TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

Professur Verteilte und Selbstorganisierende Rechnersysteme

Praktikumsbericht

Entwurf und Implementierung einer gesicherten Netzwerkumgebung zur Isolation und Bereinigung von infizierten Nutzergeräten

Jakob Döring, Nils Trampel, Lucas Schröder

Chemnitz, den 1. November 2020

Prüfer: Prof. Dr.-Ing. Martin Gaedke
Betreuer: Valentin Siegert

Jakob Döring, Nils Trampel, Lucas Schröder
Entwurf und Implementierung einer gesicherten Netzwerkumgebung zur Isolation und Bereinigung von infizierten Nutzergeräten
Praktikumsbericht, Fakultät für Informatik
Technische Universität Chemnitz, November 2020

Abstract

In Zeiten umfassender Digitalisierung in sämtlichen Lebensbereichen kommt der Sicherheit von Datennetzen größere Bedeutung zu. Deswegen ist es unabdingbar, Geräte, die möglicherweise mit Schadsoftware infiziert sind, zu isolieren und in einer besonderen, zusätzlich gesicherten Umgebung zu untersuchen und unter Umständen die Symptome sowie Ursachen zu beseitigen. Durch zusätzliche Maßnahmen wie Installation von Sicherheitsupdates und Bugfixes kann die Sicherheit des jeweils betroffenen Systems gestärkt und eine Neuinfektion sowie die Gefährdung benachbarter Netzgeräte reduziert werden.

Diese Arbeit entsteht in Zusammenarbeit mit dem Chemnitzer StudentenNetz. Aufgrund von Veränderungen in technischen Gegebenheiten ist die dortige derzeitige Lösung nicht mehr praktikabel. Es soll ein Ersatz erarbeitet, aber gleichzeitig durch Analyse neuer Entwicklungen aus der Forschung und Industrie eine zukunftsträchtige Lösung geschaffen werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1. Projektvision

In Campusnetzwerken muss immer wieder damit umgegangen werden, dass Nutzergeräte mit Viren oder Trojanern infiziert sind und bereinigt werden müssen. Um diese Situation in einem Netzwerk mit nutzeradministrierten Geräten zu adressieren, soll ein Quarantänenetz entstehen, in dem Nutzer ihre Rechner, welche von Viren, Trojanern, o. Ä. befallen sind, updaten können, Antivirenprogramme finden und installieren können. Der Zugriff auf das restliche Netz sollte jedoch zur Verhinderung der Ausbreitung von Schadprogrammen stark eingeschränkt sein.

In der aktuellen Installation wird der Netzzugang mittels eines Proxys beschränkt. Alle ausgehenden Anfragen der sich im Quarantänenetz befindenden Rechner werden über diesen Proxy geleitet und mittels einer Whitelist beschränkt. Ein Zugriff ist so nur noch auf Dienste möglich, die per HTTP oder HTTPS erreichbar sind. Aufgrund der Protokollerweiterungen im Bereich HTTPS ist es für die Nutzer nicht weiter zumutbar und technisch umsetzbar, einen Internetzugriff für HTTPS über einen Proxy zu leiten, wenn die Nutzer ihre Rechner selbst administrieren.

In dieser Arbeit wird dieses Problem gelöst, indem ein Captive Portal installiert wird, über das Nutzer zunächst darüber informiert werden sollen, dass sie sich in einem Quarantänenetz befinden und welche Schritte sie unternehmen müssen, um wieder in das normale Netz zu kommen. Das Netz, in dem sich die Nutzer sonst befinden, soll hierfür nachgestellt werden. Das heißt, dass das genutzte Gateway sowie die IP Konfiguration vollständig erhalten bleiben können. Hierbei soll auch die aktuelle Entwicklung von Standards im Bereich Captive Portals der IETF Working Group capport mit einbezogen werden. Einschlägige RFCs im Bereich Captive Portals sollen mit in die Planung einbezogen und evaluiert werden. Der Zugriff auf bestimmte Netzdienste soll durch eine DNS Whitelist geregelt sein. Hierzu muss sichergestellt werden, dass die Nutzer im Quarantänenetz nur den vom CSN bereitgestellten DNS-Server nutzen. Hierbei gilt es, insbesondere auch aktuelle Entwicklungen im Bereich DNS mit abzudecken. Die zu erreichenden Domains sollen, um eine möglichst einfache Wartbarkeit des Systems zu erreichen, über eine Weboberfläche oder automatisiert mit Ansible eingestellt werden. Eine zusätzliche Zugriffsbeschränkung sollte einfach zu installieren sein, um den Zugriff auf vom DFN gemeldete Angriffsziele komplett zu unterbinden.

Ein zusätzlicher Bereich des Projekts schließt die Überwachung der installierten Lösung ein. Es soll eine Schnittstelle geschaffen werden, mit der überwacht werden kann, zu welchen Zwecken das Quarantänenetz genutzt wird. Hierzu sollen alle Flows, die ihren Ursprung im Quarantänenetz haben, aufgezeichnet werden.

2. Ausgangssituation

In diesem Kapitel wird die Ausgangssituation im CSN dargestellt. Das Kapitel ist hierzu in zwei Teile unterteilt, die zum einen die aktuelle technische Implementierung darstellen und zum anderen den in Quarantänefällen genutzten Prozess. Beide Bereiche werden im Rahmen des Projekts betrachtet und optimiert.

2.1. Technischer Überblick

Das aktuell im CSN installierte Quarantänenetz setzt sich als Netzwerksystem in einer Schichtenarchitektur zusammen und wird im Folgenden unter Zuhilfenahme des OSI-Modells betrachtet:

- Layer-1: Für das Quarantänenetz wird das im CSN installierte Layer-1-Equipment unverändert weitergenutzt. Abbildung 2.1 zeigt, dass keine zusätzliche Netzwerktechnik für das Quarantänenetz installiert ist.

- Layer-2: Wird ein Nutzer in das Quarantänenetz verschoben, so ändert sich die VLAN -Zuordnung am Switchport, mit dem sein Endgerät verbunden ist. In Abbildung 2.1 wird dies durch den Übergang zwischen „Clients Quarantänenetz“ und „Clients Hausnetz“ dargestellt. Diese Konfiguration wird durch ein Skript vorgenommen, welches aus der CSN-Datenbank zuerst abruft, ob ein Nutzer im Quarantänenetz sein soll. Wenn das der Fall ist, wird per SNMP auf dem Switch, mit dem der Nutzer verbunden ist, die VLAN-Zuordnung geändert. Für jedes VLAN, an das Nutzer angeschlossen sind, existiert ein zusätzliches Quarantäne-VLAN im CSN. Hierzu ist es erforderlich, dass beide VLANs an allen Switches verbunden sind, an denen Nutzer hängen. Das dem Nutzer bekannte Gateway ist im Quarantänenetz ein anderes Gerät und besitzt deshalb auch eine andere MAC-Adresse. Diesen Umstand müssen die Geräte des Nutzers erst durch das Versenden von ARP-Anfragen erkennen.

- Layer-3: Im Quarantänenetz herrscht dieselbe Adressierung wie auch im normalen Nutzernetz. Der Nutzer kann also seine IP und Routen behalten, wenn er in das Quarantänenetz verschoben wird. Es ist allerdings weiterhin notwendig und möglich, die genutzte Adresse per DHCP zu beziehen. Da im CSN die Nutzeradressen nicht einzeln, sondern als größere Subnetze (/23 oder /24) geroutet

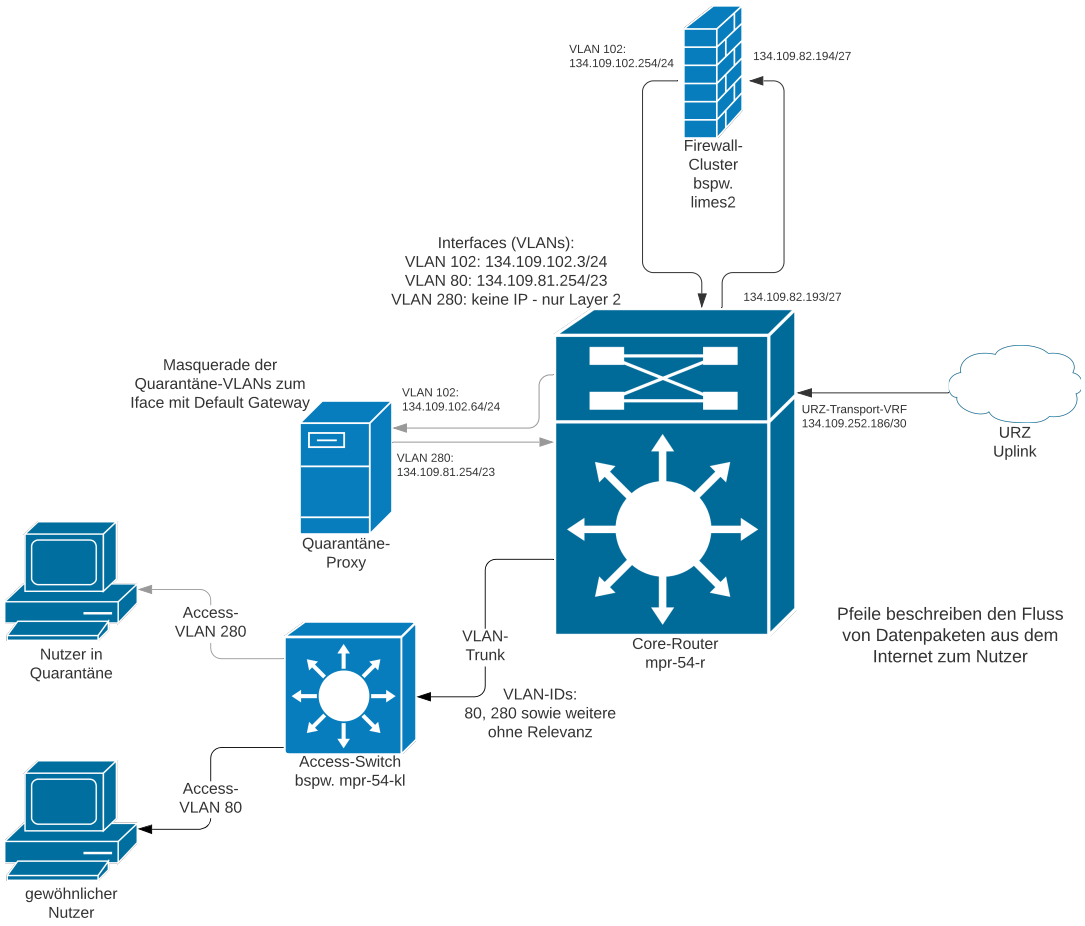

Abbildung 2.1.: Infrastrukturskizze Netzwerk im CSN

werden, muss verhindert werden, dass durch die Adressierung im Quarantänenetz eine Route doppelt vorhanden ist. Dies wird aktuell dadurch gelöst, dass das Gateway im Quarantänenetz nicht der sonst verwendete Router ist und alle im Quarantänenetz befindlichen Geräte hinter seiner IP Adresse mittels Source-NAT versteckt.

- Layer-4: TCP-Verbindungen werden zu einem transparenten Proxy auf dem Gateway im Quarantänenetz umgeleitet. Hierzu wird eine Firewall mit iptables betrieben, die in der NAT-Tabelle und der PREROUTING-Chain alle Verbindungen, die per TCP an Port 443 gehen (meist HTTPS), oder an Port 80 (HTTP) gerichtet sind, an einen Squid-Proxy umleitet. Der Proxy baut dann anstelle des Endgerätes des Nutzers die Verbindung ins Internet auf.

