So wird zum Beispiel der Zugriff auf einen HTTP-Server durch ein Tupel (Quellport = any, Zielport = 80) realisiert. Die Spezifikation ist nicht an Ports gebunden, sondern kann auch abstrakter spezifiziert werden. Der entsprechende Ausdruck von dem dazugehörigen Firewall Provisioner entsprechend interpretiert werden. So gibt es die Möglichkeit Mengen von Port-Tupeln oder Regeln für eine inhaltsabhängige Firewall zu verwendet werden, wie es zum Beispiel ein Dienst wie SIP nötig ist.

Ressourcen Ressourcen stellen eine Menge von IP Adressen da. Sie können hierarchisch aufgeteilt und von unterschiedlichen Administratoren verwendet werden. An oberster Stelle

steht die Root-Zone, die den IP-Bereich des kompletten Netzes enthält. Diese kann dann von einem Administrator aufgeteilt und an weitere Administratoren weiterdelegiert werden. Daraus ergibt sich ein hierarchisches Administrationssystem für das gesamte Netz. In Verbindung mit den Services werden somit die Ziele für Benutzer definiert.

Credentials Credentials sind Einschränkungen um Benutzer mit Hilfe von Attributen Zugriff auf die Netzressourcen zu gewähren oder zu verbieten. Durch eine eingeführte Vererbungsstruktur ist es leicht möglich weitere Credential Arten zu erstellen. Eines der realisierten Typen ist die Gruppenzugehörigkeit, die einem Nutzer zugewiesen werden kann. Ein weiteres ist die Zugangsart, also durch welche Technik der Nutzer Zugang zum Netz erhalten hat. Hier kann man beispielsweise unterscheiden, ob der Benutzer sich vom VPN oder über das WLAN eingeloggt hat und dies bei der Erstellung der Policies berücksichtigen. Weitere Credentials sind der Sicherheitszustand des Clients oder der Zeitpunkt des Einloggens. Diese verschiedenen Arten von Credentials werden von Administratoren verwendet um auszudrücken, welche Credentials ein Nutzer benötigt, damit eine Permission für ihn gilt. Ein Teil der Credentials erhält der Nutzer dadurch, dass ein Administrator ihm eine bestimmte Gruppenzugehörigkeit zuweist. So kann ein Nutzer beispielsweise der Gruppe „Router-Admin“ zugewiesen werden um das Recht zu vergeben, auf Router zu zugreifen. Den anderen Teil der Credentials werden dem Nutzer während des Logins zugewiesen. Ein Beispiel hierfür ist auf welche Art der Nutzer Zugang zum Netz bekommen hat.

Permissions Die Permission verbindet Ressource und Service, mit den notwendigen Credentials. Damit eine Permission für einen Nutzer aktiviert wird, darf dem Nutzer zum Einen kein Credential zugewiesen worden sein welches für die Permission verboten ist, andererseits müssen diesem alle notwendigen Credentials zugeordnet sein.

3.1.2 Policy Controller

Der Policy Controller ist die zentral Komponente des Firewall Managers. Er berechnet zur Laufzeit aus den Policies im Policy Framework die abstrakten Firewall-Regeln für die Firewall Provisioner. Während des Evaluierungsprozesses wertet er, mit Hilfe der Topology Awareness, die Firewalls aus, auf die die entsprechenden Regeln eingetragen werden müssen. So können durch eine Policy mehrere abstrakte Firewall-Regeln entstehen, falls zum Beispiel mehrere Firewalls auf dem Pfad zwischen Nutzer und Zielressource liegen.
Er besitzt auch eine Schnittstelle für das Datenbankbackend des Policy Frameworks bereit, um dem Policy Config Web GUI eine einheitliche Schnittstelle zur Datenbank bereitzustellen. Dieses Interface ist mandantenfähig und kann von den verschiedenen Administratoren der Netze verwendet werden um Regeln zu erstellen, Benutzergruppen zu verwalten oder Administrationsbereiche zu delegieren.

3.1.3 Topology Awareness Modul

Das Topology Awareness Modul ist dafür zuständig alle Firewalls zu finden, die auf einem Pfad zwischen zwei bestimmten Netzeilnehmer liegen. In dem zugrunde liegenden Datenbankmo-

dell kann ein Layer 3 Graph aus Routern und Links definiert werden. Die Links können entweder zwischen Routern oder Netzen definiert werden, die jeweils ein Access Netz spezifizieren. Auf solch einem Link werden dann die Firewalls definiert, die zwischen den Routern oder Netzen platziert sind. Diese Definition beinhaltet alle Informationen für die Firewall Provisioner um später Regeln in die Firewall einzutragen.

Auch ist Topology Awareness Modul dafür verantwortlich die oben bereits beschriebene Datenbank Topology DB zu füllen. Dafür wird ein Interface bereitgestellt, mit welchem dies manuell passieren kann, was dann nur den Betrieb einer statischen Topologie ermöglicht und nicht auf Änderungen reagieren kann. Zusätzlich gibt es die Möglichkeit die Topologie automatisch aus dem Netz oder von einem Netzwerk Management System importiert wird. Die kann zum Beispiel durch Benutzung des LLDP (Link Layer Discovery Protocol) [1], CDP (Cisco Discovery Protocol) oder dem SNMP (Simple Network Management Protocol) [2] geschehen.

