Integration von Endnutzer Netzwerkgeräten in Zentralisierte Protokoll Management Systeme am Beispiel der FRITZ!Box

Bachelorarbeit
zur Erlangung des Grades Bachelor of Science
des Fachbereichs Informatik und Medien der
Technischen Hochschule Brandenburg

vorgelegt von:
Max Nowak

Betreuer: Prof. Dr. Michael Pilgermann
Zweitgutachter: Tom Neubert

Brandenburg an der Havel, 18. April 2023

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Motivation

Das Sammeln, Verwalten und Identifizieren von sicherheitsrelevanten Ereignissen ist ein fester Bestandteil der meisten IT-Sicherheitsrichtlinien. Insbesondere ermöglicht es Unternehmen in Echtzeit Bedrohungen innerhalb des eigenen Netzwerkes zu erkennen und auf diese zu reagieren, bevor Schäden entstehen können.[1]

Ein wesentlicher Bestandteil dieses Sicherheitsverfahrens ist es, die auf den verschiedenen Netzwerkgeräten gesammelten Protokolldaten zu zentralisieren und auf einem Log-Server auszuwerten. Zu diesem Zweck wird das standardisierte Protokoll „System Logging Protocol“ oder kurz Syslog verwendet, welches im RFC 5424 definiert ist.[2]

Syslog wird von der meisten Hardware, welche in Firmennetzwerken eingesetzt wird, unterstützt. Durch die Coronapandemie ist der Arbeitsschwerpunkt in vielen Firmen weg aus den Großraumbüros hin zu den Privaträumen der Mitarbeiter verschoben worden.[3] Dadurch werden vermehrt auch die Heimnetzwerke zu möglichen Angriffspunkten auf ein Unternehmen.

Im Kontext von Heimnetzwerken werden allerdings häufig andere Netzwerkgeräte verwendet als in einem Firmennetzwerk. Weit verbreitet in Deutschland sind die Produkte der Marke AVM.[4]

Jedoch unterstützen Diese Syslog nicht nativ und können nur über das TR-064-Protokoll konfiguriert werden.

Mit Hilfe des TR-064-Protokolls sollte es möglich sein, die Logdaten einer FRITZ!Box auszulesen und diese an eine Syslog-Umgebung weiterzuleiten.

Wie eine solche Integration einer FRITZ!Box in ein zentralisiertes Protokoll Management System möglich ist und welche Informationen die erhaltenen Logdaten liefern, soll in der Bachelorarbeit eruiert werden.

---

[1] ibm, „What is SIEM?“, zugegriffen 8. Februar 2023, https://www.ibm.com/topics/siem.

[2] ietf, „RFC 5424“, zugegriffen 8. Februar 2023, https://datatracker.ietf.org/doc/rfc5424/.

[3] Hans-Böckler-Stiftung, „Anteil der im Homeoffice arbeitenden Beschäftigten in Deutschland vor und während der Corona-Pandemie 2020 und 2021“, zugegriffen 8. Februar 2023, https://de.statista.com/statistik/daten/studie/1204173/umfrage/befragung-zur-homeoffice-nutzung-in-der-corona-pandemie/.

[4] aetka.de, „AVM“, zugegriffen 8. Februar 2023, https://aetka.de/markenwissen/avm.

2 Recherche

2.1 Syslog

Das Syslog-Protokoll ist im RFC 5424 definiert. Es beschreibt, wie lokale Events von Endgeräten, als Nachricht, über ein IP-Netzwerk an einen Event-Nachrichten-Sammler verschickt werden. Syslog stellt keine Anforderungen an den Inhalt oder dessen Formatierung, sondern nur an die Übertragung der Nachrichten.[5]

2.1.1 Entstehung

Die ersten Versionen des Syslog-Protokolls wurden in den späten 1970er Jahren von Eric Allman an der „University of California at Berkley“ entwickelt.[6]
Während der Zeit seit der Erstveröffentlichung des Syslog-Protokolls bis 2001 wurde dieses für viele Systeme portiert und auch in vielen Netzwerkgeräten nativ integriert.
Im August 2001 wurde von Chris M. Lonvick der aktuelle Zustand, insbesondere die Funktions- und Verhaltensweisen von Syslog im informativen RFC 3164 festgehalten. [5]
Um die verschiedenen gebräuchlichen Versionen von Syslog zu vereinheitlichen, wurde im März 2009 von der Internet Engineering Task Force der RFC 5424 erstellt, welcher den Standard für Syslog definiert.

2.1.2 Funktionsweise

Syslog verwendet drei Schichten:
1. „Syslog Inhalt“ enthält den Inhalt der Syslog Nachricht
2. Die „Syslog Anwendung“ verwaltet die Generierung, Interpretierung, das Weiterleiten und Speichern der Nachrichten.
3. Die „Syslog Transport“ Schicht sendet und empfängt die tatsächlichen Daten.

[5] „RFC 3164“ (Cisco Systems, August 2001), https://www.rfc-editor.org/rfc/rfc3164.
[6] „Inductee Eric Allman“, zugegriffen 23. Februar 2023, https://www.internethalloffame.org/inductee/eric-allman/.

Das Syslog-Protokoll baut mit seiner Transport Schicht auf der Transport Schicht des TCP/IP-Referenzmodells auf. Für die tatsächliche physische Übertragung der Daten wurden historisch Protokolle verwendet die mit UDP interoperable sind. Auch UDP und TCP selbst können zur Übertragung verwendet werden, solang die Nachricht des Syslog-Protokolls nicht verändert wird, oder temporäre Änderungen rückgängig gemacht wurden, bevor diese and Syslog übergeben werden. Ein Empfänger einer Syslog Nachricht muss immer eine exakte Kopie des Originals erhalten.

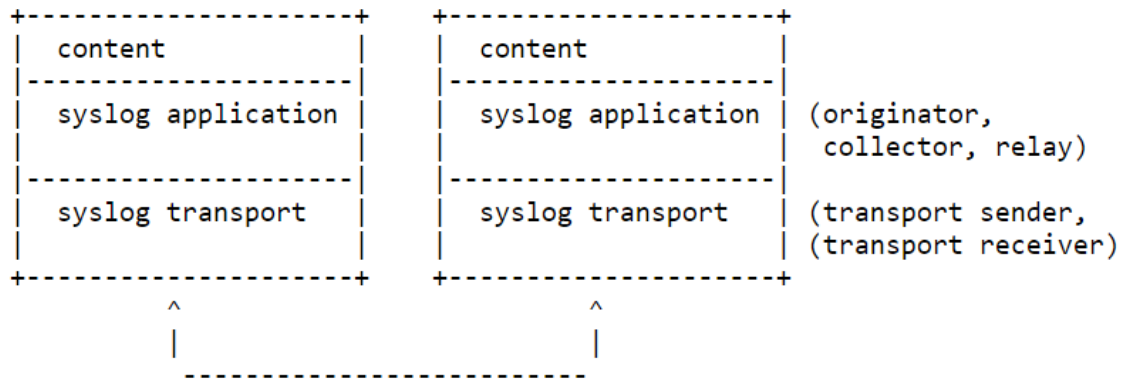

Abbildung 1: Syslog Layers (Quelle: ietf "RFC 5424")

Innerhalb dieser drei Schichten gibt es verschiedene Funktionen, die im Syslog-Protokoll aufgeführt werden. Die Funktionen der Anwendungsschicht bilden dabei ein Netzwerk von Akteuren, mit jeweils spezifischen Fähigkeiten:
In einem solchen Netzwerk gibt es ein oder mehrere „Urheber“, welche Datenströme für das Syslog-Protokoll produzieren. Das können zum Beispiel Switche, Router und ähnliche Netzwerkgeräte sein, die Ihre Logs per Syslog versenden. Zusätzlich gibt es ein oder mehrere „Sammler“ oder auch Syslog-Server, welche die Nachrichten aus dem Netzwerk sammeln, und für weitere Analysen aufbewahren. Schließlich kann es keinen bis mehrere „Relais“ geben, welche erhaltene Informationen weiterleiten. Dabei kann ein einzelnes Gerät auch mehrere dieser Funktionen innehaben. So kann zum Beispiel ein Relais, dass Informationen von einem Router an einen Syslog-Server weiterleitet, auch Informationen über seinen eigenen Status sammeln und verschicken. Dadurch arbeitet es sowohl als Relais für den Router, als auch als Urheber seines eigenen Datenstroms.
Ergänzend gibt es noch zwei weiterer Funktionen, die der Transport-Schicht angehören. Diese verwalten respektive das Senden und das Empfangen der Syslog-Nachrichten, und organisieren die Kommunikation zwischen den

Unterliegenden Transportprotokollen, wie UDP oder TCP, und den darüber liegenden Anwendungs-Akteuren von Syslog.