Hierbei kommt es aktuell zu Problemen, da die HTTPS-Verbindungen neu terminiert werden und dafür mit einem selbstsignierten Zertifikat signiert werden. Dies führt zu Warnungen im Browser der Nutzer. Im Enterpriseumfeld werden zur Vermeidung dieser Warnmeldungen die selbst signierten Zertifikate auf die Clientrechner installiert. Das ist im CSN allerdings nicht möglich, da jeder Nutzer seine Geräte selbst administriert.

Als weitere Sicherheitsmaßnahme leitet der Proxy nur Anfragen weiter, die an eine Liste von bestimmten Domains gerichtet sind. In dieser Liste sind aktuell einige Hersteller von Antivirensoftware und einige Updateserver von Microsoft enthalten. Die Updates der Clientrechner sind allerdings häufig nicht möglich, da die automatische Validierung der Zertifikate im Updateprogramm fehlschlägt.

UDP-Verbindungen oder TCP-Verbindungen, die nicht zu Port 80 oder 443 gehen, sind gesperrt.

DNS-Lookups sind nur über das Gateway im Quarantänenetz möglich, auf dem ein separater DNS-Server läuft, der den Nutzern über die DHCP-Option 6 ausgeliefert wird.

2.2. Nichttechnischer Überblick

Abbildung 2.2 zeigt, was im CSN passiert, wenn das DFN durch DPI-Mechanismen feststellt, dass ein System im CSN von Schadprogrammen befallen ist. Das DFN schickt zuerst eine Mail an eine Mailingliste beim CSN. Dies geschieht automatisiert, da beim DFN hinterlegt ist, welche IP-Bereiche dem CSN zugeordnet sind, und welche Mailadresse bei missbräuchlicher Nutzung kontaktiert werden soll.

Die Mail vom DFN wird dann von einem der Administratoren im CSN bearbeitet, indem auf der Webseite zuerst über die in der Mail angegebene IP-Adresse das entsprechende Nutzerprofil gesucht wird, welches zur in der vom DFN angegebenen Zeit die IP Adresse hatte. Der Nutzer wird dann über die Weboberfläche ins Quarantänenetz verschoben und bekommt sofort eine Mail mit Informationen, wie er/sie vorgehen kann, um wieder ins normale Nutzernetz zu kommen. Die Verschiebung in das Quarantäne-VLAN des entsprechenden Hausnetzes passiert binnen 10 Minuten automatisiert.

Der Nutzer muss nun eine aktuelle Antivirensoftware sowie alle Betriebssystemupdates installieren und sich mit einem vollständigen Systemscan bei einem Etagenverantwortlichen melden. Dieser kann auch bei Updates bzw. bei der Bedienung von Antivirensoftware behilflich sein. Wenn der Systemscan ohne Befunde durchläuft,

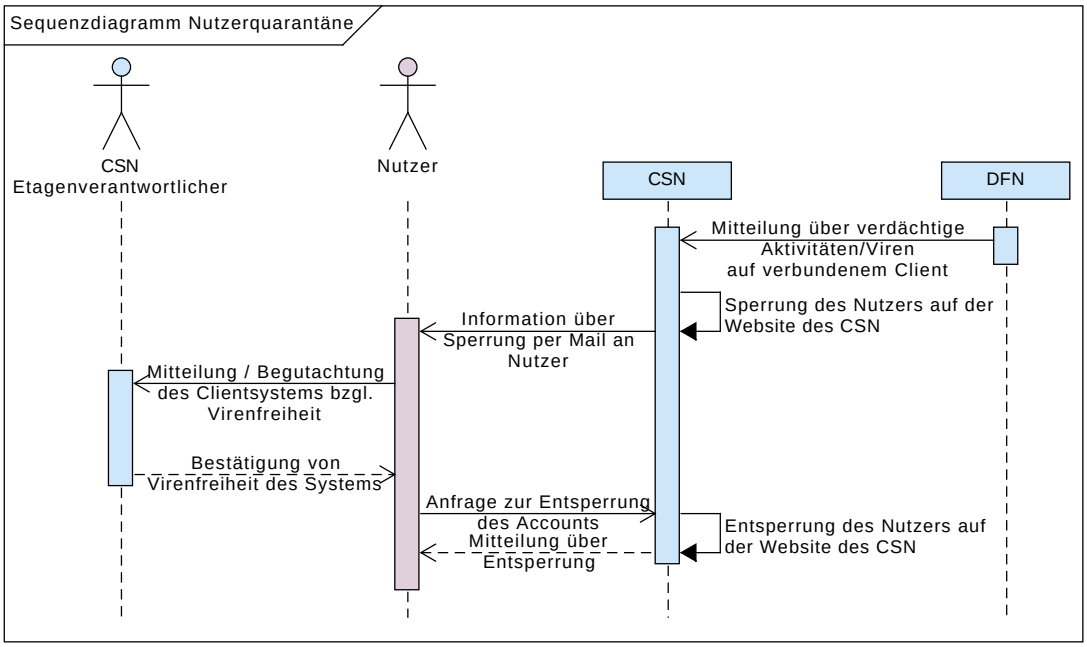

Abbildung 2.2.: Ablauf bei durch das DFN festgestelltem Virenbefall

bestätigen Nutzer oder Etagenverantwortliche nochmal die Virenfreiheit beim CSN Team/ CSN Helpdeskteam. Der Nutzer wird nun wieder manuell entsperrt und anschließend automatisiert binnen zehn Minuten wieder in das normale Hausnetz verschoben.

3. Anforderungen

Für die im Rahmen des Praktikums entstehende Netzwerkumgebung gelten die in Tabelle 3.1 aufgelisteten Anforderungen als Grundlage der Entwicklung. Dabei ist die Gliederung der einzelnen Anforderungen ungefähr an die Reihenfolge des Eintretens im Falle eines Quarantänefalls eines Nutzers angepasst.

Nummer	**Priorität**	**Anforderung**
1	Kann	Beim Eingang einer DFN-Warnmeldung wird der in der Mail erwähnte Nutzer automatisch gesperrt (eine Mail ähnlich einer DFN-Warnung muss eine Sperrung auslösen)
1.1	Soll	Nachdem ein Nutzer gesperrt wurde, erfolgt eine Bestätigung per Mail an das CSN-Administrationsteam, das Rechenzentrum der Universität und an den betroffenen Nutzer. Die Mails sollen innerhalb von 30 Minuten nach Eingang der DFN-Warnmeldung versendet werden.
1.2	Muss	Zusätzlich zu DFN-Warnmeldungen muss ein Webinterface existieren, auf dem Sperrungen manuell durch das CSN-Administrationsteam eingerichtet werden können.
2	Muss	Gesperrte Nutzer werden bei der ersten Internetnutzung nach der Sperrung auf ein Captive Portal umgeleitet. Für die Betriebssysteme Windows 10, Android und iOS muss diese Funktion erfüllt sein.
3	Muss	Die Webseite des Captive Portals muss in der CSN-Webseite integriert sein.
3.1	Muss	Das Captive Portal informiert den Nutzer umfassend über die Sperrung. Der Nutzer erhält Informationen zu den folgenden Punkten: • Der Art des Zustandekommens der Sperrung (bspw. Quarantäne aufgrund von Malware, Verbannung wegen Verstößen gegen die CSN-Ordnung, Sperrung durch das SWCZ) • Detaillierte Belege für die Sperrung (Warnmeldungen des DFN, Abuse-Reports vom URZ) • Einschränkungen für die Zeit der Sperrung (zeitlich eingeschränkte oder keine Netzkonnektivität), Auflistung erlaubter oder verbotener Netzanwendungen, siehe Punkt 4 • Anleitung zu Lösung der Sperrung (Virenscan der Rechner, Kontakt zum Studentenwerk) und Hilfsmöglichkeiten (Etagenverantwortliche des CSN, Links zu Wiki-Artikeln, etc.)
3.2	Muss	Das Gateway, an welchem das Quarantänenetz terminiert wird und auf dem zum Captive Portal umgeleitet wird, muss einfach provisionierbar sein, es existiert eine Konfigurationsautomatisierung (vgl. Ansible Playbook) für das Captive Portal

Nummer	Priorität	Anforderung
4	Muss	Nutzer im Quarantänenetz haben zeitlich begrenzten Zugang zu Internetressourcen, besonders wichtig sind hierbei die Konnektivität zu: • Üblichen Antivirensoftware und Updateservern • Netzwerkverbindungen zu notwendigen Diensten (DNS, Web, Mail) werden zugelassen, andere blockiert • Zugang zu dem vom DFN gemeldeten Angriffsziel muss unterbunden werden • Zugang zum restlichen CSN-Netz (insb. zu anderen Wohnheimnetzen) muss unterbunden werden
5	Muss	Etagenverantwortliche müssen Nutzern den Zugang zum Internet verlängern können.
6	Muss	Das Gateway im Quarantänenetz soll zu Qualitätssicherungs -und Problembehandlungszwecken überwacht werde. Hierbei soll vor allem ersichtlich werden, welche Nutzer welche Verbindungen aufgebaut haben.

Tabelle 3.1.: Anforderungen

4. Bisherige und aktuelle Entwicklungen

In diesem Kapitel werden bisherige und aktuelle Entwicklungen in der für die Aufgabenstellung relevanten Literatur betrachtet. Dazu zählen Aspekte der Ausbreitung von Schadsoftware und deren Eindämmung sowie die Umsetzung von Captive Portals. Dabei werden zunächst Ansätze betrachtet, wie die Ausbreitung von Schadsoftware eingedämmt werden kann, zunächst allgemein, anschließend in der Umsetzung in Quarantänenetzen. Weiterhin wird der Einsatz von Captive Portals in Quarantänenetzen betrachtet sowie damit verbundene Probleme.

In Li et al. [7] werden drei grundlegende Ansätze beschrieben, wie die Ausbreitung von Schadsoftware in Netzwerken eingedämmt werden kann. Dabei handelt es sich um Verlangsamung, Blockierung und Ablenkung. Verlangsamung beinhaltet die Methoden, um die Ausbreitungsgeschwindigkeit zu reduzieren, beispielsweise durch Verzögerung von verdächtigem Traffic. Dadurch soll Zeit gewonnen werden, um die Schadsoftware zu neutralisieren. Blockierung bedeutet, Traffic zu blocken, der vermutlich durch Schadsoftware ausgelöst wurde. Dabei kann entweder Traffic mit bestimmten Inhalten blockiert werden, oder mit bestimmten Absendern bzw. Empfängern, wobei Ersteres als effektiver angesehen wird. Ablenkung erfolgt durch Bereitstellung vermeintlicher Ziele für Schadsoftware, um diese von realen Zielen abzulenken und somit die Ausbreitung zu diesen zu unterbinden oder zu verzögern.

Moore et al. [9] beschreibt weitere Ansätze, um die Ausbreitung von selbst-verbreitender Schadsoftware zu verhindern. Eine Variante besteht darin, durch Prävention die von der Schadsoftware ausnutzbaren Schwachstellen zu reduzieren. Wenn weniger Angriffsziele existieren, kann sich diese nur langsamer ausbreiten. Ein weiterer Ansatz besteht in der Säuberung befallener Systeme, um die Ausbreitung zu verhindern. Dieser Vorgang kann jedoch zu lange dauern, sodass sich die Schadsoftware in der Zwischenzeit ungehindert ausbreiten kann. Entsprechend wird daher vorgeschlagen, durch Isolation betroffener Systeme die Ausbreitung aufzuhalten, bis eine Säuberung des Systems durchgeführt wurde. Um die Effektivität von solchen Systemen bestimmen zu können, wurden mathematische Modelle von infektiösen Krankheiten verwendet, um die Ausbreitung von Schadsoftware zu simulieren. Entsprechende Eindämmungssysteme wurden simuliert, um deren Wirksamkeit bzgl. der Isolation der Schadsoftware zu ermitteln.