3.1.4 Firewall Provisioner Modul

Das Firewall Provisioning Modul transferiert die erzeugten Firewall Regel auf die von dem Topology Awareness Module bestimmten Firewalls. Da es keine einheitliche Schnittstelle auf Firewalls gibt um Firewall-Regeln einzusetzen, kommt an dieser Stelle das SIMCO Protokoll [3] zum Einsatz. SIMCO ist ein einfaches Client-Server Protokoll, bei dem der SIMCO Client Firewall-Regeln an den SIMCO Server schickt. Diese Regeln werden dann entsprechend eingetragen. Da nicht alle Firewallhersteller SIMCO nativ unterstützen wurden verschiedene SIMCO Server für Cisco, Linux IP Tables und Juniper implementiert, die sich teilweise erheblich unterscheiden. Da beispielsweise Juniper ein BSD ähnliches Betriebssystem verwendet, kann hier ein Server direkt auf dem Router implementiert werden. Cisco Router dagegen sehen solche Programme nicht vor, so dass die Regeln von dem Server auf einen separaten Rechner mittels einer CLI Schnittstelle oder SNMP übertragen werden müssen.

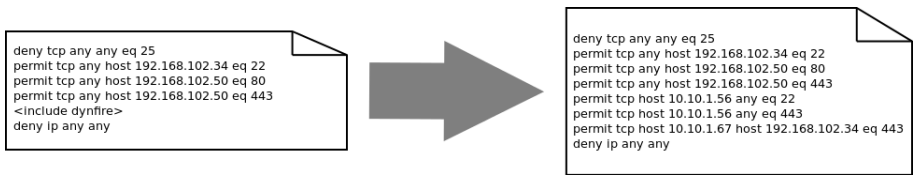

Abbildung 4: Auflösung der Firewallregeln

Je nach Implementierung müssen auch die existierenden Regelsätze und statischen Konfigurationen berücksichtigt werden. Dafür werden die DynFire spezifischen Firewall-Regeln an einer wohldefinierten Stelle im Regelsatz der Firewall eingetragen. In Abbildung 4 ist dieses Prinzip für die Cisco Firewall dargestellt. Die DynFire Regeln werden an der Stelle <include dynfire> eingefügt. Ganz anders in der Implementierung für Linux IP Tables wo eine eigene Chain innerhalb der IP Tables verwendet werden kann.

3.2 Gesamtprozess

Abbildung 5 zeigt den gesamten Prozess bei einem Login eines Benutzers. Nachdem der Benutzer sich erfolgreich authentifiziert und angemeldet hat, wird seine IP Adresse bestimmt und zusammen mit dem Benutzernamen in die client DB eingetragen. Hier werden auch Attribute wie zum Beispiel der Ort der Authentifizierung bestimmt und in Form von Credentials dem Nutzer zugewiesen. Nun wird der Policy Controller über ein RPC Schnittstelle aktiviert. Da während eines Loginzyklus mehrere neue Benutzer hinzukommen können, müssen eventuell mehrere Benutzer vom Policy Controller abgearbeitet werden. Für jeden Benutzer wird der Policy Evaluierungsprozess gestartet und alle abstrakten Policies aus der Policy DB bestimmt. Danach werden die Firewalls im Netz mit Hilfe der Topology Awareness berechnet. Daraufhin werden die Firewall Provisioner Modul aktiviert. Dies transportiert die Firewall-Regeln