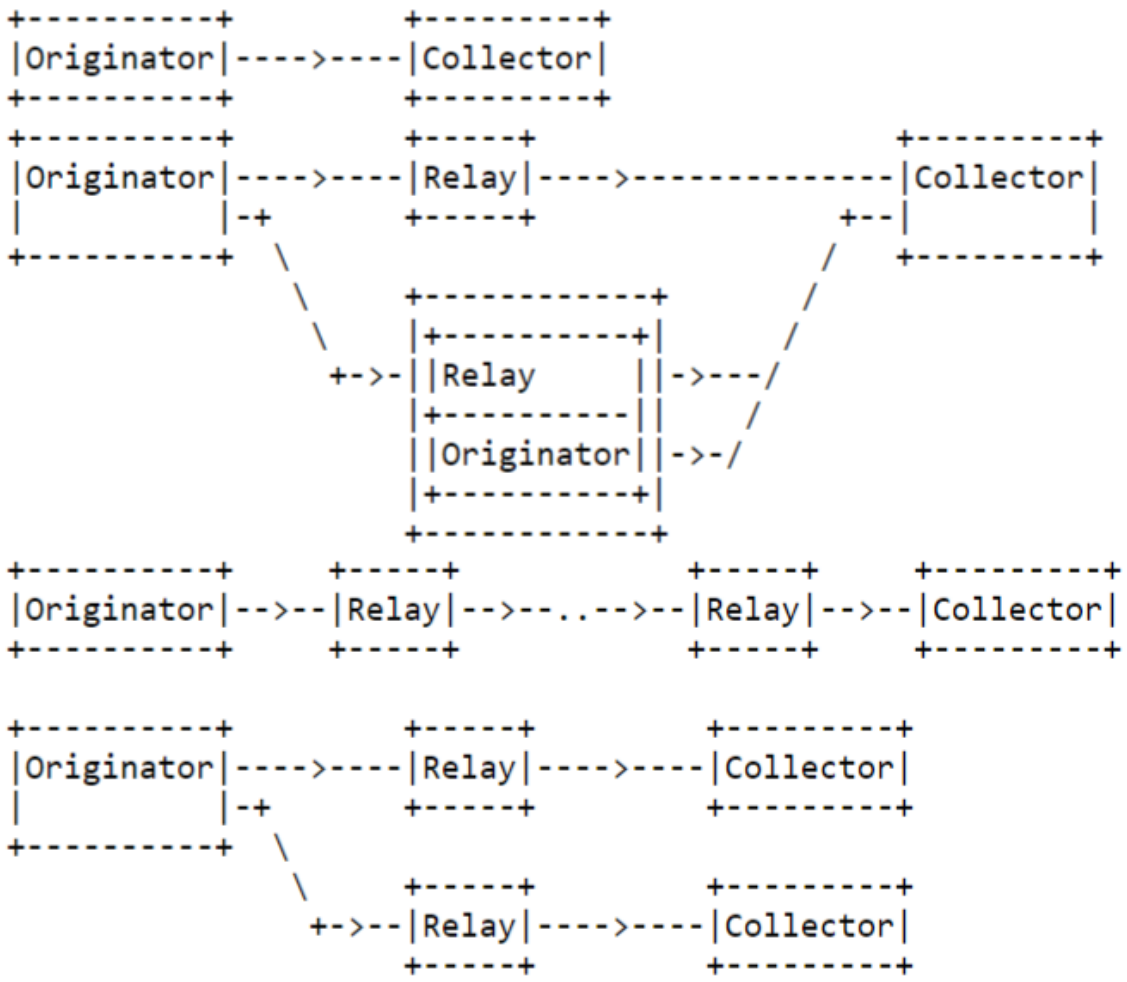

Abbildung 2: Syslog Deployments (Quelle: ietf "RFC 5424")

Schließlich gibt es noch 3 Prinzipien, nach welchen diese Syslognetzwerke arbeiten:

1. Das Syslog-Protokoll ist ein Simplex-Kommunikations-Protokoll und gibt keine Bestätigungen für eingegangene Nachrichten.
2. Urheber und Relais können so konfiguriert werden, dass die gleiche Nachricht an mehrere Sammler und Relais versendet wird.
3. Ein System kann gleichzeitig Urheber, Relais und Sammler sein.[7]

[7] ietf, „RFC 5424“.

2.2 TR-064

TR-064 ist eine 2004 vom Broadband-forum vorgestellte spezifische Implementierung zur LAN-seitigen Konfiguration von Teilnehmernetzgeräten. Dazu werden in TR-064 die „Discovery“ von Teilnehmernetzgeräten, die verwendeten Sicherheitsmechanismen, die verwendete XML-Struktur als auch der Protokoll Stack beschrieben.[8]

2.2.1 Entstehung und Verbreitung

Der Technical Report 064 des Broadband Forums wurde erstmals im Mai 2004 veröffentlicht mit dem Ziel ein einheitliches Protokoll zur LAN-seitigen Konfiguration von Teilnehmer-Endgeräten zu etablieren. TR-064 ist damit das Gegenstück zu TR-069, welches die Konfiguration aus dem WAN beschreibt.
Im August 2015 wurde im „TR-064 Corrigendum 1“ die vorherige Funktionsweise von TR-064 als überholt eingestuft und durch „TR-064 Issue 2“ ersetzt.[9]
Das Broadband Forum ist eine Plattform zur kooperativen Entwicklung von neuen Technologien und industrieweiten Standardisierung in den Bereichen Telekommunikation und Internet. Mitwirkende und Partner des Broadband-Forums sind die führenden Telekomunikations Firmen weltweit, wie die Telekom, China Unicom oder AT&T.[10]
TR-064 wird von AVM aktiv als Feature beworben. Die dabei zugrunde liegende Version von TR-064 ist die veraltete Version aus 2004.[11]

2.2.2 Funktionsweise

Im Sinne der Arbeit und dem besonderen Fokus auf die FRITZ!Box wird hier die Funktionsweise der Implementierung von AVM, an Stelle der allgemeinen Spezifikation aus „TR-064 Issue 1“, beschrieben.

TR-064 beschreibt 3 wesentliche Bestandteile:

---

[8] broadband-forum, „LAN-Side DSL CPE Configuration Corrigendum“, zugegriffen 9. März 2023, https://www.broadband-forum.org/technical/download/TR-064_Corrigendum-1.pdf.
[9] broadband-forum.
[10] broadband-forum, „About BBF“, zugegriffen 9. März 2023, https://www.broadband-forum.org/about-bbf.
[11] "AVM TR-064 - First Steps" (AVM gmbH, 18. November 2022), zugegriffen 9. März 2023, https://avm.de/fileadmin/user_upload/Global/Service/Schnittstellen/AVM_TR-064_first_steps.pdf

1. Discovery, das Finden von TR-064 fähigen Geräten in einem Netzwerk
2. Security, wie Geräte gegen Fehlkonfigurationen geschützt werden und insbesondere die Authentifizierung, um gegen unberechtigte Konfigurierungen eines Gerätes zu schützen.
3. Eine Übersicht der verfügbaren Funktionen und Variablen.