Diese Analyse ergab, dass durch Isolation eine Ausbreitung wirksam gestoppt werden kann. Als Bedingung wurde festgestellt, dass die Isolation automatisiert erfolgen sollte. Im Paper wurde zudem die Möglichkeit eines internetweiten Eindämmungssystems diskutiert, welches jedoch starke Zusammenarbeit der Internetprovider benötigt und daher als sehr herausfordernd betrachtet wurde.

Ein anderer Ansatz wurde von Matthew M Williamson in [15] untersucht, indem die Ausbreitung von Schadsoftware durch Begrenzung der Anzahl neuer Verbindungen durch Endgeräte minimiert werden sollte. Dieses Vorgehen basiert auf der Beobachtung, dass die Schadsoftware, die versucht, sich selbst zu verbreiten, nach der Infektion schnell viele Verbindungen zu anderen Geräten öffnet, um diese ebenfalls zu infizieren. Dies steht im Kontrast zum beobachteten Verhalten nicht infizierter Geräte. In dem Paper wurde untersucht, inwieweit die Begrenzung neuer Verbindungen die Ausbreitung von Schadsoftware eingrenzt, ohne den normalen Internetverkehr zu beeinträchtigen.

Dafür wurde das Verhalten normaler, nicht befallener Systeme untersucht. Dabei wurde festgestellt, dass seltener Verbindungen zu neuen Hosts aufgebaut werden und Verbindungen zu gleichen Hosts zeitlich in Bezug stehen. Die Wahrscheinlichkeit, zu einem alten Host zurückzukehren, sinkt mit der Zeit. Basierend darauf wurde ein Algorithmus entwickelt, um Verbindungen in normale und solche, die durch Schadsoftware ausgelöst werden, zu unterteilen. Letztere können dann eingeschränkt werden, ohne normale Nutzung einzuschränken. Gleichzeitig wird die Ausbreitungsgeschwindigkeit der Schadsoftware stark verringert.

Ähnlich untersucht Wong et al. [16], inwieweit die Ausbreitung von Schadsoftware durch Begrenzung der Ausbreitungsrate eingedämmt werden kann. Dafür werden verschiedene Strategien zur Begrenzung modelliert und simuliert. Als Ergebnis wurde festgestellt, dass die Begrenzung sowohl an den Außengrenzen eines Netzwerkes stattfinden muss, als auch an einem bestimmten Teil der Hosts im Inneren des Netzwerks. Anhand realer Netzwerkdaten wurden zudem Werte für Begrenzungen ermittelt, die kaum Auswirkungen auf legitimen Traffic haben, jedoch die Ausbreitung von Schadsoftware stark reduzieren.

Ahmad et al.[1] untersucht ebenfalls, wie Schadsoftware gefunden und eingedämmt werden kann. Als Basis der Erkennung werden Verbindungen zu IP-Adressen ohne entsprechende vorherige DNS-Anfrage Schadsoftware zugeordnet, die versucht, sich im Netzwerk schnellstmöglich auszubreiten. Dafür werden die Layer-4-Datagramme mit einem DNS-Resolution-Cache verglichen, um Verbindungen ohne vorherige DNS-Anfragen zu ermitteln. Überschreitet die Anzahl der unzulässigen Verbindungen einen Grenzwert, werden Verbindungen des Endgerätes blockiert, um eine Ausbreitung von Schadsoftware zu verhindern.

Im Gegensatz dazu beschreibt Yasutome et al. in [17] verschiedene Ansätze, um Netzwerksicherheit durch Quarantäne zu gewährleisten. Dabei können Endgeräte, die frei von Schadsoftware sind und ordnungsgemäß konfiguriert und gepatcht sind, normal das Netzwerk benutzen. Endgeräte, die diesen Anforderungen nicht entsprechen, werden mittels Quarantäne vom restlichen Netzwerk isoliert, um mögliche Schadsoftware einzudämmen. Während sich Endgeräte in Quarantäne befinden, bekommen sie die Möglichkeit, Updates und Patches anzuwenden und gegebenenfalls Schadsoftware wieder zu entfernen. Wenn dies erfolgreich ist und ein Endgerät den Anforderungen wieder entspricht, kann es in das normale Netzwerk zurückkehren und wird nicht länger isoliert.

Für die Umsetzung der Quarantäne werden mehrere Ansätze beschrieben. Eine Möglichkeit besteht darin, das Quarantänenetzwerk als separates VLAN einzurichten. Endgeräte, die isoliert werden sollen, werden nach Authentifizierung dem Quarantäne-VLAN zugewiesen und dadurch vom normalen Netzwerk isoliert. Eine andere Möglichkeit besteht in einer Firewall auf dem Gateway System, welches Endgeräte am Gateway zum restlichen Netzwerk anhand fixer IP-Adressen isoliert. Die Firewall kann sich auch auf dem Endgerät selbst befinden, jedoch erfordert dies spezielle Software auf allen Endgeräten. Die Firewall kann sich auch auf Servern im Netzwerk befinden, jedoch werden dadurch nur die Server geschützt und keine Isolation zwischen anderen Endgeräten erwirkt.

Weiterhin werden Probleme diskutiert, die durch Einführung eines Quarantänenetzwerks entstehen können. Zunächst muss im Vorfeld festgelegt werden, welche Ziele (Netzwerke, Endgeräte) geschützt werden müssen. Darauf basierend soll dann die Methode zur Isolation ausgewählt werden. Diese Entscheidung wird dann weiterhin beeinflusst durch Vorhandensein spezieller Hardware und Kosten für eventuell notwendige Hardware. Weiterhin muss entschieden werden, ob die Quarantäne nur für Endgeräte in bestimmten Bereichen eingesetzt wird oder für alle Netzwerke.

Bomhoff et al. [3] beschreibt und evaluiert, wie ein existierendes Quarantänenetz an der University of Twente automatisiert wurde. In dem Netzwerk wurden von Schadsoftware befallene Systeme ursprünglich manuell vom restlichen Netzwerk getrennt.

In dem neuen System existierten mehrere Honeypot Systeme, die Schadsoftware erkennen und die davon betroffenen Geräte automatisiert in ein Quarantänenetz verlagern. Die Isolation erfolgte durch MAC- und IP-basierendes Blocken der betroffenen Geräte. Deren Anfragen in das normale Campusnetzwerk wurden geblockt, sodass die Schadsoftware sich nicht weiter in das Netzwerk ausbreiten kann. Nur bestimmte Anfragen wurden anhand einer Whitelist zugelassen, um die Bereinigung des Systems von Schadsoftware zu ermöglichen. Benutzer der betroffenen Systeme wurden per E-Mail über die Quarantäne informiert.

Weiterhin erhielten Systeme im Quarantänenetz eine Möglichkeit, sich selbst aus der Quarantäne zu entlassen, nachdem das System bereinigt wurde. Diese Option kann jedoch nur begrenzt oft in einem bestimmten Zeitraum verwendet werden, um Missbrauch zu verhindern. Möglichkeiten zur Erweiterung des Systems bestehen in Verbesserung der Erkennung der Schadsoftware sowie durch bessere Isolation der Systeme im Quarantänenetz. Dafür wurden unter anderem VLANs in Betracht gezogen.

Die Eigenschaften von VLANs und deren Implikationen in Bezug auf Sicherheit wurden von Bassey et al. [2] untersucht. In VLANs nutzen mehrere logisch getrennte Netzwerke die gleiche Layer-2-Hardware. Dabei werden in den Switches die Pakete je nach VLAN unterschiedlich weitergeleitet. Im Switch werden Ports zu bestimmten VLANs zugeordnet, und Pakete von einem VLAN werden immer nur an Ports weitergeleitet, die für das gleiche VLAN konfiguriert sind.

Im Ergebnis wurde festgestellt, dass VLANs ermöglichen, Geräte unterschiedlicher logischer Netzwerke an beliebigen Switches mit dem physischen Netz zu verbinden, ohne dass auf andere logische Netze zugegriffen werden kann. Zudem wurde festgestellt, dass die Netzwerksicherheit in den untersuchten Netzen durch die Verwendung von VLANs erhöht wurde. VLANs können auch die Anzahl an Broadcast-Domains erhöhen und gleichzeitig deren Größe verringern, was logische Gruppierung von Nutzern erleichtert, da die logischen Netzwerke nicht durch physische Kapazitäten, wie die Anzahl der Ports eines Hubs, eingeschränkt sind.

Poger und Baker [12] untersuchen die Möglichkeiten, Internetzugriff an einem physischen Port auf autorisierte Nutzer zu beschränken. Dafür wurde ein Captive Portal entwickelt, bezeichnet als „Prisonwall“. Wenn sich ein Nutzer mit dem Netzwerk verbindet, muss sich dieser erst authentifizieren, bevor Zugriff in das Internet ermöglicht wird. Der dafür konfigurierte Router agiert dann als Gateway und lässt nur Kommunikation von autorisierten Nutzern in das Internet zu.

Die Probleme, die bei der Verwendung eines Captive Portals entstehen können, wurden von Nottingham [10] zusammengefasst. Captive Portals werden aus unterschiedlichen Gründen eingesetzt: Nutzerauthentifizierung, Bezahlung für Internetnutzung, Information des Nutzers, z.B. über rechtliche Bedingungen, sowie Benachrichtigungen für Nutzer. Dabei können jedoch verschiedene Probleme auftreten.

Captive Portals sind effektiv ein Man-in-the-Middle-Angriff, da sie auf eine angefragte Ressource eine andere, nämlich das Captive Portal, zurückliefern. Dies kann zu Verwirrung bei dem Endnutzer führen. Bei Verwendung von TLS erfordert das Captive Portal, dass der Nutzer die Sicherheitswarnungen ignoriert, um das Captive Portal zu erreichen. Zudem können Probleme auftreten, wenn ungewöhnlich konfigurierte Geräte sich mit dem Captive Portal verbinden. Internetzugriffe außerhalb

eines Browsers können zudem nicht mit dem Captive Portal interagieren und meist das entsprechende Netzwerk nicht nutzen.

Um die Captive Portals von einem Man-in-the-Middle-Angriff zu unterscheiden, führen Betriebssysteme und/oder Webbrowser eine Captive Portal Erkennung durch. Dieses Verhalten wurde von Motoyuki [11] in einer Studie in einem Campus-Netzwerk untersucht. Um die Erkennung durchzuführen, verbindet sich der Client mit einer bekannten URL, die eigens für diesen Zweck existiert. Die erwartete Antwort auf die Anfrage ist dem Client im Voraus bekannt. Wird die Anfrage durch ein Captive Portal verhindert und als Antwort eine Umleitung auf das Captive Portal an den Client gesendet, kann dieser schlussfolgern, dass ein Captive Portal existiert und entsprechende Schritte einleiten, um den Nutzer damit interagieren zu lassen. Diese Erkennung existiert zwar bei den meisten Betriebssystemen und/oder Webbrowsern, ist jedoch nicht immer auch für LAN aktiviert, sondern teilweise nur für WLAN.

Eine andere Möglichkeit der Erkennung eines Captive Portals wird in RFC7710 [5] definiert. Dabei wird eine DHCP-Nachricht sowie Router Advertisement (RA) definiert, die von einem Client genutzt werden können, um die URI eines möglichen Captive Portals zu erfragen. In der genannten Studie wurde jedoch festgestellt, dass der RFC zu dem Zeitpunkt weder von Windows 10 noch macOS unterstützt wurde und die im RFC definierten Möglichkeiten zur Erkennung von Captive Portals nicht genutzt wurden.