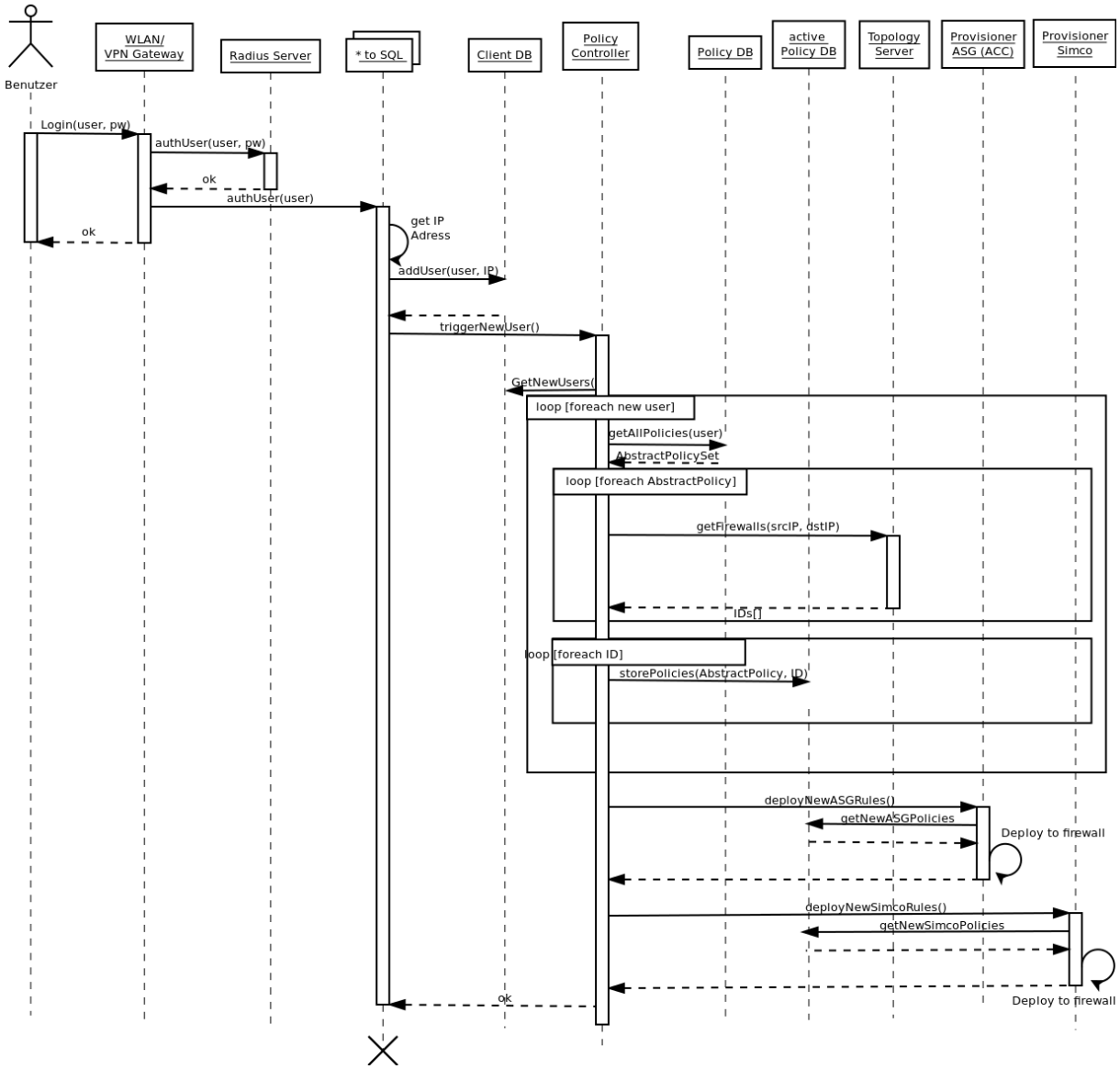

Abbildung 5: DynFire Benutzer Login Sequenzdiagramm

auf die jeweiligen Firewalls um dort von den Firewalltyp abhängigen Implementierungen der

betroffenen Firewall-Regeln eingetragen. Die Freischaltungen stehen nun zur Verfügung und der Benutzer kann auf die entsprechenden Netzressourcen zugreifen.

Falls sich der Benutzer ausloggt oder sich ein Benutzer mit der gleichen IP Adresse einloggt (was implizit ein Logout des alten Benutzers darstellt), wird der Policy Controller erneut von dem Loginprozess getriggert. Da jeder Loginprozess einer eindeutigen ID zugeordnet ist, müssen die Regeln nicht neu berechnet werden, sondern können direkt in der activePolicy DB deaktiviert werden. Die Firewall Provisioner entfernen dann die entsprechenden Freischaltungen.

4 Leistungsbewertunug

Im Rahmen der Leistungsbewertung von DynFire wurden mögliche Engpässe des Systems identifiziert und näher untersucht. Aus diesen Untersuchungen entwickelten sich grundlegende System-Designentscheidungen und Parameter. Eine identifizierte und untersuchte Komponente sind die Firewalls selbst. Als Referenzen sind eine Linux Firewall und eine Cisco Firewall zum Einsatz gekommen.

4.1 Scenario

Um abschätzen zu können, mit welcher Menge an Daten und Dynamik das System und die Firewalls umgehen können und welche Anforderungen die Komponenten des Systems erfüllen müssen, sind aus einem anonymisiertem Log-File des DHCP-Servers der Universität Stuttgart verschiedene Werte abgeleitet worden.

Da Mitarbeiter von der Zahl der an DynFire teilnehmenden Nutzer den bei weitem größten Anteil darstellen, sind diese Werte für diese auch noch einmal gesondert betrachtet worden.

Extrahierte Werte sind unter anderem die Login-Rate beziehungsweise Logout-Rate, welche beide bei circa 8 Vorgängen/Sekunde zur Mittagszeit liegen, falls man Mitarbeiter und auch Studenten berücksichtigt. Diese Login-Rate kann im Falle einer Ausnahmesituation wie beispielsweise der kurzfristige Ausfall eines Wireless LAN Controllers, auf den Wert 50 Vorgängen/Sekunde steigen. Da diese Situationen mit einer Häufigkeit von circa 1-mal pro Woche auftreten, muss dieses vom System gehandhabt werden können. Auch die maximale Anzahl gleichzeitig angemeldeter Nutzer ergab sich aus dieser Analyse und liegt bei ca. 1200 Nutzern bzw. bei alleiniger Betrachtung der Mitarbeiter bei circa 250 Nutzern. Des Weiteren wird im Folgenden angenommen, dass ein durchschnittlicher DynFire Nutzer 10 Dienste benutzen darf, was also in 10 Permissions und folglich 10 Firewall-Regeln resultiert, die für diesen eingesetzt werden müssen. Eine Anforderung die an das System formuliert wurde war, dass der zeitliche Anteil des Eintragens von Firewall-Regeln nicht über 30 Sekunden steigt