Discovery: Zum Finden von Tr-064 fähigen Geräten wird das Simple-Service-Discovery-Protokoll verwendet.
Dieses sendet per HTTP Request mit der Methode „NOTIFY“ oder „M-SEARCH“ ein Paket an das gesamte Netzwerk.
Dieses Paket muss den spezifischen Search Type, abgekürzt als „ST“, „urn:dslforum-org:device:InternetGatewayDevice:1“ enthalten.
Wenn ein TR-064 fähiges Gerät diese Anfrage erhält muss es auf jene antworten und sich dabei mit einem einmaligen Namen identifizieren. Dieser Name hat die Form: „USN: uuid:739f75f0-a90c-4e42-ac13-2cc42d3c243e“, wobei der 128 Bit hexadezimal String eine UUID bilden, bei welcher die letzten 48 Bit die MAC Adresse des primären LAN Interfaces sind.

```
M-SEARCH * HTTP/1.1
MX: 10
ST: urn:dslforum-org:device:InternetGatewayDevice:1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
```

Abbildung 3: TR-064 Discovery Request (Quelle: broadband-forum "LAN-Side DSL CPE Configuration Corrigendum")

```
HTTP/1.1 200  OK
ST: urn:dslforum-org:device:InternetGatewayDevice:1
EXT:
SERVER: pX/2.0 UPnP/1.0 Server/1.0
USN: uuid:739f75f0-a90c-4e42-ac13-2cc42d3c243e::urn:dslforum-
org:device:InternetGatewayDevice:1
CACHE-CONTROL: max-age=1209600
LOCATION: http://192.168.0.1:51004/devicedesc.xml

Content-Length: 0
```

Abbildung 4: TR-064 Discovery Response (Quelle: broadband-forum "LAN-Side DSL CPE Configuration Corrigendum")

Security: Der wesentliche Teil der implementierten Sicherheitsfunktionen für TR-064 auf AVM-Geräten besteht aus der Authentifizierung von Nutzern mit Hilfe von Nutzername und Passwort. Für AVM-Geräte gibt es drei mögliche Authentifizierungsoptionen:

Nutzername und Passwort, nur Passwort oder kein Passwort.
Für den Fall, dass kein Passwort gesetzt wurde, sind alle Funktionen, abgesehen von „Dial Number“, frei zugänglich.
Wenn hingegen ein Passwort oder ein Passwort und Nutzername festgelegt worden sind, so werden diese per „HTTP authentication using digest“ authentifiziert.
Zusätzlich zur Authentifizierung per http-digest aus TR-064 ermöglicht AVM eine Content-Level-Authentifizierung basierend auf den Ideen für SOAP-Erweiterungen aus SOAPAUTH01, einem Draft der ietf von 2002.
Neben der grundlegenden Authentifizierung sind für einige Funktionen auch eine zwei Faktor Authentifizierung vorgesehen, für welche in der Regel ein beliebiger Knopf an dem Gerät gedrückt werden muss.

Neben der Authentifizierung von Nutzern sieht TR-064 auch eine Möglichkeit zur Verschlüsselung der Kommunikation vor. Die Verwendung dieser ist aber nur für die Funktionen aus „LANConfigSecurity“ vorgeschrieben, welche zum Ändern von Passwörtern verwendet werden können. Verschlüsselte Kommunikation soll mit Hilfe von SSL3.0 oder TLS1.0 und mit vorherigen Anonymous Key Exchange realisiert werden.

Zuletzt bietet TR-064 eine Übersicht über alle Funktionen und Informationen, welche einem Nutzer zu Verfügung stehen. Das Zugrundeliegende Modell baut auf dem UPnP IGD 1.0 Modell auf und erweitert es durch verschiedene Services, wie „DeviceInfo“, „DeviceConfig“ oder „LANConfigSecurity“.
Für diese Arbeit relevant ist ausschließlich „DeviceInfo“, da es über „GetDeviceLog“ eine Funktion zur Verfügung stellt, welche die aktuellen Logdaten des Endnutzergerätes zurückliefert.[12] [13]

[12] „AVM TR-064 - First Steps“ (AVM GmbH, 18. November 2022), https://avm.de/fileadmin/user_upload/Global/Service/Schnittstellen/AVM_TR-064_first_steps.pdf.
[13] broadband-forum, „TR-064 Corrigendum 1“.

```
POST /upnp/control/deviceinfo HTTP/1.1
Host: 192.168.0.115:49000
[…]
SOAPACTION: urn:dslforum-org:service:DeviceInfo:1#GetDeviceLog
[…]

<?xml version="1.0"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<s:Body>
      <u:GetDeviceLog xmlns:u="urn:dslforumorg:service:DeviceInfo:1">
      </u:GetDeviceLog>
</s:Body>
</s:Envelope>
```

Abbildung 5: TR-064 Request an die FRITZ!Box

2.3 Security Event Management

Im routinemäßigen Betrieb eines jeden Unternehmens kommt es regelmäßig zu „Rohereignissen“, welche mit Hilfe der Protokollierung gesammelt und gesichert werden. Mit Hilfe weiterer Informationen, zum Beispiel der Kombination mehrerer Events oder anderer Kontextinformationen, können diese Ereignisse auf sicherheitsrelevante Vorfälle schließen lassen. Diese gesammelten Rohdaten bieten den Schlüssel zum Erkennen und zum Nachvollziehen dieser Ereignisse und sind somit elementarer Baustein einer jeden Sicherheitsinfrastruktur.

Die Grundlage der Mechanismen zur Detektion von sicherheitsrelevanten Ereignissen bildet die Protokollierung. Sie ist ein kontinuierlicher Prozess aus den folgenden drei Bestandteilen:
Im ersten Bestandteil, des Planens, müssen alle Protokollierungsquellen identifiziert werden. Dazu zählen zum Beispiel: Arbeitsrechner, Netzwerkgeräte wie Drucker, IoT Systeme oder auch Infrastruktur Geräte wie Router. Ebenfalls muss geplant werden, wie die erfassten Daten in die Protokollierungsinfrastruktur einzuspeisen sind und welche Datenaufkommen erwartet werden.
Im zweiten Bestandteil, dem Dokumentieren, werden die Ergebnisse des Planens in einer Protokollierungslandkarte dokumentiert. Diese dient als Kommunikationsgrundlage, anhand welcher die Protokollierungsereignisse interpretiert werden.

Der letzte Bestandteil, das Sammeln, beschreibt den automatisierten Prozess, um erzeugte Protokollereignisse von den Quellen an die Protokollinfrastruktur weiterzuleiten und zu speichern. Dieser Bestandteil beinhaltet auch eine kontinuierliche Überwachung der Infrastruktur auf Fehlerzustände.[14] [15]

Das Security-Event-Management bildet zwei wesentliche Aufgaben eines vollen Security-Information-and-Event-Management-Systems. Es ist zuständig dafür die Ereignisse aus den verschiedensten Datenquellen zu extrahieren. Solche Datenquellen können zum Beispiel Logdaten von Firewalls, Routern, Cloudsystemen oder Endnutzergeräte sein. Aber auch Treffer von Antiviren Softwares, mitgeschnittene Datenströme aus dem Netzwerk oder auch Alarmierungen durch Endpoint-Detection-and-Response-Systeme können Datenquellen sein, welche von Logging-Systemen gesammelt werden.
Neben dem puren Sammeln der Rohdaten ist das Security-Event-Management ebenfalls dafür zuständig die Daten zu speichern, zu filtern und die relevantesten Daten an weitere Analyse- beziehungsweise Korrelationsmechanismen weiterzugeben. Dadurch wird es ermöglicht, in einem effizienten Workflow frühzeitig Anzeichen für Bedrohungen oder Angriffe zu finden, ohne von der Masse an Daten überwältigt zu werden. Und dennoch, sofern es notwendig wird, zurück zu den Rohdaten gehen zu können, um per Hand in diesen nach weiteren Zusammenhängen zu suchen.