IEEE Standard 802.1X stellt eine Alternative zu einem Captive Portal dar, jedoch können hier betriebssystemspezifische Probleme auftreten, die eine Konfiguration des Endgeräts durch den Nutzer erfordern, da 802.1X für Verbindungen via Kabel teilweise nicht standardmäßig aktiviert ist. In [8] ist ein Ansatz beschrieben, wie Captive Portals in Kombination mit 802.1X betrieben werden könnten, jedoch wird auch hier eine Konfiguration durch den Nutzer benötigt. Dies basiert auf einer Erweiterung von EAP, genannt EAP-SH (Secure Hosts). Diese soll erlauben, bei der Authentifizierung ein Captive Portal in einem Browser zur Nutzerinteraktion zu öffnen. Dafür muss der Client jedoch das Zertifikat des Hotspots im Voraus importieren.

Trotz weiter Verbreitung von Captive Portals gibt es jedoch bislang keine offiziellen Standards, die weit verbreitet implementiert sind. Derzeit befindet sich ein IEEE Internet-Draft [6] für eine Captive Portal Architektur in Arbeit. Die geplante Architektur soll einem Endgerät erlauben, seinen Zustand bzgl. der Einschränkung durch das Captive Portal sowie die URI des Portals über eine API zu ermitteln. Dadurch soll die Notwendigkeit, eine Anfrage an eine bekannte URL zu stellen, nur um die Existenz eines Captive Portals zu erkennen, zu eliminieren.

Stattdessen soll via RFC7710 die URL der Captive Portal API erfragt werden, an der abgefragt werden kann, ob das Gerät normal mit dem Internet interagieren kann oder

ob es eingeschränkt ist. Außerdem soll die URI der Portal Anwendung, mit der der Nutzer in einem Browser interagiert, abgefragt werden können. Darüber hinaus soll ein Protokoll zum Einsatz kommen, um Geräten mitzuteilen, dass Anfragen durch das Captive Portal geblockt werden, ohne eine falsche Antwort auf die Anfrage zu schicken, wie es von bisherigen Captive Portals gehandhabt wird.

Koht-Arsa et al. [4] betrachtet eine andere Architektur für Large-Scale Captive Portals. Diese besteht aus parallelen Firewalls, einem Stateless-HTTP-Redirector und Authentication-Servern. Dabei blockt die Firewall nicht authentifizierten Traffic. HTTP-Requests werden an den HTTP-Redirector weitergeleitet. Traffic von authentifizierten Clients wird weitergeleitet. Der HTTP-Redirector leitet die Clients an die Authentifizierungswebsite weiter. Dort kann sich der Client authentifizieren, was eine Änderung der Firewallregeln hervorruft, sodass der Traffic des Clients nicht länger durch die Firewall geblockt wird.

Ziel der Architektur liegt darin, eine große Anzahl gleichzeitiger Authentifizierungen durchzuführen und dabei gleichzeitig robust gegenüber mit Schadsoftware infizierten Clients und fehlerhaft konfigurierter Software zu sein. Das Captive Portal wurde in einem Campus-Netzwerk evaluiert.

Watanabea et al. [14] beschreibt das Captive-Portal-System „Opengate“, welches zwei Besonderheiten besitzt. Erstens wird das Ende jeder Nutzersitzung exakt festgestellt, indem zusammen mit dem Captive Portal eine weitere Website angefordert wird. Diese beinhaltet eine JavaScript-Funktion, die wiederholt eine AJAX-Anfrage an einen speziellen Beobachtungsprozess sendet. Dabei wird die Antwort durch den Beobachtungsprozess verzögert. Sobald die Nutzersitzung durch Schließen der Website beendet wird, wird gleichzeitig auch die TCP-Verbindung zum Beobachtungsprozess geschlossen. Als Reaktion darauf wird die Nutzersitzung als beendet betrachtet, wodurch verhindert wird, dass die Sitzung von einem nicht authentifizierten Nutzer weiterverwendet werden kann.

Um die IPv4- und IPv6-Adressen zu erfahren, werden von der Firewall, die nicht autorisierte Anfragen an das Captive Portal weiterleitet, zunächst die IPv6-Anfragen ignoriert und ein Reset-Signal zurückgesendet. Für weitere Kommunikation muss das Endgerät auf IPv4 zurückfallen. Das Captive Portal erhält dadurch zunächst die IPv4-Adresse und kann bei der Firewall anfragen, um gegebenenfalls die IPv6-Adresse zu erfahren.

Für eine weitere Variante, Captive Portals in einer IPv4/IPv6-Dual-Stack Umgebung umzusetzen, wird von Sanguanpong und Koht Arsa [13] ein Design eines Captive Portals in einem IPv4/IPv6-Dual-Stack-System, welches Clients ermöglicht, sich beliebig über IPv4 oder IPv6 zu authentifizieren, ohne sich zweimal authentifizieren zu müssen, beschrieben. Das Design basiert ebenfalls auf einer Firewall, die Traffic

von nicht authentifizierten Clients blockt und deren HTTP- oder HTTPS-Anfragen an ein Authentifizierungssystem umleitet. Letzteres geschieht mittels eines Stateless-HTTP-Redirector innerhalb der Firewall.

Für authentifizierte Nutzer werden sowohl IPv4- als auch IPv6-Adressen hinterlegt, sodass die Firewall den Nutzer anhand beider Adressen identifizieren kann. Um die jeweiligen Adressen zu ermitteln, werden vom Nutzer beim Aufrufen der Captive Portal Website zwei weitere Dateien nachgeladen, deren URLs jeweils auf eine IPv4- und eine IPv6-Adresse verweisen. Dadurch werden dem Authentifizierungssystem beide Adressen bekannt, sofern der Client beide Stacks benutzt.

5. Beurteilung des State of the Art

Im Folgenden sollen die in 4 betrachteten Ansätze in Bezug auf die gestellten Anforderungen beurteilt werden. Dabei werden insbesondere die automatisierte Mailverarbeitung, die Quarantäne der Nutzergeräte sowie die Umsetzung des Captive Portals näher betrachtet.

5.1. Automatisierte Mailverarbeitung

Um die eingehenden Warnmeldungen zu verarbeiten, muss ein XML-Anhang geparst werden. Dafür kann entweder ein SAX-Parser oder ein DOM-Parser eingesetzt werden. Ein SAX-Parser bietet im Anwendungsfall keine nennenswerten Vorteile, wohingegen die Verwendung eines DOM-Parsers zu einer vergleichsweise höheren Lesbarkeit und Nachvollziehbarkeit der resultierenden Implementation führt.

5.2. Quarantänenetz

Für das Quarantänenetz müssen die Nutzer identifiziert werden und Nachrichten von Nutzern in Quarantäne müssen anders behandelt werden als Nachrichten von Nutzern im normalen Netz. Eine Möglichkeit besteht darin, die Nutzer anhand von MAC- oder IP-Adressen zu identifizieren und deren Nachrichten zum Captive Portal weiterzuleiten [3]. Da es für Nutzer möglich ist, die MAC-/IP-Adressen zu fälschen, ist diese Variante nicht sinnvoll.

Alternativ kann ein separates VLAN als Quarantänenetz verwendet werden, um die Netze zu isolieren. Dadurch findet die gleiche Nutzeridentifikation und -isolation statt, ohne dass die Nutzer darauf Einfluss nehmen können. Dafür wird die Nutzeridentifikation auf den Switches durchgeführt und Nutzer anhand des Netzanschlusses identifiziert. Die Variante ist vor Nutzereinfluss geschützt, solange die Nutzer keinen Zugriff auf die Switches haben. [2, 3]

5.3. Captive-Portal-Erkennung

Captive-Portal-Erkennung wird teilweise vom Betriebssystem und teilweise von Webbrowsern unabhängig voneinander durchgeführt. Dabei wird meist eine bekannte, eigens dafür existierende Ressource abgerufen. Wird der Zugriff auf die Ressource durch das Captive Portal verwehrt, erhält das Betriebssystem bzw. der Webbrowser eine andere Antwort als die Erwartete und kann daraus die Existenz eines Captive Portals herleiten. [11]

Im derzeit in Entwicklung befindlichen *capport* Entwurf [6] soll stattdessen dem System durch eine DHCP-Nachricht mitgeteilt werden, dass ein Captive Portal existiert und Informationen an einer bestimmten Adresse abgerufen werden können.

5.4. Captive-Portal-Umsetzung

Das Captive Portal fängt Anfragen des betreffenden Gerätes ab und antwortet mit der Captive-Portal-Website, die dann dem Nutzer angezeigt werden soll. Dies ist für das Gerät nicht von einem Man-in-the-Middle-Angriff unterscheidbar, da die erhaltene Antwort des Captive Portals nicht vom erwarteten Absender stammt, bzw. dies nicht durch das Captive Portal bewiesen werden kann.

Anstelle eines Captive Portals könnte auch ein Proxy verwendet werden, der die Anfragen des Nutzers gegen eine Whitelist prüft und nur bestimmte Anfragen ins normale Internet zulässt. Dann kann auf das Senden einer „gefälschten“ Antwort durch das Captive Portal verzichtet werden, jedoch erfährt der Nutzer nicht direkt, dass sich das Gerät im Quarantänenetz befindet.

6. Erfüllung der Anforderungen durch aktuelle Ansätze

Ausgehend von den in Kapitel 3 gestellten Anforderungen für das Quarantäne-Netz werden im Folgenden Komponenten und Ansätze evaluiert, die in der entstehenden Netzwerkumgebung Anwendung finden können. Es werden hierbei Ansätze zur automatischen Verarbeitung von Mails, Verfahren zur Quarantäne-Vollstreckung, sowie Software zur Bereitstellung von Captive-Portal-Netzwerken verglichen.

6.1. Bewertungsmaßstab

Um objektiv die Ansätze und Software vergleichen zu können, werden vorab Maßstäbe definiert, welche erfüllt sein müssen, um eine Anforderung aus Kapitel 3 teilweise oder vollständig zu erfüllen. Diese Maßstäbe sind in Tabelle 6.1 beschrieben. Es werden nur solche Anforderungen aus Kapitel 3 betrachtet, die durch eine schon bestehende Lösung umgesetzt werden können.

Anforderung	++	+	-	--
1	Beliebige Informationen aus dem XML-Anhang können direkt ausgelesen werden	Kompletter XML-Anhang muss ausgewertet werden um alle Informationen zu erhalten	Informationen können in bestimmten Fällen nicht ausgelesen werden	Nicht Möglich
2	Umleitung des Nutzers auf eine einfach einzurichtende Captive-Portal-Website	Umleitung des Nutzers auf eine Captive-Portal-Website, die selbst konfiguriert werden muss	Umleitung des Nutzers auf eine Captive-Portal-Website, die nicht konfiguriert werden kann	Nutzer wird nicht umgeleitet und erfährt nicht direkt über die Quarantäne
3.1	Die Captive-Portal-Website kann individuell angepasst werden und ist einfach einzurichten	Die Captive-Portal-Website kann nur aufwendig individuell angepasst werden	Die Captive-Portal-Website kann nicht individuell angepasst werden	Die Captive-Portal-Website kann nicht konfiguriert werden oder existiert nicht

Anforderung	++	+	-	--
3.3	Vergleichbare Ansible-Playbooks existieren und API verfügbar	Vergleichbare Ansible-Playbooks existieren	Einstellung nur über Konfigurationsdateien	Einstellung nur manuell über z.B. Weboberfläche
4	Eingebaute Voucher für zeitliche Freistellung	Zeitliche Freistellung kann konfiguriert werden, jedoch nicht direkt eingebaut	Aufwendige eigene Lösung	Nicht Möglich
6	Einfach automatisiert konfigurierbar	Externe Lösung	Aufwendige eigene Lösung	Nicht Möglich

Tabelle 6.1.: Bewertungsmaßstab

6.2. Bewertung

Tabelle 6.2 fasst aus obigem Bewertungsmaßstab sowie der Analyse bestehender Systeme (siehe Anhang, Tabelle A.1) zusammen mit Abschnitt 5.1 und Abschnitt 5.2 eine Bewertung für Prinzipien automatischer Mailverarbeitung, Konzepte zur Isolierung infizierter Rechner sowie Software für eine Captive-Portal-Bereitstellung zusammen. Die grau unterlegten Konzepte stellen hier die Ansätze dar, welche am besten die Anforderungen erfüllen.

Anforderung	SAX Parser	DOM Parser	Verbindungslimit	VLAN	IP/MAC basiertes Blocken	Wifidog	Packetfence	Coova Chili	pfSense	OPNsense	Proxy + Whitelist
1	+	++									
2						++	++	+	++	++	--
3						-	++	+	++	++	--
3.2						--		-	+	++	
4			-	++	++	--	+	-	++	++	+
6						--		--	+	+	-

Tabelle 6.2.: Bewertung

7. Konzept

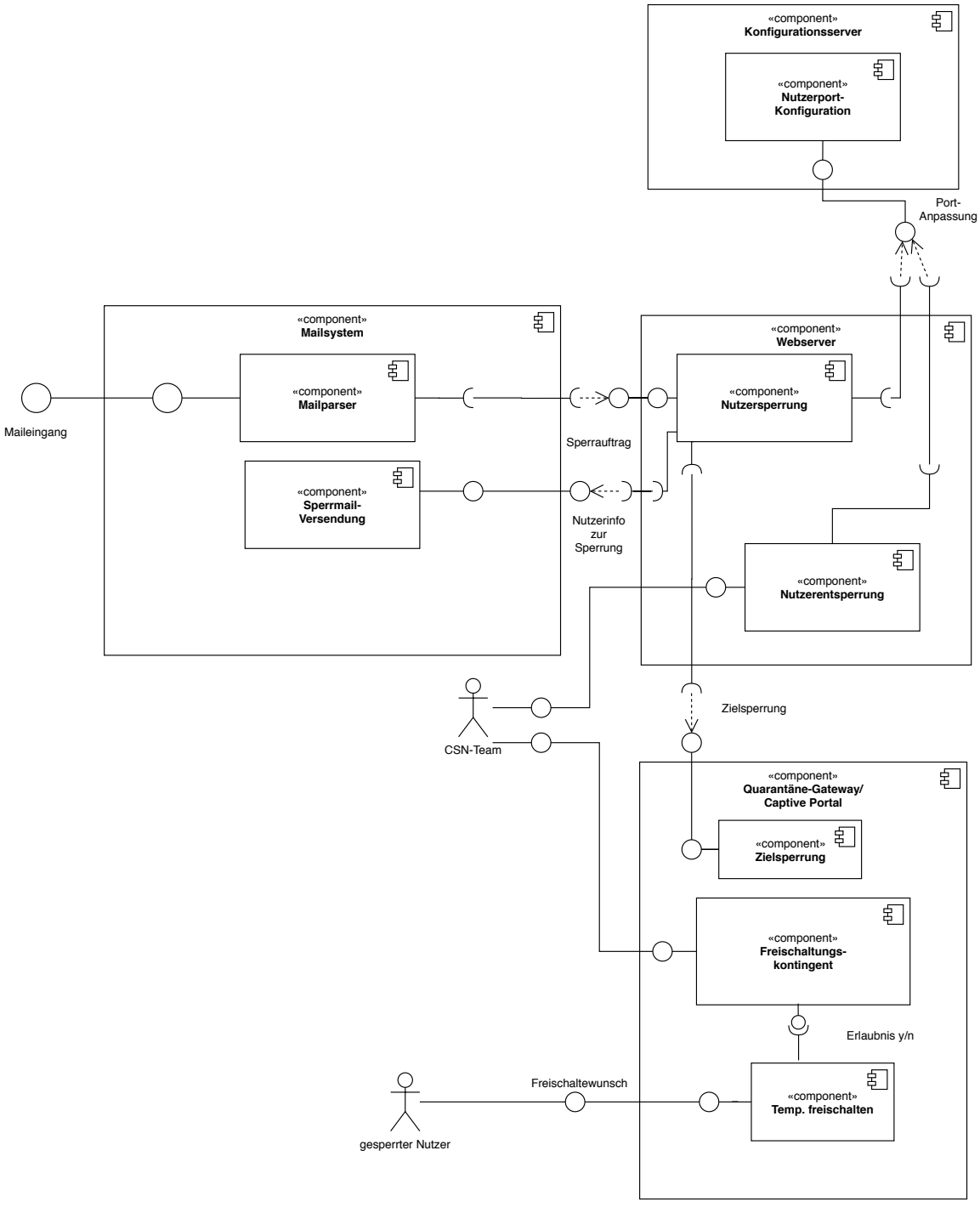

Abbildung 7.1.: Architektur

Ausgehend von den Evaluationen möglicher Ansätze in Kapitel 6 sowie den Netzwerkgegebenheiten im CSN soll eine folgend geartete Architektur der Quarantäne-Umgebung, wie in Abbildung 7.1 skizziert, entstehen. Für die Komponente *Quarantäne-Gateway* werden schon existierende Lösungen evaluiert, für die Komponenten *Mailsystem*, *Webserver* und *Konfigurationsserver* ist dies nicht möglich, da hier im CSN etablierte und individuell erstellte Lösungen weiterverwendet oder erweitert werden sollen.

Anstelle der zuvor manuellen Mail-Verarbeitung werden Warnmeldungen des DFN automatisch ausgewertet. Hierzu leitet der Mailserver des CSN besagte Meldungen des DFN an einen extra Mailserver weiter, der im Sinne von Microservices ausschließlich für die Verarbeitung derer zuständig ist und Sperrungen per API an die CSN-Website weiterleitet. Für die Entwicklung wird ein DOM-Parser genutzt.

Im Sinne der Nutzung bestehender Komponenten löst die Meldung einer Nutzersperrung über im CSN schon existierende Komponenten den Wechsel des Access-VLANs der Nutzerports aus, welches den Nutzer in das Quarantänenetz befördert. Durch die Nutzung eigener VLANs kann man durch Layer-2-Trennung von normalen Nutzern eine möglichst hohe Sicherheit für diese erreichen. Im Vergleich zur früheren Lösung gibt es jedoch nicht zu jedem Haus- ein zusätzliches Quarantäne-Netz, sodass die Nutzer nicht ihre normalen Adressen behalten, wie unten dargelegt.

Aufgrund des Wechsels in das Quarantäne-Netz verändert sich für die Endgeräte der Nutzer die Layer-2-Domäne, dadurch vor allem ihr Default-Gateway für Netzverbindungen, da dieses Netz nicht mehr am Core-Router des CSN terminiert, sondern in einem alternativen Routing-Gerät, dem Quarantäne-Gateway. Dort bekommen die Nutzer per DHCP temporäre Adressen aus einem privaten Netzbereich nach RFC1918 zugeteilt. Im Vergleich zu Verwendung der gewöhnlichen Nutzer-IP-Adressen, benötigt die Vergabe von temporären Adressen nicht die Konfiguration des Quarantäne-Gateways mit den Entsprechungen von MAC-Adresse zu IP-Adresse und -Subnetz, welches im Falle eines erfolgreichen Angriffes auf das Quarantäne-Gateway potenziell geringeren Schaden hervorruft. Deswegen ist es nicht mehr nötig, analog zur gewöhnlichen Netzstruktur des CSN einzelne Quarantäne-Netze für die sonst bestehenden Haus-Netze zu unterhalten.

Für die Interaktion zu dem Nutzer besteht ein Captive Portal, welches die Aufmerksamkeit des Nutzers erlangt, in dem ohne eine Freischaltung der Nutzer auf dieses umgeleitet wird. Da heutzutage eine Vielzahl von Webseiten HTTPS einsetzen, bei dessen Verbindungsaufbau eine Manipulation der Verbindung, wie es das Captive Portal tut, auffallen und zu einer Unterbrechung dessen führen würde, werden die automatischen Erkennungsverfahren für Captive Portals der Browser den Nutzer zu dem Captive Portal des CSN weiterleiten.

Für eine Empfindung der Legitimität des Captive Portals durch die in Quarantäne gestellten Nutzer ist es notwendig, dass die Oberfläche an jene der normalen CSN-Webseite erinnert. Aus Gründen der einfacheren Pflege der Webseiten sowie Sicherheitsgedanken (nicht benötigte Verbindung zwischen Quarantäne-Gateway und Datenbank mit Nutzerdaten) findet ein Betrieb des Captive Portals außerhalb des Quarantäne-Gateways statt, sodass das Gateway auf das Captive Portal umleiten muss. Aufgrund des Maskierens der Nutzer im Quarantäne-Netz hinter der IP des Gateways muss nun zusätzlich eine Identifikation der Nutzer aus Sicht des Captive Portals möglich sein. Dafür wird die MAC-Adresse des anfragenden Gerätes vom Quarantäne-Gateway an das Captive Portal weitergegeben. Diese MAC-Adresse wird auf dem Captive Portal einem im CSN registrierten Nutzer zugeordnet. Im Falle einer erfolgreichen Aktivierung teilt das Captive Portal, welches die Nutzerinteraktion erfährt, folgend dies dem Quarantäne-Gateway mit, welche für eine vom Captive Portal festgelegten Zeitraum den Nutzer freischaltet. Für das Gateway soll, um den Aufwand der Entwicklung zu reduzieren, ein bestehendes Software-Produkt eingesetzt werden. Da sich OPNsense als das am besten Evaluierte darstellte, wird eine Lösung hiermit angestrebt.

Der Prozess, welcher einen Quarantäne-Fall auslöst, ist in Abbildung 7.2 in Form eines *BPMN*-Diagramms beschrieben. Darin wird der konzipierte Prozessablauf und die Interaktion zwischen den beteiligten Komponenten dargestellt. Bei einer Feststellung von Malwarebefall auf einem dem CSN zugeordneten Adressbereich wird dem CSN eine Mail übermittelt, in der Details zur erkannten Bedrohung und betroffenen Adressen aufgelistet werden. Beim Eintreffen einer solchen Mail werden zuerst alle Aktionen durchgeführt, die notwendig sind, um den Nutzer zu sperren und in die isolierte Netzwerkumgebung zu bringen. Wenn der Nutzer das nächste Mal einen Zugang zum Netzwerk versucht, wird ihm eine Captive-Portal-Seite mit Informationen angezeigt, ob und unter welchen Bedingungen ein Netzwerkzugriff für ihn möglich ist. Die Prüfung erfolgt auf dem Webserver des CSN. Der Nutzer kann nun die Probleme auf seinem Gerät beheben und informiert nach erfolgreicher Beseitigung von Viren und Schadsoftware auf seinem System einen Etagenverantwortlichen oder das CSN-Team darüber. Das CSN-Team entscheidet aufgrund der bereitgestellten Informationen, ob eine Rückkehr in das normale Netz wieder möglich ist und unternimmt alle notwendigen Aktionen, um den Nutzer wieder freizuschalten.

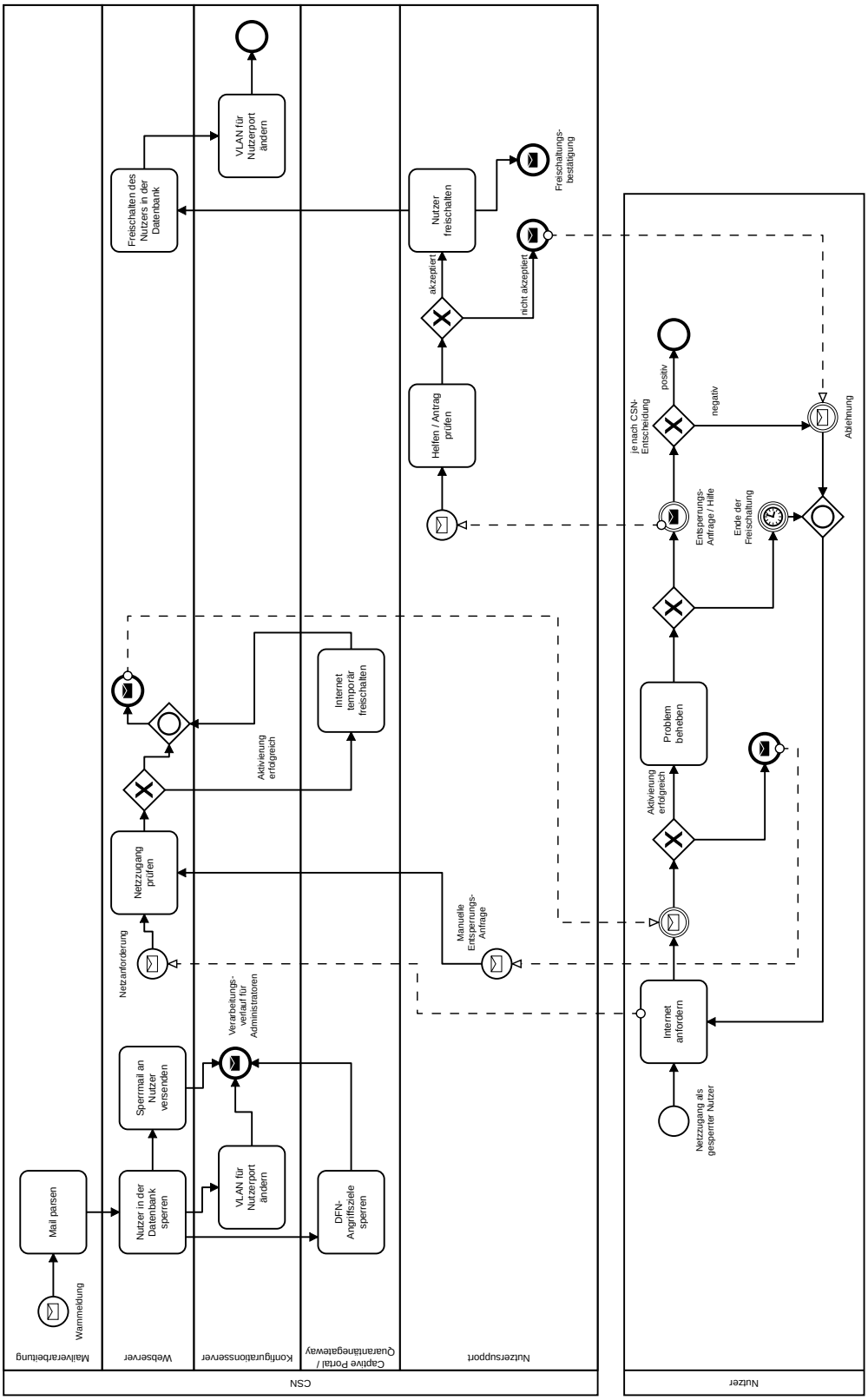

Abbildung 7.2.: BPMN

8. Implementierung

In diesem Kapitel soll die im Praktikum umgesetzte Lösung beschrieben werden. Dabei werden die eigenimplementierten Programme für das Mailparsing und Quarantäne-Gateway, das umgesetzte Captive Portal, sowie die notwendigen Änderungen an der Website und im Netzwerk näher beschrieben.

8.1. Mailverarbeitung

Gemäß Anforderung 1 sollen die Warnmeldungen des DFN automatisiert ausgewertet werden und eine Sperrung auslösen. Die Warnmeldungen besitzen einen umfangreichen XML-Anhang, in dem neben weiteren Informationen Grund der Warnung, auslösender Nutzer, sowie die angegriffenen Ziele enthalten sind. Das automatisierte Auswerten der Informationen geschieht in einem eigenimplementierten Pythonskript, welches anschließend per API-Aufruf an den CSN-Webserver die entsprechenden Sperrungen auslöst.

8.2. Implementation des Captive Portals mit OPNsense

Da OPNsense die umfangreichste Funktionalität der in Tabelle A.1 betrachteten Lösungen besitzt, sollte das Captive Portal damit implementiert werden. Dafür wurde OPNsense auf dem Quarantäne-Gateway installiert. Für das Captive Portal bietet OPNsense die Möglichkeit, dem Nutzer eine Website anzuzeigen. Leider konnte diese jedoch nicht personalisiert werden, da der integrierte Webserver keinen direkten Zugriff auf die CSN-Datenbank besitzt. Eine Möglichkeit besteht darin, den Nutzer zusammen mit entsprechenden identifizierenden Informationen auf den CSN-Webserver weiterzuleiten. Dafür wird zunächst eine AJAX-Anfrage an die OPNsense Captive Portal API gesendet, um die IP des Nutzers im Quarantänenetz zu ermitteln. Diese wird dem CSN-Webserver bei der Weiterleitung ebenfalls übermittelt, sodass dieser über einen weiteren API-Aufruf die MAC-Adresse des Nutzers ermitteln kann und somit den Nutzer in der Datenbank identifizieren kann. Nach der Identifikation können entsprechende personalisierte Informationen angezeigt werden, die unter Anforderung 3.2 definiert wurden.

Weiterhin wird dem Nutzer die Möglichkeit gegeben, sich selbst temporär freizuschalten, um das Problem, das zur Sperrung geführt hat, zu beseitigen. Leider ermöglicht OPNsense keine Freischaltung für einen beliebigen Nutzer, stattdessen kann nur jeder Nutzer sich selbst freischalten. Eine Freischaltung direkt durch den Webserver ist nicht möglich. Stattdessen kann nur ein Voucher generiert werden, der dann vom Nutzer eingelöst werden muss. Dieser Schritt kann automatisiert werden, indem der Nutzer wieder zurück an den Webserver im Quarantäne-Gateway geleitet wird und dort per AJAX direkt den Voucher einlöst.

Leider existieren mehrere Probleme mit dieser Lösung. Zunächst ist die Lösung sehr komplex, da der Nutzer mehrfach zwischen Quarantäne-Gateway und Webserver weitergeleitet werden muss. Dies ist darauf zurückzuführen, dass bestimmte API-Aufrufe nur vom Nutzer selbst durchgeführt werden können, andere Aufrufe aber nur mit autorisiertem Zugriff. Weiterhin werden Nutzer, die einen Voucher eingelöst haben und beim Fristende nicht aktiv den Internetzugang nutzen, nicht korrekt getrennt, wenn die durch den Voucher freigeschaltete Zeit abläuft. Dies kann nur durch ein hartes Timeout realisiert werden, welcher in OPNsense nur global für alle Nutzer mit einer festen Zeit eingestellt werden kann. Entsprechend können Nutzer keine unterschiedlich langen Freischaltezeiten bekommen.

Ein weiteres Problem besteht darin, dass OPNsense nicht ermöglicht, weitere Firewallregeln per API-Aufruf hinzuzufügen. Dies stellt einen Konflikt mit Anforderung 4 dar, wonach der Zugriff auf die gemeldeten Angriffsziele auf dem Quarantänenetz unterbunden werden muss. Ohne entsprechende API wird ein manuelles Eintragen der Firewallregeln vor der temporären Freischaltung des Nutzers notwendig.

Aufgrund der erhöhten Komplexität einer Implementation mittels OPNsense und den bestehenden Nachteilen wurde stattdessen eine Eigenimplementation des Quarantäne-Gateways umgesetzt.

8.3. Weboberfläche - Administrationsseiten

Zur Verwaltung der Nutzerquarantäne stehen mehrere Seiten im Administrationsbereich zur Verfügung:

- *Mein Profil* Hier wird dargestellt, welche Status aktuell für den Nutzer gelten. Die Meldungen sind zusätzlich mit Informationen wie Begründung, Start- und Ablaufdatum angereichert.
- *Verwarnen/ Sperren* Auf dieser Seite kann der Nutzer manuell in den Quarantänestatus versetzt werden.

- *Quarantäne* Es kann eingesehen werden, welche Freischaltungszeiten für den Nutzer noch verfügbar sind und wann die bestehenden Freischaltungen aktiviert wurden. Zusätzlich können hier weitere Zugriffskontingente hinzugefügt werden.

8.4. Weboberfläche - Captive Portal

Jakob Döring

Die Webseite mit der Captive-Portal-Funktionalität wird auf dem Webserver bereitgestellt, auf dem auch die öffentlich erreichbare Webseite des CSN bereitgestellt wird. Die Entwicklung der Webseite wird mit dem Webframework *Django* mit der Programmiersprache *Python* durchgeführt.

Unter dem URL *https://www.csn.tu-chemnitz.de/quarantine* wird eine Webseite bereitgestellt, auf der der Nutzer Informationen zu dem Vorfall bekommt, der zur Verschiebung in das Quarantänenetz geführt hat. Die verschlüsselte MAC-Adresse des Nutzers, die auf dem Gateway ermittelt wird, wird als GET-Parameter übermittelt. Die MAC-Adresse wird entschlüsselt und der entsprechende Nutzer wird ermittelt. Die noch zur Verfügung stehenden Zugriffsberechtigungen werden geladen und dem Nutzer die längste verfügbare Zeitspanne zur Freischaltung angeboten. Zusätzlich werden Informationen zum weiteren Freischaltungs- und Bereinigungsprozess des entsprechenden Nutzersystems angezeigt. Hier kann die Lösung noch zusätzlich mit Informationen angereichert werden, die zur erkannten Bedrohung passen.

Wenn der Nutzer die Freischaltung in Anspruch nimmt, werden mehrere *XML-RPC*-Anfragen an das Quarantäne-Gateway gesendet:

- Alle von Nutzer registrierten MAC-Adressen werden zusammen mit der Länge der Freischaltung übermittelt,
- die vom DFN angegebenen Angriffsziele,
- und ein Befehl zum erneuten Laden der Firewallregeln.

8.5. Netzwerk

Die Einführung einer neuen Routingplattform und einer neuen Virtualisierungsplattform ergeben an der dem Quarantänenetz zugrundeliegenden Netzwerkinfrastruktur einige Änderungen:

- Alle Nutzer, die im Quarantänenetz sind, befinden sich in gleichen VLAN, auch wenn sie vorher in unterschiedlichen Hausnetzen waren. Diese Maßnahme redu-

ziert den Konfigurationsaufwand für die Core-Router und die Virtualisierungsserver.

- Nach dem initialen Herstellen der Verbindung (wechseln des VLANs) bekommen die Nutzer bei ihrer nächsten DHCP-Anfrage eine Adresse aus einem in *RFC1918* benannten Netz und nicht mehr ihre alte Adresse aus dem Hausnetz.