4.2 Firewalls

4.2.1 Messungen

Ein wichtiges Leistungsmerkmal der Firewalls im Rahmen von DynFire stellt die Zeitdauer dar, welche benötigt wird um Firewall-Regeln unter verschiedenen Umständen zu erstellen und entfernen. Die Abbildung 6 zeigt, wie viele Millisekunden benötigt werden, eine Firewall-Regel, in Abhängigkeit der schon eingeschriebenen Regeln auf der Firewall einzutragen. Zusätzlich ist dargestellt, wie lange es dauert eine Regel zu entfernen. Beide Vorgänge dauern etwa gleich lang, so dass ein Unterschied nicht messbar ist. Bei dieser Messung kam ein Rechner mit Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GHz und 8192MB Arbeitsspeicher zum Einsatz. Das Linux Betriebssystem verwendete einen Kernel der Version 2.6. Wobei bei einer Linux Firewall die Einschreibe-Dauer linear mit der bereits gespeicherten Anzahl an Firewall-Regeln ist, ist diese bei den Cisco-Firewalls unabhängig von der bereits gespeicherten Anzahl an Firewall-Regeln. Allerdings ist die Einschreibe-Dauer relativ hoch. So benötigt der Cisco Catalyst 3750 etwa 10 Sekunden, der Cisco Catalyst 6500 benötigt etwa 3 Sekunden. Der Cisco Catalyst 3750 gilt als Standardgerät im LAN Accessbereich und ist somit grundsätzlich nicht als Router sondern als Accesswitch konzipiert. Allerdings hat er auch Firewallfunktionalität und kann Cisco ACLs, also Firewall-Regeln verwalten. Aufgrund seiner beschränken Ressourcen ist es allerdings nur möglich maximal 5000 Regeln zu verwalten. Der Cisco Catalyst 6500 hat im Gegensatz zum Cisco Catalyst 3750 höhere Limits. Weitere Messungen haben auch gezeigt, dass die Performance der Firewall, was das Weiterleiten an momentanem Verkehr betrifft, bei modernen Mehrkern-Architekturen, wie sie auch im Rahmen von DynFire zum Einsatz kommen, nicht durch Ein- und Austragen von weiteren Regeln beeinflusst wird.

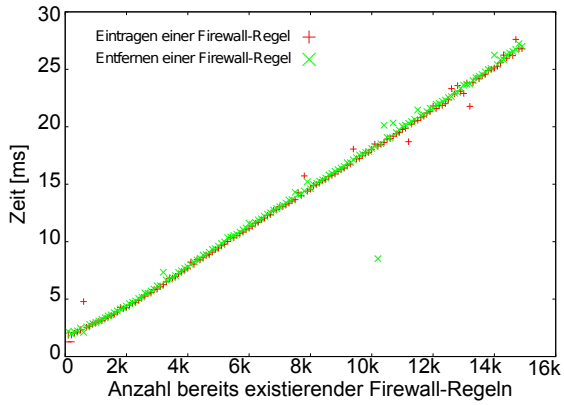

Abbildung 6: Verarbeitung einzelner FW-Regeln - Linux Firewall

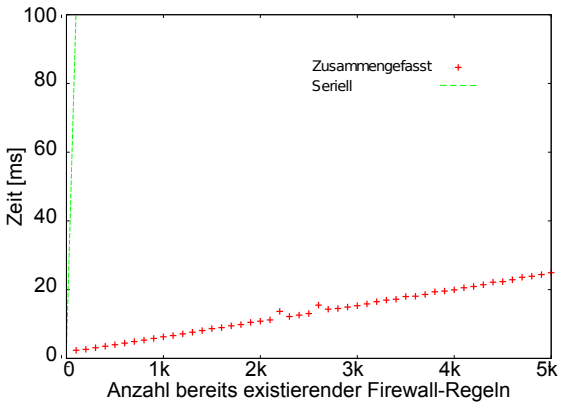

Abbildung 7: Verarbeitung zusammengefaster FW-Regeln - Linux Firewall

4.2.2 Erste Abschätzung

Für den Fall, dass eine einzelne Regel nach Transport durch SIMCO sofort in den Regelsatz eingeschrieben wird, kann leicht durch eine Worst-case Abschätzung festgestellt werden, dass in diesem Fall die Firewall-Regeln unabhängig vom Typ nicht in ausreichender Zeit abgearbeitet werden können. Worst-case Abschätzung meint in diesem Fall, dass alle Firewall-Regeln

die verteilt werden müssen, sich auf eine Firewall konzentrieren. Die oben genannte Linux Firewall, könnte theoretisch die oben bereits erwähnten 8 Logins/Sekunde, aus welchen 80 Firewall-Regeln resultieren noch seriell handhaben. In Abbildung 6 ist zu erkennen, dass eine Bearbeitungszeit von ca. 2 Millisekunden pro Regel benötigt wird. Daher es können ca. 500 Firewall-Regeln Einträge pro Sekunde bearbeitet werden. Eine Cisco-Firewall vom Typ Catalyst 3750 benötigt circa 10 Sekunden um eine Firewall-Regel zu verarbeiten und wäre damit nicht in der Lage die Anzahl an Regeln auf diese einfache Art und Weise zu verarbeiten. Zieht man zusätzlich noch in Betracht, dass es in oben bereits genannten Ausnahmesituationen Loginraten bis zu 50 Logins pro Sekunde (im Durchschnitt 500 Firewall-Regeln) geben kann, erkennt man das eine andere Strategie beim Eintragen der Regeln verfolgt werden muss, da dies auch die vom Regeleintragen her schnellere Linux Firewall nicht mehr schaffen würde.

4.2.3 Gewinn durch Zusammenfassen von Regeln

Messungen haben gezeigt, dass Zusammenfassen von Regeln und gemeinsames Eintragen effizienter, daher schneller passiert, als dies einzeln der Fall ist. Bild 7 stellt das Ergebnis einer solchen Messung für die Linux Firewall dar. Es wurden jeweils in 100 Schritten Firewall-Regeln in eine leere Firewall eingetragen. Dabei kann der zeitliche Gewinn als Differenz zwischen der Geraden, welche die Zeitdauer für das serielle Einschreiben abbildet, mit der Messung direkt abgelesen werden. Verwendet wurde hierbei das Tool iptables-restore. Der Test in Abbildung 9 zeigt den Zeitverlauf beim Deployment von 0-5000 Regeln für die Cisco Firewall Catalyst 3750. Dabei wurden die Regeln in 100er Schritten eingetragen. Bei ca. 1000 Regeln beträgt die Zeit zum Einsetzen des Regelsatzes bereits ca. 30 Sekunden. Auffällig ist hier die unstetige Zeit ab ca. 2500 Regeln, was auf die beschränkten Ressourcen hindeutet. In Abbildung 8 ist die Zeit über der einzuschreibenden Anzahl von Firewall-Regeln, für die Cisco Catalyst 6500 Firewall, aufgetragen.