Das Sammeln der Daten kann entweder auf dem zu überwachenden System selbst passieren oder auf einem unabhängigen Gerät. Diese datensammelnde Komponente wird in der Regel entweder Agent oder Sammler genannt. Beim Sammeln von Daten auf dem Host, kann ein Agent direkt auf die Logdaten, die aktiven APIs oder die zugängigen Dateien zugreifen, um relevante Daten zu erhalten. Wenn der Agent hingegen auf einem anderen Gerät läuft, kann er ein oder mehrere Geräte entweder aktiv nach neuen Daten fragen oder passiv die ihm gesendeten Daten sammeln. Dazu können verschiedene Protokolle wie Syslog oder RESTful APIs verwendet werden. Unabhängig von dem System, auf dem ein solcher Sammler arbeitet, sollte dieser sicherstellen, dass die erhaltenen Daten zeitnah und ohne Verluste ankommen. Dazu können

[14] BSI, „Mindeststandard des BSI zur Protokollierung und Detektion von Cyber-Angriffen“, zugegriffen 9. März 2023, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Mindeststandards/Mindeststandard_BSI_Protokollierung_und_Detektion_Version_1_0a.pdf?__blob=publicationFile&v=5.
[15] Michael Alexander, *Netzwerke und Netzwerksicherheit: das Lehrbuch*, 1. Aufl (Heidelberg: Hüthig, 2006).

verschiedene Strategien wie „filtering“, „batching“, „deduplication“, „caching“, „flow control“ oder „priorization“ angewendet werden.

In den meisten Security-Event-Mangement-Systemen werden die Daten lokal an einem zentralen Punkt gespeichert. Die Ansprüche an einen solchen Datenspeicher sind sehr hoch und verlangen sehr schnelle Datenspeicherung und Datenabfragen. In manchen Fällen sind solche Datenspeicher gezwungen, mehr als 100‘000 Ereignisse pro Sekunde zu verarbeiten. Zusätzlich zu der hohen Geschwindigkeit der Verarbeitung müssen die gespeicherten Daten oftmals für mehrere Jahre gespeichert werden. Um sowohl eine hohe Verarbeitungsgeschwindigkeit als auch ein hohes Volumen zu ermöglichen, und dabei die Kosten minimal zu halten, wird mit zwei oder mehr Speicherstufen gearbeitet. Wobei eine „Heiße“ für die aktuellen Daten verwendet wird. Diese bietet die höchste Verarbeitungsgeschwindigkeit, aber hat nur sehr begrenztes Volumen. Und es wird eine weitere „Kalte“ für Daten verwendet, die voraussichtlich nicht in naher Zukunft aufgerufen werden. Diese Stufe ist sehr viel langsamer, hat dafür aber ein um mehrere Größenordnungen größeres Volumen.[16]

[16] Kathryn Knerler, Ingrid Parker, und Carson Zimmerman, *Cybersecurity Operations Center*, 2te Auflage (The MITRE Corporation, o. J.).

3 Versuchsaufbau

Zum Entwickeln und Testen eines Systems, welches die FRITZ!Box in eine bestehende Syslog-Umgebung einbindet, werden 3 Geräte benötigt.

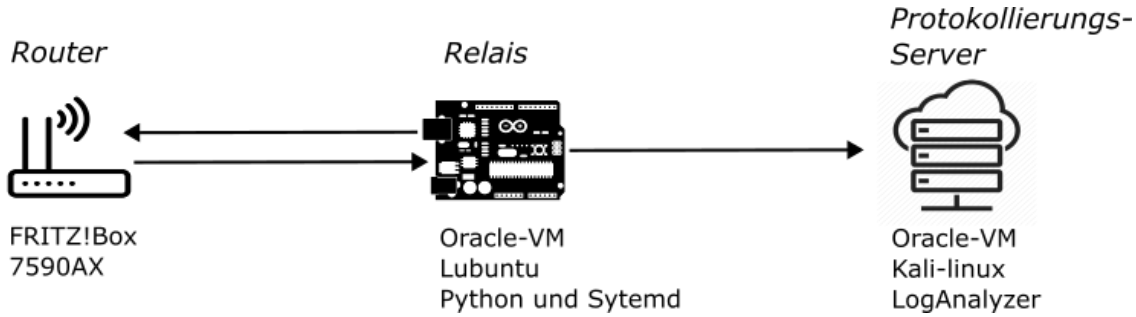

Abbildung 6: Versuchsaufbau

Zuerst wird eine FRITZ!Box benötigt, welche die Grundlage für den Versuchsaufbau bietet. Sie stellt als Host für das TR-064-Protokoll die Schnittstelle für die Kommunikation, als auch die Logdaten bereit, welche an Syslog weitergegeben werden sollen.
Um die Konfiguration und somit auch die Abfrage der Logdaten per TR-064 zu ermöglichen, muss in der Konfiguration der FRITZ!Box die Option „Zugriff für Anwendungen zulassen“ aktiviert werden. Ferner muss entweder die Authentifizierung deaktiviert oder Anmeldedaten für einen Nutzer mit mindestens Konfigurationsrechten bereitgestellt werden. Letztlich wurde die zum Testen verwendete FRITZ!Box 7590AX im Heimnetzwerk als „IP-Client“ registriert. Dieses erlaubt der FRITZ!Box eine bestehende WAN-Verbindung mitzubenutzen, anstatt diese selbst aufzubauen. Diese Einstellung wäre für den Versuch nicht zwingend nötig. Da aber der WAN-Anschluss per Kabel erfolgt, welches die FRITZ!Box 7590 AX nicht unterstütz, wird die Internetverbindung durch einen weiteren Router aufgebaut. Dieser Router ist nicht Teil des Versuches und stellt nur die Internetverbindung her. Alle relevanten Funktionen der FRITZ!Box bleiben auch im Betrieb als IP-Client vorhanden.

Als zweites Gerät in der Testumgebung wird ein Syslog-Protokollierungsserver benötigt, welcher als Endpunkt für die Systemprotokolle funktioniert. Dieser Server erlaubt es zu verifizieren, dass die von der FRITZ!Box erhaltenen Logdaten vollständig in die Syslog-Umgebung weitergeleitet werden.
Zu diesem Zweck wurde eine virtuelle Maschine mit der Hilfe von Oracle VM VirtualBox erstellt. Auf dieser VM wurde Kali Linux, als typisches Beispiel für eine häufig verwendete Linux Distribution, installiert. Die VM wurde außerdem so konfiguriert, dass sie als direkter Teilnehmer im lokalen Netzwerk agieren kann, indem die Netzwerkschnittstelle der VM als Netzwerkbrücke gewählt wurde. Schließlich wurden in der Kali Umgebung ein Apache Server, PHP 8.1 und „RSyslogd“ 8 installiert, sodass mit Hilfe von „LogAnalyzer“ alle eingehenden Syslog-Nachrichten geprüft werden können.
Zuletzt wird ein drittes Gerät im Netzwerk benötigt, welches als Host für den Relais zwischen TR-064 und Syslog, beziehungsweise als Relais zwischen der FRITZ!Box und dem Protokollserver dienen kann. Typischerweise würde ein solcher Relais-Client in einem Netzwerk auf einem Nanorechner mit minimaler Konfiguration laufen. Um ein solches Gerät zu simulieren, wurde eine VM mit Lubuntu als Betriebssystem erstellt. Lubuntu ist eine lightweight Variante von Ubuntu mit minimaler Konfiguration und somit ein typisches Betriebssystem für Nanorechner wie den Raspberry PI. Die VM wurde ebenfalls als echter Netzwerkteilnehmer konfiguriert, um die Kommunikation zwischen der FRITZ!Box, dem Relais und dem Protokollserver zu ermöglichen. Zur Umsetzung der Implementierung muss auf dem Client zusätzlich zu einer beliebigen minimalen Konfiguration „python3“, „pip“ und „Systemd“ vorhanden sein.