8.6. Router - Quarantäne-Gateway

Aufgrund der in Abschnitt 8.2 beschriebenen Unzulänglichkeiten selbst mit der Lösung, die die besten Überschneidungen zu den Anforderungen darstellte, im speziellen *OPNsense*, fand eine Eigenentwicklung einer Router/Gateway-Lösung für ein solches Netz mit Captive-Portal-Nutzer-Schnittstelle statt. Darunter ist zu verstehen, dass nur die zugrundeliegenden Komponenten des Netzanschlusses der Endgeräte durch das Gateway bereitgestellt werden, und nicht zwingend Notwendiges, wie beispielsweise die Nutzeroberfläche des Captive Portals, nicht bereitgestellt wird, sodass es möglichst leicht wiederverwendbar und integrierbar ist. Folgende Komponenten bilden dabei das Gateway:

DHCP Für die IP-Adressierung von Geräten im Quarantäne-Netz verteilt eine DHCP-Server-Instanz der weitverbreiteten *ISC*-Implementierung aus einem Adresspool private Adressen nach *RFC1918*.

Firewalling Eine auf *iptables* sowie *ipset* basierende Firewall führt über Methoden der *nat*- sowie *filter*-Tabellen folgendes durch:

- Umleitung von nicht freigeschalteten Nutzern zu einem Server, der HTTP zu der Website des Captive-Portals umleitet
- Freigabe nur von speziellen Zielen bevor man freigeschaltet ist oder während der Freischaltung in das eigene Netz
- Dynamische Sperrung von gemeldeten Angriffszielen
- Maskieren der privaten Adressen der Endgeräte
- Umleitung der DNS-Anfragen auf eigene Resolver

HTTP-Umleitung Umleitung von HTTP-Anfragen zum Captive Portal

Management API Konfiguration der Gateways hinsichtlich freizuschaltender Nutzer sowie zur dynamischen Sperrung von gemeldeten Angriffszielen mit *XML RPC*

Die letzten drei Komponenten sind unter Apache-2.0-Lizenz veröffentlicht wurden.[1]

8.6.1. Firewall

Die Firewall stellt die Kernkomponente des Gateways dar, welche, abhängig oder unabhängig vom Status des Nutzers – Internetzugang freigeschaltet oder nicht – in den OSI-Layern 3 und 4 in die Netzkommunikation eingreift. Durch den späteren Einsatz auf Linux-Systemen basiert die Firewall auf den standardmäßig mitgelieferten Paketfilter iptables. Dieser ist in verschiedene Tabellen untergliedert, unter anderem die hier genutzte *nat*-Tabelle für die Umsetzung von Netzadressen, bspw. für den Internetzugang von Netzgeräten mit privaten nicht-gerouteten Adressen sowie zum Umleiten von Diensten, sowie der *filter*-Tabelle zur Beschränkung sowohl des Internetzugangs von Nutzern sowie der Absicherung des Gateways. Innerhalb dieser Tabellen gibt es verschiedene Chains, welche zu unterschiedlichen Zuständen der Verarbeitung der Netzwerkpakete durchlaufen werden, darunter vor sowie nach der Entscheidung über das Routing (*PRE-* und *POSTROUTING*) sowie der Entscheidung über das Ziel – lokal terminierend in der *INPUT*- und weiter geroutet in der *FORWARD*-Tabelle. Weiterhin werden eigene zusätzliche Chains angelegt, welche die Verarbeitung von Paketen mit einer gewissen Eigenschaft zusammenfassen, darunter bspw. *FORWARD_ACTIVATED_USERS*, welches die Regeln für das Forwarding von Datenpaketen beschreibt, welche von Nutzern stammen, die den Internetzugang freigeschaltet haben. Zur komfortablen Parametrisierung dieser Firewall-Regeln wird sich der Komponente *ipset* bedient, welche es ermöglicht gegen eine Liste von sowohl MAC- als auch IP-Adressen zu überprüfen und bei Übereinstimmung bestimmte Aktionen auszuführen.

DNS-Umleitung

Für die Umleitung der DNS-Anfragen zu eigenen Servern zur Vermeidung von Tunneln durch DNS-Traffic, findet folgende Destination-NAT-Regel (Übersetzung im Paketziel) Anwendung:

Listing 8.1: DNS Umleitung

```
iptables -t nat -A PREROUTING -i ens4 -p udp --dport 53
                        -j DNAT --to-destination 134.109.83.200:53
```

[1] https://github.com/camelusferus/cpob

Hier wird jedweder Traffic zu einem UDP-Port 53 zu einem vom *CSN* betriebenen DNS-Server weitergeleitet. Es spielt auch keine Rolle, ob der Nutzer die DNS-Server aus den DHCP-Parametern nimmt, welche ebenfalls dorthin zeigen, oder selbst einen, wie bspw. die großen Öffentlichen von *Google* oder *Cloudflare* konfiguriert. Seine Anfragen werden immer die CSN-Server erreichen, die ihm auch eine Antwort bereitstellen.

Sperrung von nicht freigeschalteten Nutzern & Umleitung HTTP-Traffic

Wenn ein Nutzer im nicht freigeschalteten Zustand, also ohne aufgeführter MAC-Adresse im *ipset*, versucht, auf Seiten zuzugreifen, die nicht in diesem Zustand auch erreichbar sein sollen, wie unter anderem die Seite des Captive Portals, wird dieser auf einen speziellen Dienst umgeleitet, welcher HTTP-Anfragen mit einem *Redirect* zum Captive Portal beantwortet.

Listing 8.2: HTTP-Umleitung mittels NAT

```
iptables -t nat -A PREROUTING -i ens4
   -m set ! --match-set activated_quarantine_macs src
   -j PREROUTING_DEACTIVATED_USERS
iptables -t nat -A PREROUTING_DEACTIVATED_USERS
   -p tcp -m multiport --dports 80,443,143,993,587,589,465
   -m set --match-set important dst -j RETURN
iptables -t nat -A PREROUTING_DEACTIVATED_USERS
   -d 134.109.83.133/32 -p tcp -m multiport
   --dports 80,443 -j RETURN
iptables -t nat -A PREROUTING_DEACTIVATED_USERS
   -p tcp --dport 80
   -j DNAT --to-destination 10.128.0.254:80
```

Jene Regeln beschreiben aber nur die Veränderung von Zieladressen der Pakete. Netzverkehr von gesperrten Nutzern, welcher nicht mit einer der letzten Regeln übereinstimmt, wird nicht umgeleitet, und würde somit einen Netzzugang ermöglichen. Zur Verhinderung dessen, werden solche Szenarien im Filterbereich der Firewall verhindert, in dem nur Verbindungen zu expliziten Zielen erlaubt werden. Die restlichen Verbindungen werden blockiert.

Listing 8.3: Nutzerfreischaltung

```
iptables -A FORWARD
   -m set ! --match-set activated_quarantine_macs src
   -j FORWARD_DEACTIVATED_USERS
iptables -A FORWARD_DEACTIVATED_USERS -p tcp
   -m multiport --dports 80,443,143,993,587,589,465
   -m set --match-set important dst -j ACCEPT
iptables -P FORWARD DROP
```

Private Adressen maskieren

Für die Anbindung der Geräte im Quarantänenetz ans Internet werden die Quelladressen durch die externe IP des Gateways ersetzt.

Listing 8.4: Adressmaskierung

```
iptables -t nat -A POSTROUTING -s 10.128.0.0/24 -j MASQUERADE
```

Sperrung gemeldeter Angriffsziele

Die vom *DFN* gemeldeten Angriffsziele werden durch das Firewalltool in ein *ipset* eingelesen, welches für die Blockierung von Verbindungen zuständig ist.

Listing 8.5: Sperrung von Angriffszielen

```
iptables -A FORWARD -m set --match-set dfn_targets dst -j DROP
```

Sicherung des eigenen Netzes gegenüber Nutzer mit erweitertem Netzzugriff

Um das eigene Netz gegen mögliche Angriffe von noch nicht gereinigten Endgeräten aus dem Quarantänenetz zu schützen, werden nur spezielle Ziele im eigenen Netz für freigeschaltete Nutzer erlaubt und das restliche Netz davon abgetrennt und durch die sequenzielle Abarbeitung von Regeln unbegrenzter Netzzugang nur außerhalb des eigenen Netzes erlaubt.

Listing 8.6: Netzsicherung

```
iptables -A FORWARD
   -m set --match-set activated_quarantine_macs src
   -j FORWARD_ACTIVATED_USERS
iptables -A FORWARD_ACTIVATED_USERS -p tcp
   -m multiport --dports 80,443,143,993,587,589,465
   -m set --match-set internals dst -j ACCEPT
iptables -A FORWARD_ACTIVATED_USERS
   -m set --match-set pub_net dst -j DROP
iptables -A FORWARD_ACTIVATED_USERS -p tcp
   -m multiport --dports 80,443,143,993,587,589,465 -j ACCEPT
```

Sicherung des Quarantäne-Gateways

Für die Sicherheit des Quarantäne-Gateways werden Verbindungen zur Konfiguration nur von ausgewählten Gegenstellen angenommen sowie Verbindungen zu dem Server-Dienst, welcher nicht freigeschaltete Nutzer an das Captive Portal weiterleitet.

Listing 8.7: Eigenschutz des Quarantäne-Routers

```
iptables -t filter -A INPUT -i ens3 -p tcp
   -m multiport --dports 22,2000,8090
   -m set --match-set cfg_access src -j ACCEPT
iptables -t filter -A INPUT -i ens4 -p tcp
   -m tcp --dport 80 -j ACCEPT
```

8.6.2. HTTP-Umleitung und Clientseitige Captive-Portal-Erkennung

Wie auf Seite 13 beschrieben, prüfen Endgeräte anhand ihnen bekannter Websites, ob eine Umleitung zu einem Captive Portal stattfindet. Um diese Erkennung auszulösen, werden HTTP-Requests von nicht freigeschalteten Nutzern mittels einer *302-Redirect*-Antwort auf die Seite des Captive Portals weitergeleitet. Dafür läuft auf dem Captive Portal ein in *Python* geschriebener HTTP-Server, auf den durch zuvor genannte *iptables*-Regeln der Nutzer hingeleitet wird. Da die MAC-Adresse des Nutzers für die Identifikation dessen am Captive Portal nötig ist, wird die aus der IP-Adresse der Anfrage sowie *ARP*-Tabelle bestimmt und anschließend verschlüsselt dem Nutzer übergeben, dessen Client diese an das Captive Portal weiterreicht.

Listing 8.8: Server auf Flask-Basis für HTTP-Umleitungen

```
@app.route("/<path:name>", methods=['GET', 'POST'])
def redirect_to_captive_portal(name):
  try:
    request_ip = request.remote_addr
    p1 = subprocess.Popen(["ip", "-4", "n"], stdout=subprocess.PIPE)
    p2 = subprocess.Popen(["grep", request_ip + " "], stdin=p1.stdout,
      stdout=subprocess.PIPE)
    p1.stdout.close()
    neighbor_reply = p2.communicate()[0]
    neighbor_state = neighbor_reply.split()[3]
    if neighbor_state != b'lladdr':
      raise Exception("No layer2_connectivity for getting the MAC address")
    request_mac = neighbor_reply.split()[4]
    if config.has_option('Main', 'mac_encryption_key'):
      password = config['Main']['mac_encryption_key']
      crypto_in = io.BytesIO(request_mac)
      crypto_out = io.BytesIO()
      pyAesCrypt.encryptStream(crypto_in, crypto_out, password, 65536)
      return redirect(config['http_redirector']['portal_url_template'] +
          urllib.parse.quote(base64.b64encode(crypto_out.getvalue())), code=302)
    return redirect(config['http_redirector']['portal_url_template'] +
        request_mac, code=302)
  except:
    abort(500)
```

Dies ist eine überarbeitete Fassung im Vergleich zur ersten produktiven Inbetriebnahme im *CSN* mit einer Fassungen basierend auf Basis von Pythons *HTTPServer*. Auch bei grundsätzlich identischer Logik zeigte sich seltsames Verhalten des Umleite-Servers hin bis zu einem Ausfall bei gewissen HTTP-Requests durch Nutzergeräte. Mit Wechsel zu einem Webframework, welches für produktiven Einsatz konzipiert ist, ist davon auszugehen, dass derartige Effekte in Zukunft vermieden werden.

8.6.3. XML RPC API

Da das Gateway auf Eingaben von außen angewiesen ist, welche dieses im Betrieb konfigurieren, wird eine API mittels *XML RPC* bereitgestellt, welche folgende Endpunkte zur Verfügung stellt:

reload Anwendung der zuletzt eingespielten Änderungen

block Sperrung von möglichen Ziel-IPs

- Parameter:
  addresses: ein String aller von nun an zu sperrenden IPs

activate Freischalten eines Nutzers für eine bestimmte Zeit

- Parameter:
  mac_address: MAC-Adresse des freizuschaltenden Nutzers
  minutes: Dauer der Freischaltung in Minuten

clear_activations Löschen aller Freischaltungen

9. Kritische Betrachtung

Die in der Projektvision angestrebten technischen Basisfunktionen sowie einer Verbesserung des Prozesses wurden bei der Erstellung der Arbeit erreicht. Die Implementierung war etwas schwieriger als vermutet, da die Benutzung einer zumindest teilweise vorgefertigten Lösung gescheitert ist. Für weitere Verbesserungen könnten noch folgende Punkte betrachtet werden:

- Netzwerkumschaltung ohne Unterbrechung

  Aktuell werden Netzwerkadressen auf einem im RFC1918 beschriebenen Netzwerk für das Quarantänenetz genutzt. Ein Nutzerrechner muss also beim Umschalten nicht nur eine neue ARP-Anfrage für das Gateway stellen, sondern eine vollständige neue Netzwerkkonfiguration vom DHC- Server beziehen. Das kann u. U. 7/8 der DHCP-Lease-Zeit brauchen, die aktuell auf 10 Minuten eingestellt ist. Der Wechsel zu einem Netzwerk aus dem RFC1918 ist deshalb erfolgt, weil nur noch ein VLAN für alle Geräte eingesetzt wird, die sich im Quarantänestatus befinden. Hier könnte allerdings evaluiert werden, ob es möglich ist, in diesem VLAN mehrere Gateways mit unterschiedlichen Subnetzen bereitzustellen, damit die Clients ihre Netzwerkkonfiguration behalten können.

- OCSP

  Das Online Certificate Status Protocol kann überprüfen, ob das Zertifikat einer per HTTPS aufgerufenen Webseite von der ausstellenden Certificate Authority zurückgezogen/ invalidiert wurde. Diese Verifikation kann zu einem Problem werden, da die Webseite des Captive Portal per HTTPS ausgeliefert wird, aber eine Verbindung zu den Validierungsservern für den Client nicht möglich ist. Es muss evaluiert werden, wie sich Browser verhalten, wenn die Validierung fehlschlägt und ob das Captive Portal bei einer fehlgeschlagenen Validierung weiterhin erreichbar bleibt.

- Nutzerfeedback

  Aktuell wurde noch kein Feedback von betroffenen Nutzern eingeholt, ob die Prozesse und verfügbaren Optionen zur Bereinigung der Systeme ausreichend und verständlich erklärt sind. Da aktuell wenige Nutzer von Virenbefall betroffen sind, ist die Menge der für eine Umfrage zur Verfügung stehenden Personen

zudem sehr gering. Eine längerfristige Auswertung der Servicequalität ist jedoch sicherlich sinnvoll.

- Prozess

  Aktuell wird den Nutzern empfohlen, eine beliebige Antivirensoftware auf ihren Systemen zu installieren und eine Überprüfung durchzuführen. Da die Entwicklung von Viren jedoch immer vor der Entwicklung von Erkennungsmustern für ebenjene liegt, dauert es häufig recht lange, bevor Viren auf den Nutzersystemen überhaupt erkannt und bereinigt werden können. Die Nutzer erschweren zudem durch den Einsatz von NAT die genaue Eingrenzung auf ein bestimmtes Endgerät, sodass häufig alle Geräte gescannt oder im Zweifelsfall neu aufgesetzt werden müssen.

- Captive-Portal-RFCs jetzt implementiert

  Die im Kapitel „Bisherige und aktuelle Entwicklung“ erwähnten RFCs zur Erkennung von Captive Portals wurden in „iOS 14“ und „macOS Big Sur“ implementiert. Zwar sind die klassischen Erkennungsverfahren auch noch aktiv, eine Implementierung des RFC7710 im DHCP-Server des CSN ist jedoch wünschenswert.

Literaturverzeichnis

[1] Ahmad, Muhammad Aminu und Steve Woodhead: *Containment of fast scanning computer network worms.* In: *International Conference on Internet and Distributed Computing Systems*, Seiten 235–247. Springer, 2015.

[2] Bassey, Donatus Enang, Bernard Okon und Remigus Umunnah: *The Security Implications of Virtual Local Area Network (VLAN), Niger Mills, Calabar, Nigeria.* International Journal of Scientific & Engineering Research (IJSER), 7(3):1187–1194, 2016.

[3] Bomhoff, Matthijs, Casper Joost Eyckelhof, Remco van de Meent und Aiko Pras: *Quarantine Net: design and application.* In: *9th IFIP/IEEE International Symposium on Integrated Network Management 2005.* IEEE, 2005.

[4] Koht-Arsa, Kasom, Anan Phonphoem und Surasak Sanguanpong: *Architectural design for large-scale campus-wide captive portal.* In: *43rd Annual 2009 International Carnahan Conference on Security Technology*, Seiten 72–76. IEEE, 2009.

[5] Kumari, Warren, Olafur Gudmundsson, Paul Ebersman und Steve Sheng: *Captive-Portal Identification Using DHCP or Router Advertisements (RAs).* RFC 7710, RFC Editor, December 2015.

[6] Larose, Kyle, David Dolson und Heng Liu: *Captive Portal Architecture.* Internet-Draft draft-ietf-capport-architecture-09, Internet Engineering Task Force, August 2020. Work in Progress.

[7] Li, Pele, Mehdi Salour und Xiao Su: *A survey of internet worm detection and containment.* IEEE Communications Surveys & Tutorials, 10(1):20–35, 2008.

[8] Marques, Nuno, André Zúquete und João Paulo Barraca: *Integration of the Captive Portal paradigm with the 802.1 X architecture.* arXiv preprint arXiv:1908.09927, 2019.

[9] Moore, David, Colleen Shannon, Geoffrey M Voelker und Stefan Savage: *Internet quarantine: Requirements for containing self-propagating code.* In: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE*

Computer and Communications Societies (IEEE Cat. No. 03CH37428), Band 3, Seiten 1901–1910. IEEE, 2003.

[10] Nottingham, Mark: *Captive Portals Problem Statement.* Internet-Draft draft-nottingham-capport-problem-01, Internet Engineering Task Force, April 2016. Work in Progress.

[11] Ohmori, Motoyuki: *A Case Study of Captive-Portal Detection for Web Authentication on Wired LAN in a Campus Network.* In: *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Seiten 1–4. IEEE, 2019.

[12] Poger, Elliot und Mary Baker: *Secure Public Internet Access Handler (SPINACH).* In: *USENIX Symposium on Internet Technologies and Systems*, 1997.

[13] Sanguanpong, Surasak und Kasom Koht-Arsa: *A design and implementation of dual-stack aware authentication system for enterprise captive portal.* In: *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, Seiten 118–121. IEEE, 2013.

[14] Watanabea, Yoshiaki, Makoto Otanib Hirofumi Etob Kenzi Watanabea und Shin-ichi Tadakib: *Control of users' sessions and IP dual stack for captive portal type of authentication system.*

[15] Williamson, Matthew M: *Throttling viruses: Restricting propagation to defeat malicious mobile code.* In: *18th Annual Computer Security Applications Conference, 2002. Proceedings.*, Seiten 61–68. IEEE, 2002.

[16] Wong, Cynthia, Chenxi Wang, Dawn Song, Stan Bielski und Gregory R Ganger: *Dynamic quarantine of Internet worms.* In: *International Conference on Dependable Systems and Networks, 2004*, Seiten 73–82. IEEE, 2004.

[17] Yasutome, Yoshio, Tomoo Adachi und Takayuki Yoshida: *Quarantine Network in the Age of Internal Governance.* NEC technical journal, 2(1):17–22, 2007.

Anhang A.

Bewertung existierender Captive-Portal-Anwendungen

Im Folgenden werden verschiedene existierende Captive Portal/Router-Produkte verglichen. Der Vergleich fand im Februar 2020 in Vorbereitung der Implementierung statt. Die Bewertung basiert auf Auswertung von Funktionalitäten, die Entwickler oder Hersteller auf beispielsweise ihren Internetauftritten bewarben. Die Software mit dem besten Ergebnis ist grau hinterlegt.

Anforderung	**Wifidog**	**Packetfence**	**Coova Chili**	**pfSense**	**OPNsense**
n/a Lizenz	GPL	GPL	GPL	Apache-2.0	BSD Lizenz / kommerzielle Zusatzmodule
n/a Aktualität / Letztes Release	Oktober 2015	Januar 2020	November 2019	Mai 2019	Februar 2020
n/a Dokumentation	Sehr schlecht	Gut	Mäßig	Gut	Sehr gut
3	Ja	Ja	Ja, dieses läuft aber dann außerhalb, CoovaChili ist nur der Controller	Ja	Ja
3.1					Ja - Templating
3.2 und Unterpunkte	Ja - Iframe => bringt Probleme bzgl. Individualisierung	Ja, vollständig individuelle Captive-Portal-Seiten sind möglich	Eigens entwickeltes CP	Ja - Preauth Redirection	Ja - Templating

Anforderung	Wifidog	Packetfence	Coova Chili	pfSense	OPNsense
3.4	Nein - Viele Einstellungen lassen sich nur über die Weboberfläche vornehmen		Ja, Einstellung über Linux-typische Config-Files	Ja - Vergleichbare ansible-Playbook existieren. Konfiguration über xml möglich → provisionierbar	Ja - Vergleichbare ansible-Playbook existieren. Zusätzliche Existenz einer API. Konfiguration über xml möglich → provisionierbar
4	Nein, Zugangssteuerung immer durch Nutzerauthentifizierung mit Nutzername/ Passwort ohne Kontingente	Möglich - Voucher über externe API	Jein, Konfiguration für den unautorisierten Zustand möglich, restliches durch weitere Software auf dem Gateway möglich, wie iptables - Zeitsteuerung durch Authentifizierung gegen Captive Portal und Radius	Ja - Stateful firewall, Captive Portal mit Timeouts, Voucher Management, co.	Ja - Stateful firewall, Captive Portal mit Timeouts, Voucher Management, co.
4.2	n.a. wg. 4	Möglich - Externe API	schwer möglich	Ja - bspw. durch Voucher	Ja - bspw. durch Voucher
4.3	Nein	Ja	Siehe 4.	Ja	Ja
4.4	Nein	Ja	Siehe 4.	Ja - externe Firewall möglich	Ja - Firewall + API
4.5	Nein	Ja	Siehe 4.	Ja	Ja
n/a	-		-	SNMP / Netflow / ...	SNMP / Netflow / ...

Tabelle A.1.: Bewertung existierender Captive-Portal-Anwendungen