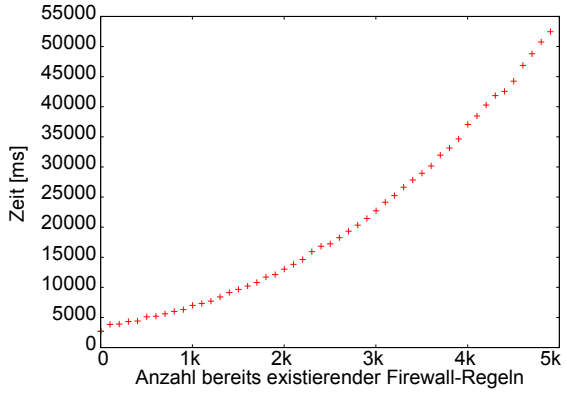

Abbildung 8: Regeleintragung Cisco Catalyst 6500

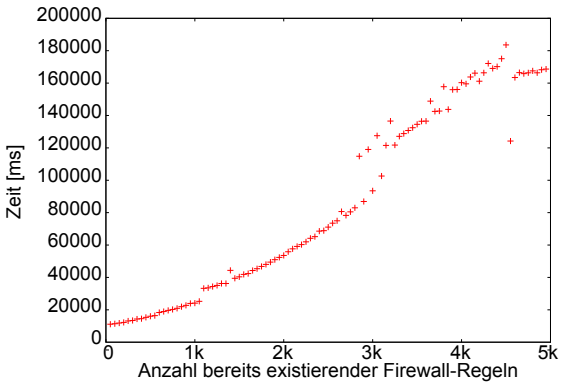

Abbildung 9: Regeleintragung Cisco Catalyst 3750

Aufgrund dieser Messungen und der Tatsache, dass es im Falle der betrachteten Cisco Firewalls keine effektive Möglichkeit gibt, mehrere Firewall-Regeln auf einmal zu entfernen, ist eine andere Herangehensweise untersucht worden. Dabei wird eine bestimmte Zeit gewartet und ein aktueller und vorgehaltener Regelsatz wird, nach dem Löschen des alten Regelsatzes, auf einmal eingesetzt. Dies hat den Vorteil das Firewall-Regeln nicht aus dem sich Einsatz

befindlichen Regelsatz gelöscht werden muss, sondern nur in der vorgehaltenen Liste. Daher verwendet DynFire die Strategie eine konstante Zeit zu warten und die vollständig sich ergebene aktualisierte Liste von Firewall-Regeln einzutragen. Bei Berechnungen zu der Linux Firewall auf Basis der DHCP-Daten hat sich gezeigt, dass bereits das Warten von einer Sekunde dazu führt, dass sich auch in Ausnahmesituationen alle anfallenden Firewall-Regeln ohne Aufstauung solcher bearbeitet werden können. Eine noch detaillierte Analyse, dahingehend die Wartezeit zu verkürzen, war hier nicht möglich, da die DHCP-Logs nur Sekunden genau vorlagen. Bei der Analyse dieser Strategie im Falle der Cisco Firewall einer Catalyst 6500 hat sich gezeigt, dass die Performance, auch durch Gewinn der sich durch Zusammenfassen von Regeln ergibt, nicht ausreichen würde für jeden Login-Vorgang der Universität 10 Firewall-Regeln einzutragen. Dazu wäre es nötig 12000 Firewall-Regeln einzutragen, dies wäre wegen der genannten Obergrenze von 500 Regeln mit einer Cisco Catalyst 3750 nicht mehr möglich. Bei einer Cisco Catalyst 6500 würde dies die unbefriedigende Zeitdauer von bis zu circa 4 Minuten zur vollständigen Login-Dauer des Nutzer beitragen. Mit einem Catalyst 3750 könnte man, unter der Bedingung das die oben genannte Forderung von 30 Sekunden für das schreiben eines neuen Regelsatz erfüllt wird, 1000 Firewall-Regeln deployen, daher circa Regeln für 100 Nutzer. Im Falle eines Catalyst 6500 wären dies 3500 Firewall-Regeln, daher 350 Nutzer. Geht man von dem realistischen Szenario aus, dass die Anzahl der Studenten für welche dynamische Regeln gesetzt werden vernachlässigbar ist und betrachtet man nur die Mitarbeiter der Universität Stuttgart, so kann man von etwa 2500 Firewall-Regeln ausgehen die eingesetzt werden müssen, was mit einer Cisco Catalyst 6500 Firewall also realisierbar wäre. Unter der zusätzlichen Annahme, dass große Institute etwa 20 Mitarbeiter besitzen und im Schnitt mit noch einmal derselben Anzahl an Mitarbeitern kollaborieren, was einer maximalen Firewall-Regel Anzahl von 400 entsprechen würde, würde also eine Cisco Catalyst 3750 Firewall ausreichen. Für Firewalls die nahe am netzinneren platziert sind, würde dann eine Cisco Catalyst 6500 Firewall oder aber eine Linux Firewall zum Einsatz kommen müssen.