4 Implementierung

Eine Implementierung, welche eine FRITZ!Box in eine bestehende Syslog Umgebung einbinden soll, muss im Wesentlichen drei Aspekte implementieren:

Erstens muss dieses Programm die auf der FRITZ!Box gespeicherten Log Ereignisse von dieser per TR-064 abfragen und zwischenspeichern.
Zweitens sollte ein solches Programm die erhaltene Menge an Logdaten gegen eine bestehende, persistente Sammlung von bereits abgerufenen Nachrichten abgleichen. Nur neue Nachrichten sollten an den Syslog-Server weitergegeben werden.
Drittens muss sichergestellt werden, dass ein solches Programm automatisiert Fehler meldet und selbständig neu starten kann. Auch bei unerwarteten Ereignissen sollte das System in der Lage sein, begrenzt Fehler selbstständig zu beheben oder mindestens per Syslog über die Fehlerzustände zu informieren.

Die Implementierung dieser drei Aspekte wurde in Python umgesetzt und alle gezeigten Quellcodeauszüge beziehen sich auf diese Version. Eine Umsetzung in vergleichbaren Skriptsprachen wäre austauschbar ebenfalls möglich.

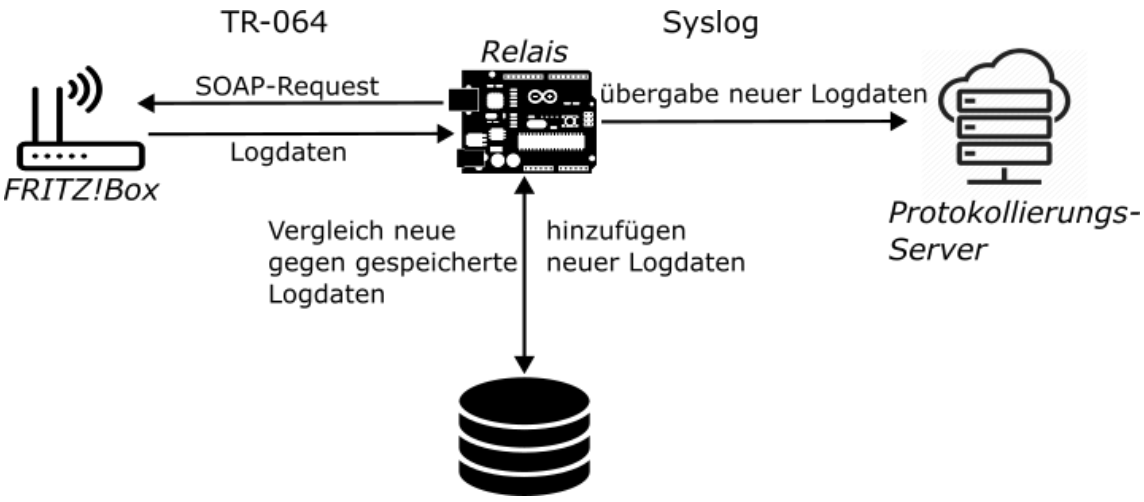

Abbildung 7: Versuchsaufbau mit Datenfluss

4.1 TR-064 Abfrage

Die benötigte Abfrage der Logs an die FRITZ!Box erfolgt mit Hilfe von TR-064, welches auf SOAP-Requests basiert. Dementsprechend muss in Python mit Hilfe der „reqeuests“ Bibliothek eine POST-Abfrage gemacht werden.
Ein solcher POST-Request benötigt in der Regel nur eine HOST-Adresse und einen PORT an welche diese gesendet wird.
AVM spezifiziert allerdings, dass für die meisten Abfragen per TR-064 an die FRITZ!Box eine Authentifizierung stattfinden muss. Wie diese ausfällt, ist abhängig von der Authentifizierungsmethode, welche für die Anmeldung an der Weboberfläche verwendet wird. Typisch ist hier die Authentifizierung per Nutzername und Passwort, welche mit Hilfe von „HTTPDigest“ authentifiziert werden. Zusätzlich muss noch die gewünschte Aktion spezifiziert werden, welche vom TR-064 Host durchgeführt werden soll. In diesem Fall, den Inhalt der Log-Datei dem Client zurückzusenden. Dazu muss ein spezieller Header „'SOAPACTION': 'urn:dslforum-org:service:DeviceInfo:1#GetDeviceLog'“ und ein spezifischer Tag im „body“ des SOAP-Payloads mit gesendet werden.[17]

```
POST /upnp/control/deviceinfo HTTP/1.1
Host: 192.168.0.115:49000
[…]
SOAPACTION: urn:dslforum-org:service:DeviceInfo:1#GetDeviceLog
[…]

<?xml version="1.0"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<s:Body>
      <u:GetDeviceLog xmlns:u="urn:dslforumorg:service:DeviceInfo:1">
      </u:GetDeviceLog>
</s:Body>
</s:Envelope>
```

Abbildung 8: POST-Request an die FRITZ!Box

In der Response wird der Inhalt der Logdatei der FRITZ!Box, also alle Ereignisse seit der letzten Trennung vom Strom, zurückgesendet. Diese Ereignisse werden aus der Response per „String Manipulation“ geparst und für die weitere Verarbeitung in einer Liste abgespeichert.

[17] „AVM TR-064 - First Steps“.

```
<?xml version="1.0"?>
 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<s:Body>
<u:GetDeviceLogResponse xmlns:u="urn:dslforum-org:service:DeviceInfo:1">
<NewDeviceLog>
[…]
</NewDeviceLog>
</u:GetDeviceLogResponse>
</s:Body>
</s:Envelope>
```

Abbildung 9: Response

4.2 Verarbeitung und Weiterleitung

Die Abfrage der Logs von der FRITZ!Box liefert alle Ereignisse zurück, welche seit der letzten Trennung vom Strom gespeichert wurden. Um zu verhindern, dass bereits gemeldete Ereignisse doppelt gesendet werden, müssen diese vorab verglichen und nur neue Ereignisse dürfen nach Syslog weitergegeben werden.
Dazu wird bei der ersten Ausführung des Skriptes eine neue Logdatei angelegt. In dieser werden alle Ereignisse, welche an Syslog gemeldet werden, auch lokal gespeichert. Nach jeder Anfrage an die FRITZ!Box werden die neu erhaltenen Ereignisse, mit denen aus der lokalen Logdatei verglichen. Nur Solche die in den neuen Logdaten, aber nicht in den lokalen enthalten sind, werden an den Syslog-server weitergeleitet. Zu diesem Zweck wird mit Hilfe von „List Comprehension“, einer speziellen kurz-Syntax von Python, eine Liste aus neuen Ereignissen erstellt.

```
lokal_logs_set = set(lokal_logs)
new_events = [event for event in tr64logs if event not in lokal_logs_set]
```

Abbildung 10: Listen Subtraktion durch "List Comprehension"

Diese Liste der neuen Ereignisse wird an eine weitere Funktion übergeben, welche zeilenweise die Ereignisse in die lokale Logdatei schreibt und sie an den Syslog-Server weitersendet.
Zum Versenden der Ereignisse an Syslog wird die „syslog“ Bibliothek von Python verwendet. Mit dieser kann ein beliebiger String an die interne Verarbeitung von Syslog, zum Beispiel per „rsyslog“, weitergegeben werden. Zusätzlich kann der Schweregrad festgelegt werden. Die von der FRITZ!Box erhaltenen logs werden mit dem Schweregrad „Informational“ an den Syslog-Server weitergeben. Es ist nicht ohne weiteres möglich den Schweregrad dynamisch und abhängig von dem Event zu ändern. Da die FRITZ!Box keine Informationen über die Bedeutsamkeit der Ereignisse per TR-064 mitliefert, wäre es nur möglich alle Nachrichtentypen zu parsen und Ihnen einen Schweregrad zuzuordnen.

```
syslog.syslog(syslog.LOG_INFO, event)
```