5 Integration und Erprobung

Die Universität Stuttgart betreibt ein Netz mit circa 800 aktiven Komponenten wie Router, Switches oder Access Points. Verteilt auf dieses Netz sind ca. 30000 bekannte IP Geräte, die von dem IP Management System des Rechenzentrums erfasst sind. Nicht berücksichtigt werden hier unter anderem Geräte hinter einem NAT Gateway. Das Netz ist physikalisch als Doppelstern implementiert, wobei die zentralen Komponenten redundant ausgelegt sind. Dies ist allerdings transparent auf für die Layer 3 Infrastruktur. Bild 10 zeigt einen vereinfachten Überblick über das Netz. Neben dem zentralen Corenetz werden circa 25 so genannte (ebenfalls redundante) Gebietsrouter betrieben. Innerhalb dieses Netz wird der gesamte Layer 3 Verkehr der Uni geroutet. Sie bilden auch die Schnittstelle für die Accessswitches. An dieser Stelle sind die Firewalls in sogenannten Cisco Access Control Lists implementiert. Aktuell sind circa 300. ACLs auf den Routern im Einsatz und sollen entsprechend dem DynFire-Ansatz auch dynamisch angepasst werden. Diese sind meistens IP Bereichen von Instituten, Studentenpools oder anderen Sicherheitszonen zugeordnet. Da die Firewalls schon in Betrieb sind und von einem Firewall Management System bereits verwaltet werden, existiert eine Schnittstelle zwischen den beiden Systemen. Es muss also bei einem neuen Deployment sowohl der statische Teil der Firewall Management Lösung als auch die Regeln von DynFire berückschtigt werden.

Da DynFire grundsätzlich über die freigeschaltenen Regeln in der Datenbank Protokoll führt, können diese verhältnismäßig einfach in das Firewall Management System bei der Änderung der statischen Regeln importiert werden. Da DynFire reines Whitelisting verwendet ist die Position der DynFire Firewall-Regeln egal. Allerdings kann es mit der Integration der statischen Firewall-Regeln zu Mehrdeutigkeiten kommen, da diese nicht aus reinem Whitelisting bestehen. Für diesen Zweck ist die Position der DynFire Regeln in dem finalen Regelsatz immer fix und wird mit einem Keyword in das Firewall Management System implementiert. Das Fire-

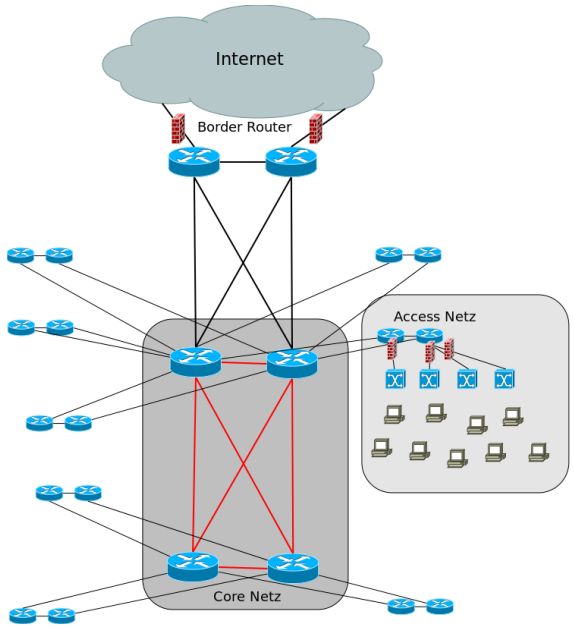

Abbildung 10: Doppelstern Netztopologie

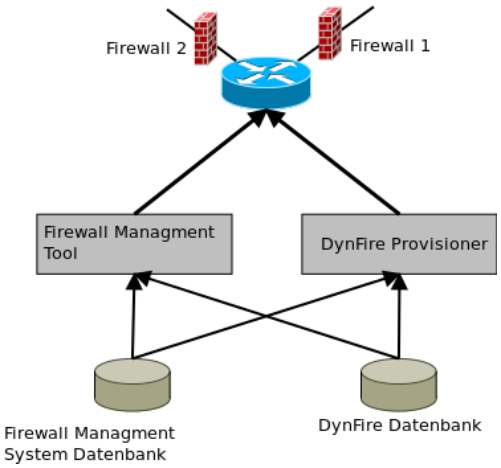

Abbildung 11: Firewall Managment und DynFire Integration

wall Management System besitzt selbst eine Datenbankschnittstelle und kann somit ebenfalls von dem ensprechenden Firewall Provisioner verwendet werden. Dieser wertet die statischen Regeln aus und fügt die entsprechenden DynFire Freischaltungen an den definierten Stelle ein. Somit können DynFire und das Firewall Management System parallel betrieben werden, was für die konsistenz der Daten unabdingbar ist. Das Firewall Management System hällt für jede ACL eine eigene List vor. Es existieren also mehere hundert Listen. Für eine Abbildung der Firewalls in dieses Netz wäre es also nötig für jede Liste einen Firewall Provisioner zu implementierunen und zu starten. Da dies praktisch kaum möglich ist und in ebenso mehrere hundert gestartete Programme resultieren würden, wurden verschiedene Provisioner zusammengefasst und ein Provisioner pro Gebiet verwendet. Zusätzlich werden alle diese Prozesse in einem Programm gestartet. Bild 11 zeigt die verschiedenen Datenquellen und Programme, die auf die entsprechende Firewall zugreifen. Danke der einheitlichen Datenbankschnittstelle, ist die Implementierung und Integration wesentlich einfacher. Die Lösung selbst deckt mehr als 98% der Firewalls der Universität Stuttgart ab. Somit wird bsi jetzt nur ein Firewall Provisioner verwendet. Die anderen Provisioner stehen bereit und es wird erwartet, dass neben dem Cisco Provisioner auch der Simco Provisioner für IP Tables verwendet wird.

Bei der Integration und Erprobung wurde in einem sogenannten Testgebiet begonnen. Dies hat nur sehr wenige Netze, die auch keine Produktionumgebung darstellen. Der Provisioner selbst wurde nur für dieses Gebiet aktiviert um mögliche Signalisierungsfehler in andere Gebiete zu verhindern. Erst nach einigen Testiterationen und insbesondere der experimentellen Veri-

fikation, dass DynFire die entsprechenden Firewalls nicht schliesst, konnte auf den anderen Routern mit den Tests fortgefahren werden. Grunsätzlich muss DynFire aktuell immernoch im Provisioner für die entsprechenden Netze aktiviert werden um im unerwarteten Fehlerfall keine Netze zu beinflussen, die nicht an DynFire beteiligt sind.

6 Verwandte Arbeiten

Es gibt eine Vielzahl an Arbeiten, die eine vollständige Architektur mit dem Ziel, definierte Policies in Form von Firewall-Regeln oder anderen Filtermechanismen in einem Netz zu etablieren, beinhalten. Die Architektur SANE [4] ermöglicht es nutzerbezogen Verbindungen zu Diensten im Netz erstellen zu können, allerdings werden hier, für die Realisierung, Änderungen des IP Protokolls vorgeschlagen und bedürfen daher einer Anpassung aller Netzkomponenten unter anderem der Endgeräte. Eine Kompatibilität zu IP-Netzen kann durch so genannte “Translation Proxies” erreicht werden, was allerdings zu steigender Anzahl zusätzlicher Geräte führt. DynFire stellt eine Lösung dar, die weder der Unterstützung der Endgeräte, noch eine Änderung der IP-Protokoll Architektur bedarf. Während DynFire einen proaktiven Ansatz beim Einschreiben neuer Firewall Regeln verfolgt, wird dieses in der Netzarchitektur Ethane [5] von Martin Casado reaktiv gehandhabt. Diese Netzarchitektur nutzt dazu einen OpenFlow ähnlichen Ansatz. Die Policies der Nutzer werden zentral gespeichert, falls so genannte Ethane-Switches einen Verbindungsaufbau beobachten, welcher nicht schon deren FlowTable vorkommt, fragt diese eine zentrale Instanz nach der weiteren Handhabung. Allerdings werden die Ethane-Switches nicht aufgrund eines Logins eines Nutzers am Netz neu konfiguriert, sondern jedes Mal wenn eine neue Verbindung (Flow) des Nutzers beobachtet wird. Erst daraufhin konfiguriert ein Controller, eine zentrale Instanz, die entsprechenden weiteren Ethane-Switches auf dem Pfad oder gibt Anweisung die Pakete zu verwerfen. Guha und Francis [6] schlagen eine Lösung für das komplette Internet vor, wobei sie dabei Unterstützung von den entsprechenden Endgeräten benötigen. DynFire zielt nur auf die Nutzung in geschlossenen Benutzergruppen und kontrollierten Netzen ab. Da nur in solchen, die notwendige Zuordnung von IP Adresse und Nutzer ID sichergestellt werden kann. Cisco Systems’ TrustSec Technology [7] kann eine so genannte “downloadable Access Control Lists” (dACL) auf eine Firewall laden, wenn der Benutzer sich an das Netzwerk anmeldet. Dies ist allerdings eine proprietäre Lösung.

Auch wurden schon Teilaspekte der Anforderungen von DynFire näher betrachtet, so wurde die dynamische Anpassung von Firewalls detailliert für VoIP Anwendungen untersucht [8, 9, 10]. Hier wurden, aufgrund von Mediastömen, die durch das Signalisierungsprotokoll SIP ausgehandelt wurden, die Firewalls in einem geschlossenem Netz dementsprechend konfiguriert, dass neu ausgehandelte Verbindungen zustande kommen konnten. DynFire hingegen ist flexibel für jedes Protokoll verwendbar und macht hier keine Einschränkungen.

Im Bereich der Formulierungen von Policies und Firewall-Regeln existieren bereits Lösungen, wobei alle aus unterschiedlichen Gründen keinen Einsatz in DynFire gefunden haben. Laborde et al. [11] verwenden ein sehr allgemeines und theoretisches RBAC Modell um ein policybasiertes Netzwerkmanagement (PBMN) System zu spezifizieren. Allerdings existiert dies nur ein einer theoretischen Form die Implementierung erscheint durch die Allgemeinheit