Abbildung 11: Event an Syslog weiterleiten

4.3 Fehlerresistenz und kontinuierliche Ausführung

Die dritte Anforderung an ein solches Projekt ist eine gewisse Fehlerresistenz und eine kontinuierliche Ausführung. Besondere relevant ist die Funktion eines automatisierten Neustart des Programms nach einem kritischen Fehler.
Obwohl eine gewisse Fehlerresistenz auch innerhalb eines Python Skriptes mit Hilfe des Error Handlings erreichbar ist, gibt es keine Möglichkeit, dass sich ein solches Skript komplett selbständig neustarten kann, sollte zum Beispiel das Gerät vom Strom getrennt werden. Um diese Funktion zu erhalten, gibt es zwei probate Varianten: das regelmäßige Ausführen des Skriptes durch „cron“ oder die Ausführung als „Sytemd-Service“. Hier soll die Implementierung anhand eines „Sytemd-Service“ vorgestellt werden.
Mit Hilfe von „Systemd“ kann das Python Skript kontinuierlich überwacht werden. Außerdem kann im Falle eines unerwarteten Fehlers das Programm automatisiert neugestartet und bei Start des Systems ausgeführt werden.
Zur Registrierung des Skriptes als „Sytemd-Service“ muss eine “.service“ Datei in „/etc/systemd/system/“ abgelegt werden, welche die gewünschten Funktionen beschreibt. Besonders relevant ist, dass die „Restart“ Funktion auf „always“ und „TimeoutStartSec“ auf „infinity“ gesetzt werden. Diese Einstellungen

gewährleisten, dass „Sytemd“ immer wieder versucht, das Skript neu zu starten, sollte ein Fehler auftreten.

```
[Unit]
Description=Logging service for Tr-064 Device
After = network.target

[Service]
Type = simple
ExecStart =/usr/bin/python3 /home/tr064Logger/tr064CheckLog.py
User =  tr064Logger
WorkingDirectory =/home/tr064Logger
Restart = always
SyslogIdentifier = TR-064 Relais Service
RestartSec = 5
TimeoutStartSec = infinity

[Install]
WantedBy = multi-user.target
```

Abbildung 12: TR064Logging.service

Zusätzlich zur Funktion des automatisierten Neustartens ist es wichtig, dass alle Fehler, welche während der Ausführung auftreten, lokal gespeichert werden. Auch alle Fehler müssen an den Syslog-Server übermittelt werden, bevor die Ausführung des Programms beendet wird.
So kann gewährleistet werden, dass alle kritischen Fehler auch in Syslog nachvollziehbar sind.

5 Bewertung der erhaltenen Logdaten

Zusätzlich zur Betrachtung der Möglichkeiten zur Einbindung einer FRITZ!Box in eine Syslog-Umgebung, sollen hier auch die erhaltenen Logdaten anhand ihrer Relevanz für den Nutzer bewertet werden. Insbesondere da die per TR-064 erhaltenen Logdaten wesentlich von denen in der Weboberfläche dargestellten abweichen, soll hier die Differenz zwischen diesen betrachtet werden.

Zu diesem Zweck wurde die FRITZ!Box für mehrere Tage durchgehend betrieben, um so gewöhnliche Ereignisse zu sammeln. Die Logdaten wurden am Ende dieser Testphase aus der Weboberfläche kopiert und anschließend mit denen aus Syslog verglichen. Daraufhin wurden in der FRITZ!Box verschiedenste kritische Einstellungen geändert, um festzustellen, wie diese in den Logdaten dargestellt werden. Beispielsweise wurde ein neuer VPN-Account angelegt, welcher volle Konfigurationsrechte hatte und aus dem Internet auf die FRITZ!Box zugreifen durfte. Anschließenden wurden manuell per TR-064 die aktuellen Logdaten abgefragt. In Abbildung 13 sieht man wie um 11:42 der neue Nutzer angelegt wurde, sich ein neues WLAN-Gerät angemeldet und abgemeldet hat und wie um 12:18 per TR-064 die Logdaten abgefragt wurden. In Abbildung 14 sieht man hingegen die Logdaten, welche per TR-064 empfangen wurden. In Ihnen sind ausschließlich zwei Ereignisse bezüglich des WLANs zu finden. Die restlichen kritischen Ereignisse fehlen.

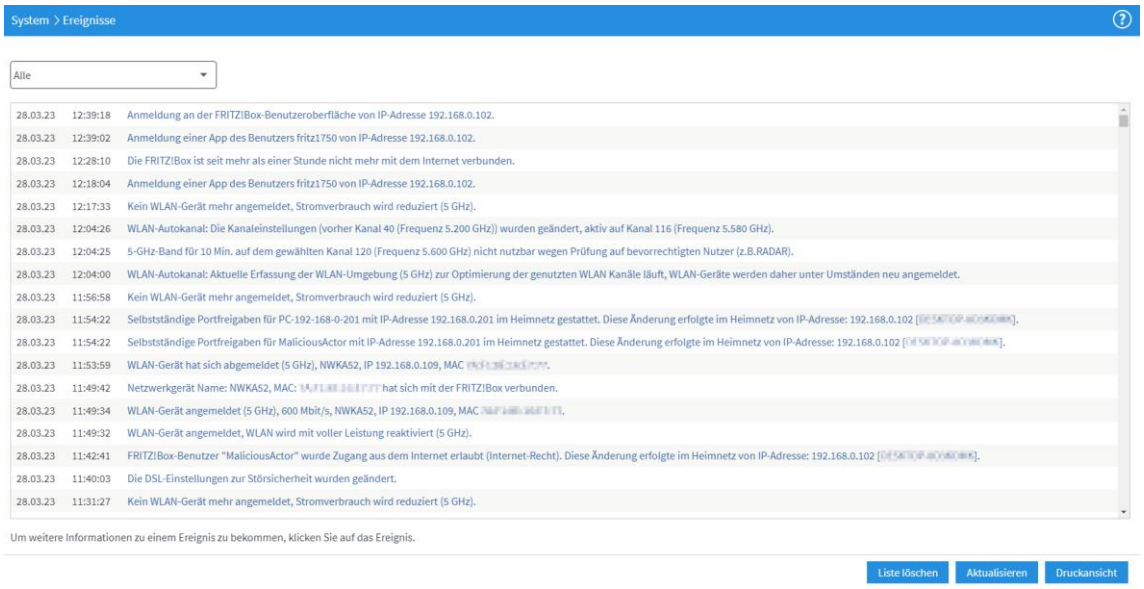

Abbildung 13: Weboberfläche Ereignisse FRITZ!Box

28.03.23 12:04:25 5-GHz-Band für 10 Min. auf dem gewählten Kanal 120 (Frequenz 5.600 GHz) nicht nutzbar wegen Prüfung auf bevorrechtigten Nutzer (z.B.RADAR).

28.03.23 12:04:00 WLAN-Autokanal: Aktuelle Erfassung der WLAN-Umgebung (5 GHz) zur Optimierung der genutzten WLAN Kanäle läuft, WLAN-Geräte werden daher unter Umständen neu angemeldet.

28.03.23 10:51:43 WLAN-Autokanal: Aktuelle Erfassung der WLAN-Umgebung (2,4 GHz) zur Optimierung der genutzten WLAN Kanäle läuft, WLAN-Geräte werden daher unter Umständen neu angemeldet.

28.03.23 10:51:43 WLAN-Autokanal: Aktuelle Erfassung der WLAN-Umgebung (5 GHz) zur Optimierung der genutzten WLAN Kanäle läuft, WLAN-Geräte werden daher unter Umständen neu angemeldet.

28.03.23 09:51:43 WLAN-Autokanal: Aktuelle Erfassung der WLAN-Umgebung (2,4 GHz) zur Optimierung der genutzten WLAN Kanäle läuft, WLAN-Geräte werden daher unter Umständen neu angemeldet.

28.03.23 09:51:43 WLAN-Autokanal: Aktuelle Erfassung der WLAN-Umgebung (5 GHz) zur Optimierung der genutzten WLAN Kanäle läuft, WLAN-Geräte werden daher unter Umständen neu angemeldet.

28.03.23 08:51:43 WLAN-Autokanal: Aktuelle Erfassung der WLAN-Umgebung (2,4 GHz) zur Optimierung der genutzten WLAN Kanäle läuft, WLAN-Geräte werden daher unter Umständen neu angemeldet.

28.03.23 08:51:43 WLAN-Autokanal: Aktuelle Erfassung der WLAN-Umgebung (5 GHz) zur Optimierung der genutzten WLAN Kanäle läuft, WLAN-Geräte werden daher unter Umständen neu angemeldet.

28.03.23 07:51:43 WLAN-Autokanal: Aktuelle Erfassung der WLAN-Umgebung (2,4 GHz) zur Optimierung der genutzten WLAN Kanäle läuft, WLAN-Geräte werden daher unter Umständen neu angemeldet.

28.03.23 07:51:43 WLAN-Autokanal: Aktuelle Erfassung der WLAN-Umgebung (5 GHz) zur Optimierung der genutzten WLAN Kanäle läuft, WLAN-Geräte werden daher unter Umständen neu angemeldet.

Abbildung 14: TR-064 Logdaten FRITZ!Box

Um diese Unterschiede zu bewerten, muss festgestellt werden, dass gewisse Ereignisse einen besonders hohen Wert gegenüber anderen haben. Events, die Handlungen dokumentieren, welche besonders geeignet zur Kompromittierung des Routers sind oder jene die über direkte Zugriffe auf die Konfiguration hinweisen, haben die höchste Relevanz. Hingegen haben Ereignisse, die über

automatisierte wiederkehrende Prozesse der FRITZ!Box berichten oder solche, die über geringfüge technische Änderungen berichten, eine niedrige Relevanz für die Sicherheit.
Die Information über einen geänderten WLAN-Kanal oder die regelmäßige Neuanmeldung im Betreibernetz kann wichtige ergänzende Informationen liefern. Die Meldung über einen neu angelegten FRITZ!Box Benutzer mit vollen Konfigurationsrechten und freiem Zugriff aus dem Internet hat aber eine sehr viel höhere Relevanz für die Sicherheit und sollte priorisiert behandelt werden.

Die FRITZ!Box sammelt eine Vielzahl an Ereignissen, welche in die Bereiche: System, Internetverbindung, Telefonie, WLAN und USB-Geräte unterteilt werden. Besonders relevant für die Sicherheit sind die Ereignisse aus dem Bereich System. In diesem werden die wichtigsten Ereignisse für die Sicherheit dokumentiert, wie Zugriffe auf die Konfiguration (per Weboberfläche oder per TR-064), Änderungen der FRITZ!Box-Einstellungen, erstmalig verbundene Netzwerkgeräte oder neu angelegte Benutzer mit Zugang aus dem Internet.
Diese Ereignisse haben die höchste Priorität, da sie, sofern nicht vorher bekannt und dokumentiert, einen eindeutigen Hinweis auf einen unautorisierten Zugriff oder sogar Änderung von kritischer Infrastruktur bieten.

Bei der Abfrage der Ereignisse per TR-064 ist auffällig, dass ein Großteil der in der Weboberfläche dargestellten Ereignisse nicht erhalten werden kann. In Abbildung 15 sind die aktuellen Ereignisse zusehen, nachdem die FRITZ!Box auf Werkseinstellungen zurückgesetzt und ein neuer Nutzer mit Internet-Recht erstellt wurde. In Abbildung 16 ist hingegen, die Ausgabe des TR-064-Protokolls zur gleichen Zeit zu sehen. Vergleicht man die aus TR-064 erhaltenen Ereignisse mit jenen aus der Weboberfläche, ist es eindeutig, dass alle relevanten Ereignisse fehlen.
Keines der für die Sicherheit besonders relevanten Ereignisse wird in TR-064 dargestellt. Aufgrund dessen sind die Logdaten, welche durch TR-064 erhalten werden können, ungeeignet die Grundlage für eine effektive Überwachung der FRITZ!Box zu bilden.

Abbildung 15: FRITZ!Box Weboberfläche

<?xml version="1.0"?>

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

<s:Body>

<u:GetDeviceLogResponse xmlns:u="urn:dslforum-org:service:DeviceInfo:1">

<NewDeviceLog>28.03.23 12:49:19 5-GHz-Band für 10 Min. auf dem gewählten Kanal 120 (Frequenz 5.600 GHz) nicht nutzbar wegen Prüfung auf bevorrechtigten Nutzer (z.B.RADAR).</NewDeviceLog>

</u:GetDeviceLogResponse>

</s:Body>

</s:Envelope>

Abbildung 16: TR-064 Response

6 Diskussion

Ein wesentliches Problem der Implementierung ist, dass keine Möglichkeit besteht die Schwere der Ereignisse dynamisch an Syslog weiterzugeben.
Dies stellt ein Problem dar, da es das Identifizieren von besonders kritischen Ereignissen wesentlich erschwert. So könnte ein Ereignis, welches auf die unauthentifizierte Verbindung mit dem VPN der FRITZ!Box hinweist, zwischen der Menge von Hinweisen über die Kalibrierung des WLAN-Kanals untergehen.
Die wesentliche Ursache für diese Problem ist, dass die FRITZ!Box keine Informationen über den Schweregrad des Ereignisses mitliefert, und auch keine Einordnung der Art des Ereignisses anbietet. Das Problem lässt sich lediglich über das direkte parsen der Lognachrichten mittigeren, welches allerdings durch eine fehlende Übersicht aller möglichen Ereignisse erschwert wird und so keine garantierte Lösung bieten kann.

Zwei weitere Probleme der Implementierung stellen die Performance und die Sicherheit der Anmeldedaten dar. Beide Probleme wurden aufgrund der geringen Relevanz für die Fragestellung nicht weiter untersucht, müssten aber bevor dieses Programm tatsächlich verwendet werden könnte, optimiert werden.
Die Performance des Programms ist nicht optimiert und könnte bei der Inbetriebnahme auf leistungsschwachen Internet-of-Things Geräten ein Problem darstellen. Die Belastung für den Prozessor kann minimiert werden, in dem eine Pause zwischen den Ausführungen implementiert wird. Hier ist zu beachten, dass eine solche Pause auch die Reaktionszeit auf kritische Ereignisse reduziert. Es muss zwischen Performance und Reaktionszeit abgewogen werden.
Zusätzlich kann durch das langfristige Speichern der Logs, der Speicher des Gerätes gefüllt werden und die Ausführung durch übermäßig lange Ladezeiten von Dateien verlangsamt werden. Es empfiehlt sich eine Löschroutine für die angelegte Logdatei zu implementieren.

Das letzte Problem, welches bei der Untersuchung aufgefallen ist, sind die fehlenden Ereignisse in TR-064 im Vergleich zu der Weboberfläche. Es ist zwar möglich eine Aussage über die Integrationsfähigkeit der FRITZ!Box zu treffen, jedoch hat die Aussage darüber nur geringe Relevanz für den Nutzen, wenn der mögliche Mehrwert mitbetrachtet wird.
Zugleich ist es schwierig eine Hypothese über die Ursache der fehlenden Ereignisse aufzustellen. Denn es war nicht möglich eine Dokumentation von AVM bezüglich der internen Sammlung der Ereignisse zu finden. Weder darüber, wie die Ereignisse intern gespeichert, noch wie diese klassifiziert werden, lassen sich Informationen finden. Auch eine Sammlung aller möglichen Ereignisse lässt sich nicht finden, damit hätte erprobt werden können, welche Ereignisse in TR-064 gesendet werden und welche nicht.
Allerdings muss an dieser Stelle erwähnt werden, dass im Rahmen dieser Arbeit nur mit einem Testgerät, einer FRITZ!Box 7590 AX und dem aktuellen Betriebssystem FRITZ!OS 7.31 getestet wurde. Möglicherweise treten bei anderen Modellen und oder anderen FRITZ!OS Versionen andere Verhaltensmuster auf.

7 Fazit und Ausblick

Mit Hilfe von einfachen auf SOAP basierenden POST-Requests ist es möglich, ein Skript zu entwickeln, welches von der FRITZ!Box per TR-064 Log-Daten anfordert, jene auf einen Relais-Client zwischenspeichert und anschließend mittels typischen Syslog Implementationen wie „RSyslogd“ an einen Protokollierungsserver weitergibt. Obwohl die Kommunikation per TR-064 mit der FRITZ!Box einen geringen Mehraufwand gegenüber einer reinen Syslog Implementation darstellt, ist das Einbinden der FRITZ!Box in eine Protokollierungsumgebung effizient realisierbar.

Obwohl die Komplexität der Einbindung gering ist, erscheint es dennoch kaum lohnend eine solche Integration durchzuführen, da die Logdaten, welche durch TR-064 übermittelt werden, keine relevanten Informationen enthalten. Da die durch AVM per TR-064 bereitgestellten Logdaten nicht einmal die grundlegendsten Ereignisse, wie das Einloggen an der Weboberfläche oder das Ändern von Einstellungen enthalten, erscheint der Wert einer Überwachung per TR-064 leider gering.

Die Integration von Endnutzer Netzwerkgeräten in bestehende Protokollierungsumgebungen könnte für viele Firmen und solche, die nach der Coronapandemie einen erhöhten Teil an Angestellten im Homeoffice haben, einen hohen Mehrwert bieten. Eine solche könnte es vereinfachen Angreifer frühzeitig ausfindig zu machen, welche zum Beispiel die schlechter geschützten Heimgeräte der Mitarbeiter ausnutzen wollen, um eine Verbindung in Firmennetzwerke aufzubauen.
Leider ist es bei der aktuellen Implementierung von TR-064 in AVM-Geräten nicht möglich eine effektive Überwachung dieser Systeme aufzubauen. Sofern durch AVM allerdings alle Logs per TR-064 erreichbar gemacht werden würden, könnte eine effektive zusätzliche Schutzschicht mit Hilfe eines solchen Systems eingerichtet werden.

Literaturverzeichnis

1. „What is SIEM?“,
zugegriffen 8. Februar 2023,
https://www.ibm.com/topics/siem.

2. „RFC 5424“,
zugegriffen 8. Februar 2023,
https://datatracker.ietf.org/doc/rfc5424/.

3. „Anteil der im Homeoffice arbeitenden Beschäftigten in Deutschland vor und während der Corona-Pandemie 2020 und 2021“,
zugegriffen 8. Februar 2023,
https://de.statista.com/statistik/daten/studie/1204173/umfrage/befragung-zur-homeoffice-nutzung-in-der-corona-pandemie/.

4. „AVM“,
zugegriffen 8. Februar 2023,
https://aetka.de/markenwissen/avm.

5. „RFC 3164“ (Cisco Systems, August 2001),
zugegriffen 22. Februar 2023
https://www.rfc-editor.org/rfc/rfc3164.

6. broadband-forum, „LAN-Side DSL CPE Configuration Corrigendum“,
zugegriffen 9. März 2023,
https://www.broadband-forum.org/technical/download/TR-064_Corrigendum-1.pdf.

7. broadband-forum, „About BBF“,
zugegriffen 9. März 2023,
https://www.broadband-forum.org/about-bbf.

8. „AVM TR-064 - First Steps“ (AVM GmbH, 18. November 2022), zugegriffen 9. März 2023, https://avm.de/fileadmin/user_upload/Global/Service/Schnittstellen/AVM_TR-064_first_steps.pdf.

9. BSI, „Mindeststandard des BSI zur Protokollierung und Detektion von Cyber-Angriffen“, zugegriffen 9. März 2023, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Mindeststandards/Mindeststandard_BSI_Protokollierung_und_Detektion_Version_1_0a.pdf?__blob=publicationFile&v=5.

10. Michael Alexander, *Netzwerke und Netzwerksicherheit: das Lehrbuch*, 1. Aufl (Heidelberg: Hüthig, 2006).

11. Kathryn Knerler, Ingrid Parker, und Carson Zimmerman, *Cybersecurity Operations Center*, 2te Auflage (The MITRE Corporation, o. J.).

Anhang

Anhang 1.1: tr064CheckLog.py

```
import requests
from requests.auth import HTTPDigestAuth
import syslog


credentials_filename = "credentials.ini"
logs_filename = "tr064.log"

# Used for more granular Error handling acording to HTTP Response Code
class CustomError(Exception):
    def __init__(self, value):
        self.value = value

    def __str__(self):
        return "Error: %s" % self.value


def get_log_from_tr64(host, port, uid, passwd):
    payload = """<?xml version="1.0"?>
    <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <s:Body><u:GetDeviceLog
xmlns:u="urn:dslforumorg:service:DeviceInfo:1"></u:GetDeviceLog>
    </s:Body>
    </s:Envelope>"""

    headers = {
        'Content-Type': 'text/xml; charset=utf-8',
        'SOAPACTION': 'urn:dslforum-org:service:DeviceInfo:1#GetDeviceLog'
    }

    locator = "/upnp/control/deviceinfo"

    url = "http://" + host + ":" + port + locator
    try:
        response = requests.request("POST", url, headers=headers,
data=payload, auth=HTTPDigestAuth(uid, passwd))

        try:
            if response == "<Response [200]>":
                pass
            elif response == "<Response [401]>":
                raise CustomError("TR-064 Relais failed to authorize with
TR-064 Log Origin Device")
        except CustomError:
            syslog.syslog(syslog.LOG_ERR, str(CustomError))
            raise CustomError

    except Exception:
        syslog.syslog(syslog.LOG_ERR, str(Exception))
        raise Exception
```

```
    try:
        temp, garbage = response.text.split("</NewDeviceLog>", 1)

        garbage, temp = temp.split("<NewDeviceLog>", 1)

        events = temp.split("\n")

        return events



    except Exception:
        syslog.syslog(syslog.LOG_ERR, str(Exception))
        raise Exception


def get_credentials_from_file(file):
    # get credentials needed for execution from a file
    try:
        with open(file, 'rt') as file:
            lines = [str(line.strip()) for line in file]

        return lines

    except Exception:
        syslog.syslog(syslog.LOG_ERR, "Error in reading files for TR064
logging service " + str(Exception))
        raise Exception


def get_logs_from_file(file):
    # get logs from file if file not existend return empty lines
    try:
        with open(file, 'rt') as file:
            lines = [str(line.strip()) for line in file]

    except Exception:
        syslog.syslog(syslog.LOG_INFO, "New Log File will be created for
TR064 logging service")
        lines = ""

    return lines
```

```
def compare_logs(file_logs, tr64logs):
    # compares new logs from tr064 with logs saved in file
    # if new log entries exist return them, if not return none

    try:

        if file_logs:

            s = set(file_logs)
            new_messages = [x for x in tr64logs if x not in s]

        else:
            new_messages = tr64logs

        return new_messages

    except Exception:
        syslog.syslog(syslog.LOG_ERR, "Error in comparing files for TR064
logging service " + str(Exception))
        raise Exception


def log_to_syslog_and_file(new_logs, filename):
    # log all new log entries to syslog and add them afterwards to the file
of logged messsages




    try:
        logfile = open(filename, "at")

        for line in new_logs:
            syslog.syslog(syslog.LOG_INFO, line)
            logfile.write(line + "\n")

        logfile.close()
        return 1

    except Exception:
        syslog.syslog(syslog.LOG_ERR, "Error in logging new messages for
TR064 logging service " + str(Exception))
        raise Exception


# HOST = lines[0], PORT = lines[1], UID = lines[2], PASSWD = lines[3]
creds = get_credentials_from_file(credentials_filename)

while True:
    # to reduce impact on system consider a sleep phase between executions
    # time.sleep(5)

    logs_from_file = get_logs_from_file(logs_filename)

    logs_from_device = get_log_from_tr64(creds[0], creds[1], creds[2],
creds[3])

    new_log_messages = compare_logs(logs_from_file, logs_from_device)

    log_to_syslog_and_file(new_log_messages, logs_filename)
```

Anhang 1.2: TR064Logging.service

```
[Unit]
Description=Logging service for Tr-064 Device
After = network.target

[Service]
Type = simple
ExecStart =/usr/bin/python3 /home/tr064Logger/tr064CheckLog.py
User =  tr064Logger
WorkingDirectory =/home/tr064Logger
Restart = always
SyslogIdentifier = TR-064 Relais Service
RestartSec = 5
TimeoutStartSec = infinity

[Install]
WantedBy = multi-user.target
```