sehr komplex. Basile et al. [12] haben ein ontologybasierendes Policy System für Netzwerke spezifiziert und implementiert. Allerdings geht dies von einer zentralen Administrationsschnittstelle aus und berücksichtigt weder die Delegation von Administrationsrechten noch dynamische Logins von Benutzer. Ein Benutzer hätte also somit einen festen Arbeitsplatz und eine feste IP Adresse, was also den Anforderungen von DynFire widersprechen würde. Generelle Modelle für die Modellierung von Zugriffen werden rollen-basiert (RBAC) von Ferraiolo and Kuhn [13] vorgeschlagen. Attribut-basierte Zugriffsrechte können mit einem ABAC Framework realisiert werden. Diese werden z.B. für Webservices von Yuan and Tong [14] vorgestellt. Zhang et al. [15] definieren eine high-level Sprache um Firewall-Regeln zu beschreiben und Konflikte zwischen den verschiedenen Firewalls aufzulösen. Grundsätzlich ist diese Möglichkeit der Regelbeschreibung sehr mächtig. DynFire ermöglicht allerdings nicht verbietende Regeln, da dies die Administration - insbesondere für Teilnetzadministratoren – schwer verständlich macht. XACML [16] ist eine sehr verbreitete und ausgereifte Spezifikationssprache für Policies und wurde während der Designphase sehr gründlich untersucht. Allerdings deckt sie nur einen Teil der Anforderungen von DynFire ab. Unter anderem, wird beispielsweise nicht die Delegation von Administrationsrechten berücksichtigt. Des Weiteren wird in XACML die entsprechende Regel erst beim Zugriff evaluiert, während bei DynFire die Regel bereits beim Login und nicht beim Zugriff auf die Netzwerkressource erzeugt. Somit passt die Architektur von XACML nicht zu DynFire.

7 Zusammenfassung

Wir haben das Design sowie die Implementierung für DynFire, einer Architektur zur dynamischen Konfiguration von Firewalls, vorgestellt. Diese erlaubt es Netzressourcen nutzerbasiert freizugeben. Es wird ermöglicht vor allem existierende und Embedded Geräte mit unzureichendem Sicherheitsstand, sicher in einem heterogen Netz ohne die Erstellung von einer Vielzahl von VPN, zu nutzen. Dazu benötigt DynFire keine spezielle Hardware. Es nutzt ausschließlich Firewalls welche in nahezu allen Netzen bereits vorhanden sind sowie einen zentralen Linux-Server als Firewall-Manager. Während der letzten Phase der Implementierung werden wir DynFire evaluieren und auf Skalierbarkeit hinsichtlich verschiedener Parameter untersuchen. Anschließend ist der vollständige Einsatz im Wirkbetrieb im Netz der Universität Stuttgart geplant.

Danksagung

Diese Arbeit wurde durch das DynFire-Projekt unterstützt, einem vom Deutschen Bundesministerium für Bildung und Forschung (BMBF) gefördertes Forschungsprojekt (Förderkennzeichen: 01BY1151). Die in diesem Dokument dargestellten Ansichten, Ergebnisse und Schlussfolgerungen sind die der Autoren und stimmen daher nicht notwendigerweise vollständig mit den offiziellen Positionen und Richtlinien des DynFire-Projekts oder des BMBF überein.

Literatur

[1] IEEE LAN/MAN Standards Committee, “Station and Media Access Control Connectivity Discovery,” IEEE, Std. 802.1ab, 2009.

[2] J. Schoenwaelder and T. Jeffree, “Simple Network Management Protocol (SNMP) over IEEE 802 Networks,” IETF, RFC 4789, Nov. 2006.

[3] M. Stiemerling, J. Quittek, and C. Cadar, “NEC’s Simple Middlebox Configuration (SIMCO) Protocol V3.0,” IETF, RFC 4540, May 2006.

[4] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, “Sane: a protection architecture for enterprise networks,” in *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS’06. Berkeley, CA, USA: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267336.1267346

[5] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, “Ethane: taking control of the enterprise,” in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM ’07. New York, NY, USA: ACM, 2007, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1282380.1282382

[6] S. Guha and P. Francis, “Towards a Secure Internet Architecture Through Signaling,” Cornell University, Ithaca, NY, Technical Report cul.cis/TR2006-2037, 2006.

[7] Cisco Systems, Inc, “Cisco TrustSec Solution Overview,” 2012.

[8] C. Aoun, “Plan de signalisation Internet pour l’interfonctionnement entre NAT et Firewall,” PhD Thesis, ENST, Paris, 2005.

[9] S. Kiesel and M. Scharf, “Modeling and performance evaluation of transport protocols for firewall control,” *Computer Networks*, vol. 51, no. 11, pp. 3232–3251, Aug. 2007.

[10] ETSI TISPAN, “NGN Functional Architecture,” ETSI, Standard ES 282 001 V3.4.1, 2009.

[11] R. Laborde, M. Kamel, F. Barrère, and A. Benzekri, “Implementation of a formal security policy refinement process in wbem architecture,” *J. Netw. Syst. Manage.*, vol. 15, no. 2, pp. 241–266, Jun. 2007.

[12] C. Basile, A. Lioy, S. Scozzi, and M. Vallini, “Ontology-based policy translation,” in *Computational Intelligence in Security for Information Systems*, ser. Advances in Intelligent and Soft Computing, A. Herrero, P. Gastaldo, R. Zunino, and E. Corchado, Eds. Springer Berlin / Heidelberg, 2009, vol. 63, pp. 117–126.

[13] D. F. Ferraiolo and D. R. Kuhn, “Role-based access controls,” *15th National Computer Security Conference*, pp. 554–563, 1992.

[14] E. Yuan and J. Tong, “Attributed based access control (abac) for web services,” in *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, july 2005, pp. 2 vol. (xxxiii+856).

[15] B. Zhang, E. Al-Shaer, R. Jagadeesan, J. Riely, and C. Pitcher, “Specifications of a high-level conflict-free firewall policy language for multi-domain networks,” in *Proceedings of the 12th ACM symposium on Access control models and technologies*, ser. SACMAT ’07. New York, NY, USA: ACM, 2007, pp. 185–194.

[16] XACML spezification - OASIS eXtensible Access Control Mark-up Language. [Online]. Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml