http://dx.doi.org/10.25673/111642

Edgeshark: Einblick in die virtuelle Kommunikationswelt (nicht nur) von Containern

Ein (OpenSource) Beitrag zu einer konvergierenden IT/OT

Dr. Harald Albrecht[1]

Abstract: Nicht zuletzt mit aktuellen „vPLC"-Offerten gewinnt der Einsatz von IT-Technologien wie Docker und Kubernetes zum Virtualisieren und Orchestrieren der Industrieautomation an Fahrt. Die IT verfügt über bewährte Blaupausen zum Strukturieren von Applikationen und deren virtueller Kommunikation auch in größeren Container-Systemen. In der Praxis stoßen Anwender, Entwickler, Experten und Forschende aus Automatisierung und Industrie-Kommunikation immer wieder auf überraschendes oder unerwünschtes Systemverhalten. Umfangreiche Online-Dokumentation und heutige Kommandozeilen-Werkzeuge sind dabei sowohl Hilfe als auch Hürde, um die virtuellen Systeme und Netzwerke zu durchdringen und diagnostizieren: warum kommen keine Telegramme in der Anlage an, warum gehen keine Daten die die Cloud, …? In der eigenen mehrjährigen Praxis als „containerisierter Systemarchitekt" u.a. bei der Siemens „Industrial Edge" ist deshalb das anfangs interne Projekt „Edgeshark" entstanden und stetig weiterentwickelt worden. Siemens steuert nun das Projekt (MIT-Lizenz) zur Github OpenSource Community bei. In diesem Beitrag steht jedoch nicht das Werkzeug im alleinigen Mittelpunkt, sondern es soll vielmehr wie ein guter Ausbilder Interessierten einen einfachen Zugang zu verschiedenen Elementen der virtuellen (und auch realen) Kommunikation in Containersystemen ermöglichen. Der Zugang kann explorativ über eine Weboberfläche erfolgen, genauso kann Edgeshark auch per REST API einfach in neuen Aufgaben eingebunden werden. Weiterhin existiert eine Integration samt Live-Übertragung zum OpenSource-Werkzeug Wireshark.

Keywords: Software-Container, virtuelle Kommunikation, Diagnose, Wissenstransfer

1 „Wirrtuelle" Kommunikation?

Was kommt mit den IT-Technologien zur Virtualisierung und Orchestrierung in Form von Docker und Kubernetes auf Automation und Industrie-Kommunikation zu? Müssen wir Automatisierenden und „Industrie-Kommunizierenden" uns überhaupt zumindest in Teilen mit diesen Technologien auseinandersetzen oder können wir sie nicht „einfach nur benutzen"? Und lassen sich bisherige Erfahrungen mit virtuellen Netzwerken überhaupt in die virtuelle Welt der Container so einfach übertragen?

In der Praxis stoßen Beteiligte – egal, ob industrieller Anwender, Systemadmins, Entwickler, Netzwerker, Automatisierende, … – bald auf überraschendes oder auch unerwünschtes Systemverhalten. Zwar stehen den Betroffenen vielfältige Quellen offen,

[1] Siemens AG, DI CTO, Gleiwitzer Straße 555, 90475 Nürnberg, harald.albrecht@siemens.com

beispielsweise die Docker-eigene Dokumentation [DoDo] und die bekannte Frage-und-Antwort-Plattform „Stack Overflow“ [SO]. Daneben existiert auch ein schier unüberschaubarer Markt an kommerziellen Schulungen. Diese vielfältigen Angebote können jedoch nur eingeschränkt die Fragen beantworten: wie sieht die virtuelle Kommunikation *konkret in diesem Moment* in meinem (vermutlich) liebevoll und mühsam konfigurierten Automatisierungs-System wirklich aus? “Macht” das System das, was ich ihm als Konfiguration vorgegeben habe? Erschwerend kommt hinzu, dass virtuelle Kommunikation in Container-Systemen heute weitestgehend anderen Strukturen und Richtlinien folgt, als es bislang bei virtuellen Netzwerken wie beispielsweise der IEEE 802.1 VLANs Praxis ist.

Die zum Beantworten dieser Fragen heute verfügbaren Programmier-Schnittstellen und Werkzeuge sind zwar *prinzipiell* ausreichend. Sie erfordern jedoch ein sehr hohes Maß an Verständnis der unterlagerten mehreren Ebenen der Virtualisierungstechnologien, wie beispielsweise die sog. „namespaces“ des Linux-Kernels [Namsp7] sowie deren spezielle Nutzung für Container.

Gängige und beliebte Werkzeuge zur Netzwerkdiagnose – insbesondere Wireshark [WS] – funktionieren zwar grundsätzlich auch mit Containern, sind jedoch ihrer nicht gewahr. Letztlich sollen sich Anwender auch nicht mit OS-eigenen Referenzen wie „/proc/12345/ns/net“ befassen müssen, die nur über komplizierte Kommandozeilen-Ketten in speziellen Terminalsitzungen direkt in Spezial-Container hinein ausgeführt werden können („war das Argument nun „-t“ samt Prozessnummer oder doch „--net“ und eine procfs-Referenz?“). Außerdem kann und will nicht jeder industrielle Anwender es erlauben, zur Diagnose seiner Container-unterstützten Automatisierungsanwendung sich tief in sein laufendes System auf eine Kommandozeile zu begeben.

2 OpenSource-Projekt Edgeshark

Seit Jahren mangelt es unserer Beobachtung nach an Werkzeugen, um die (virtuelle) Kommunikation in Container-Systemen auch für Einsteiger einfach begreif- und diagnostizierbar zu gestalten. Weder der kommerzielle Markt noch die OpenSource Gemeinschaft bieten derartige Werkzeuge an. Das kommerzielle Angebot „Cloudshark“ [CS] adressiert im Gegensatz zu dem hier verfolgten Ziel speziell das *Cloud-gestützte Auswerten von Paketaufzeichnungen*, ohne dass vor Ort noch ein Wireshark-Programm installiert werden muss. Wie allerdings die Paketaufzeichnung erfolgte, bleibt in diesem speziellen Fall außer Acht.

Anwender, Entwickler und Forschende müssen deshalb üblicherweise und bei entsprechendem Leidensdruck mit verschiedenen Hilfsmitteln sowie länglichen und fehlerträchtigen Kommandozeilen versuchen, mehr schlecht als Recht ihre Probleme bei der Container-Kommunikation zu diagnostizieren. In Folge bauten in der Vergangenheit Entwickler dafür vorübergehend das Wireshark-Programm mit in ihre Anwendungs-Container ein.

Nicht zuletzt für System- und Softwarearchitekten war es zudem eine ständig wiederkehrende Herausforderung, Anwendern ein glaubhaftes und nachvollziehbares Bild der zumeist vielgestaltigen Kommunikation in Docker-Systemen zu vermitteln.

In diesem mehr als unbefriedigenden Zustand entstand in der Siemens AG zunächst intern das Projekt „Edgeshark“. Dieses trägt Siemens nun als OpenSource-Projekt mit MIT-Lizenz auf Github [ES] zur OpenSource Community bei – nicht zuletzt, um damit Automatisierungsanwendern aus Industrie und Forschung beim sanften Einstieg in die „containerisierte Automatisierung“ zu unterstützen.

Der Projektname „Edgeshark“ ist der Historie verdankt, dass die meisten bisherigen Anwender das Werkzeug zuerst im Rahmen der Siemens Industrial Edge-Plattform [SIE] kennengelernt und (erfolgreich) einsetzen konnten. Edgeshark ist jedoch nicht auf die Industrial Edge-Plattform beschränkt, sondern kann auf jedem (Docker) Container-Host eingesetzt werden. Weiterhin erkennt Edgeshark die Besonderheiten lokaler Kubernetes-in-Docker-Installationen („KinD“, [KinD]), wie Pods und KinD-Knoten-Container, ohne dass dazu ein Zugriff auf das Kubernetes-API nötig ist. Aktuell umfasst das Edgeshark-Projekt rund 23.500 Zeilen Quellcode und 13.000 Zeilen Dokumentation, sowohl als Endanwender- als auch Programmdokumentation.

3 Architektur

Den Kern von Edgeshark bilden die beiden (containerisierten) Dienste einerseits für das Auffinden der virtuellen Netzwerkstruktur und deren Zustands („ghostwire“ genannt) und andererseits das Erfassen und Streamen von Netzwerkpaketen („packetflix“), siehe die folgende Abb. 1. Die Ausgestaltung als einfach konsumierbare Dienste erlaubt eine möglichst vielfältige Nutzung (und zugleich Wiederverwendung) in unterschiedlichen Anwendungen. Zudem werden Einschränkung auf ein bestimmtes Programmiersprachen-Ökosystem in der Nutzung vermieden. Nicht zuletzt folgt das Projekt hier den in der IT und den Cloud-Technologien bewährten Mustern.

Auf die Edgeshark-Dienste kann per REST- sowie Websocket-APIs auf vielfältige Weise zugegriffen werden:

- interaktiv von einem gängigen Webbrowser,
- interaktiv oder auch programmiert vom OpenSource-Werkzeug Wireshark zur Analyse des Netzwerkverkehrs von/zu Containern.
- von eigenen (neuartigen) Applikationen, gegebenenfalls unter Zuhilfenahme der “csharg” Go-Bibliothek.

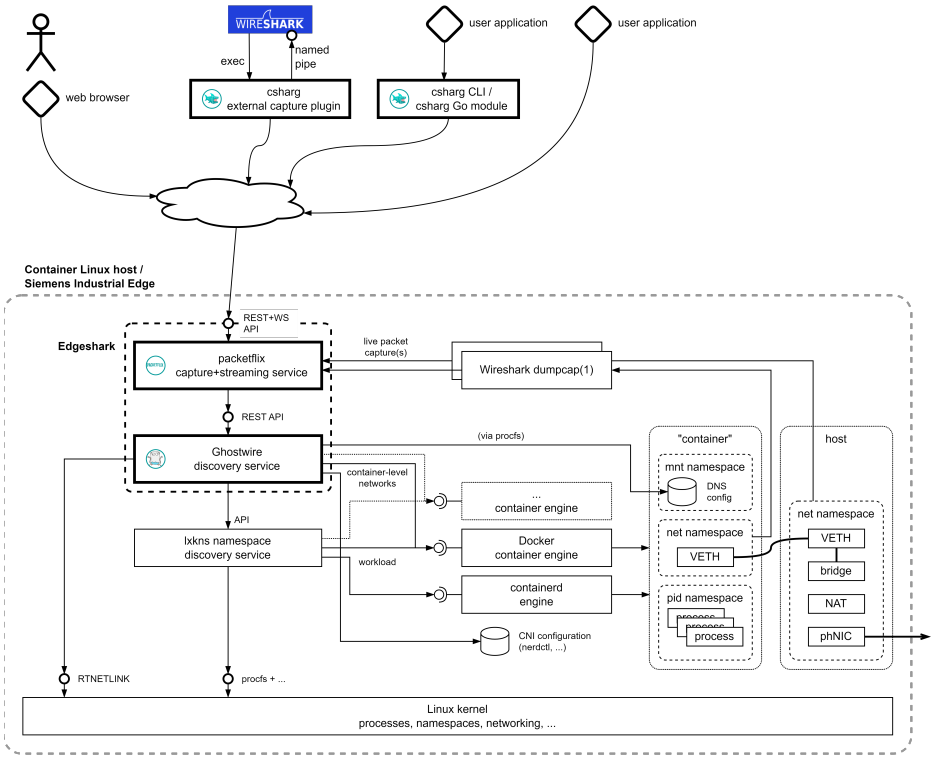

Abb. 1: Die Edgeshark-Architektur im Überblick.

Ein wichtiger Vorteil ist, dass keinerlei Änderungen an anderen Applikationen bzw. Containern vorgenommen werden müssen: die Edgeshark-Dienste „wandern“ sozusagen im Rahmen der ihnen erteilten Rechte durch das Linux-System mit seinen Containern, um die erforderlichen Informationen zu sammeln.

Die Edgeshark-Dienste kapseln hierbei die inhärente Komplexität beim Auffinden der über Container und möglicherweise verschiedene Container-Engines verteilten Netzwerk- und Konfigurationsinformationen. Anwender (und Anwendungen) können sich somit auf ihre eigentlichen Ziele wie Diagnose und ein besseres Verständnis der virtuellen Kommunikationsstrukturen konzentrieren.

Der Ghostwire-Dienst verfügt über mehrere Erweiterungspunkte („extension points“), um beispielsweise einfach APIs weiterer Container-Engines nachzurüsten oder neue Orchestrierungs-Markierungen zu nutzen. Eine experimentelle Unterstützung für podman [Podman] ist vorhanden.

Der Ghostwire-Dienst unterstützt nur Linux. Eine Unterstützung der Windows-Kommunikations-Architektur ist aufgrund der unklaren Informationslage sowie als in der Regel

untypisches Einsatzgebiet von Containern nicht vorgesehen. Ebenso wird auch WSL2 bislang nicht unterstützt. Pull Requests sind herzlich willkommen und sollten freundlicherweise zuvor mit dem Edgeshark-Projekt besprochen werden.

4 Ghostwire Discovery-Dienst

Der Ghostwire-Dienst – so benannt in Anspielung auf die virtuellen „Leitungen“ zwischen Containern – ist für das Auffinden von virtuellen und realen Netzwerk-Schnittstellen zuständig, sowie deren „Verschaltungstopologie“, siehe auch Abb. 2 mit dem Datenmodell.

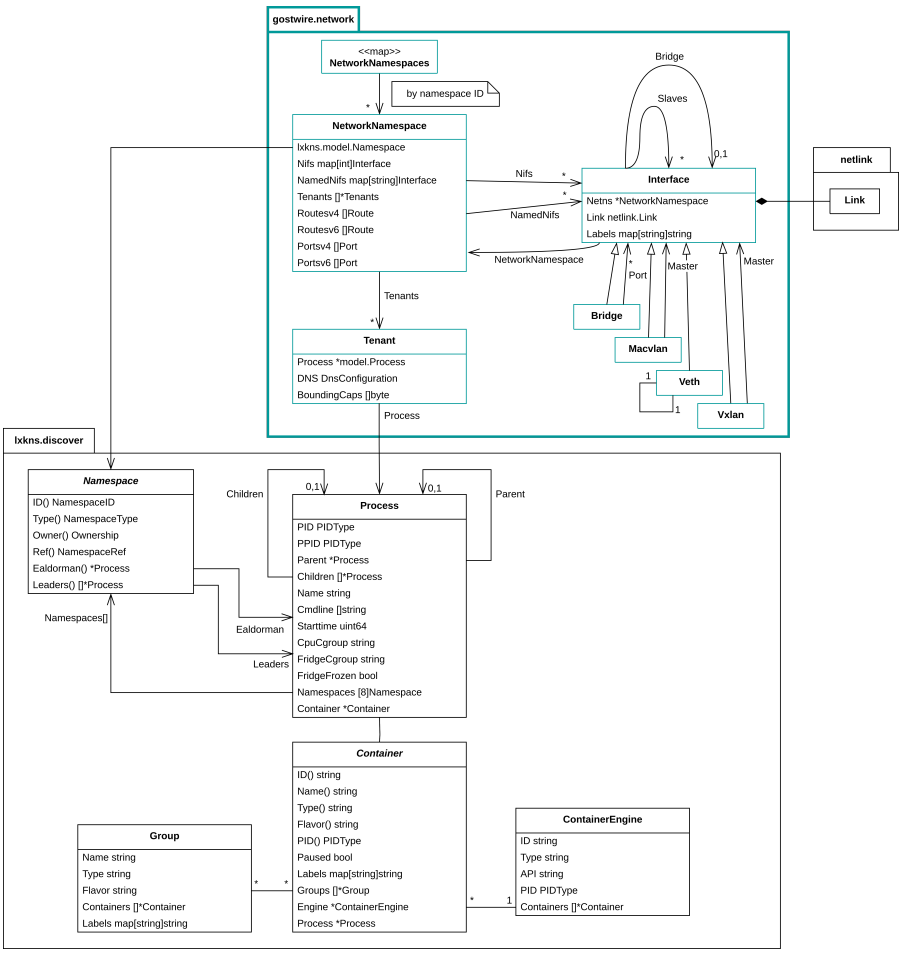

Abb. 2: Das Datenmodell.

Dabei greift der Dienst auf einen (integrierten) Erkundungsdienst für die sogenannten Linux-Kernel „namespaces“ [Namsp7] sowie deren Beziehungen zu Prozessen und Containern zurück. Auf der Basis dieser grundlegenden Informationen sammelt Ghostwire dann verschiedene Netzwerk- und Kommunikations-spezifische Daten und stellt diese als JSON-Datenmodell an seinem REST API bereit:

- reale und virtuelle Netzwerkschnittstellen und deren topologische Beziehungen (wie beispielsweise paarweise „VETH“-Netzwerkkarten).
- IP-Adress- und Routen-Konfiguration der verschiedenen (virtuellen) IP-Stacks sowie deren Zuordnung zu Containern, Pods, …
- die Konfiguration der DNS „stub resolver“ („DNS-Clients“ im umgänglichen Sprachgebrauch) innerhalb der Container.
- geöffnete TCP- und UDP-Port sowie die diese Ports bedienenden Prozesse (sowie entsprechend darauf aufsetzende „höhere“ Konstrukte wie Container, Pod, …)
- Weiterleitungen von TCP- und UDP-Ports an andere Ports.
- Container-Details zu den erteilten Berechtigungen („capabilities“).

5 Auf Erkundung

Die Edgeshark-Weboberfläche basiert rein auf der am REST API als JSON-Modell bereitgestellten Daten. Technisch werden die JSON-Daten hierbei erst im Browser selbst in einer React-Web-Anwendung [React] in ihre HTML-Darstellung überführt. Damit lassen sich einfach und bequem spezialisierte Ansichten nur auf Teile des Informationsmodells umsetzen.

So zeigt die Weboberfläche zu Beginn zunächst eine topologische Ansicht der “virtuellen Leitungen” zwischen den Containern, siehe Abb. 3. Hier sind auch die bei Containern für die Automatisierung vermehrt anzutreffenden verschiedenen Kommunikationsverbindungen erkennbar, wie in Abb. 3 rechts gesondert gezeigt. So besitzt der hier gezeigte Anwendungscontainer nicht nur die insbesondere bei IT-Anwendungen übliche Anbindung an eine virtuelle Bridge (virtueller Switch) im Container-Host (Schnittstelle „eth0“). Zusätzlich verfügt diese Anwendung noch über eine Schnittstelle „eth1“ (vom Typ „MACVLAN“), die die direkte Kommunikation auf Ebene von Ethernet-Telegrammen beispielsweise mit Feldgeräten ermöglicht. Nicht jede Automatisierungs-Anwendung benötigt diese doppelte Anbindung, sie wird in der Regel immer dann benötigt, wenn nicht-IP-basierte Feldgeräte-Protokolle direkt benutzt werden müssen.

Durch Antippen oder Anklicken von Containern und Netzwerk-Elementen können Anwender einfach in eine stärker fokussierende Kommunikationsansicht mit höherem Detailgrad wechseln. In der in Figur 4 beispielhaft gezeigten Ansicht können Anwender

unter anderem rasch erkennen, ob und welche Kommunikationsendpunkte in ihren Containern konkret in diesem Moment vorhanden sind.

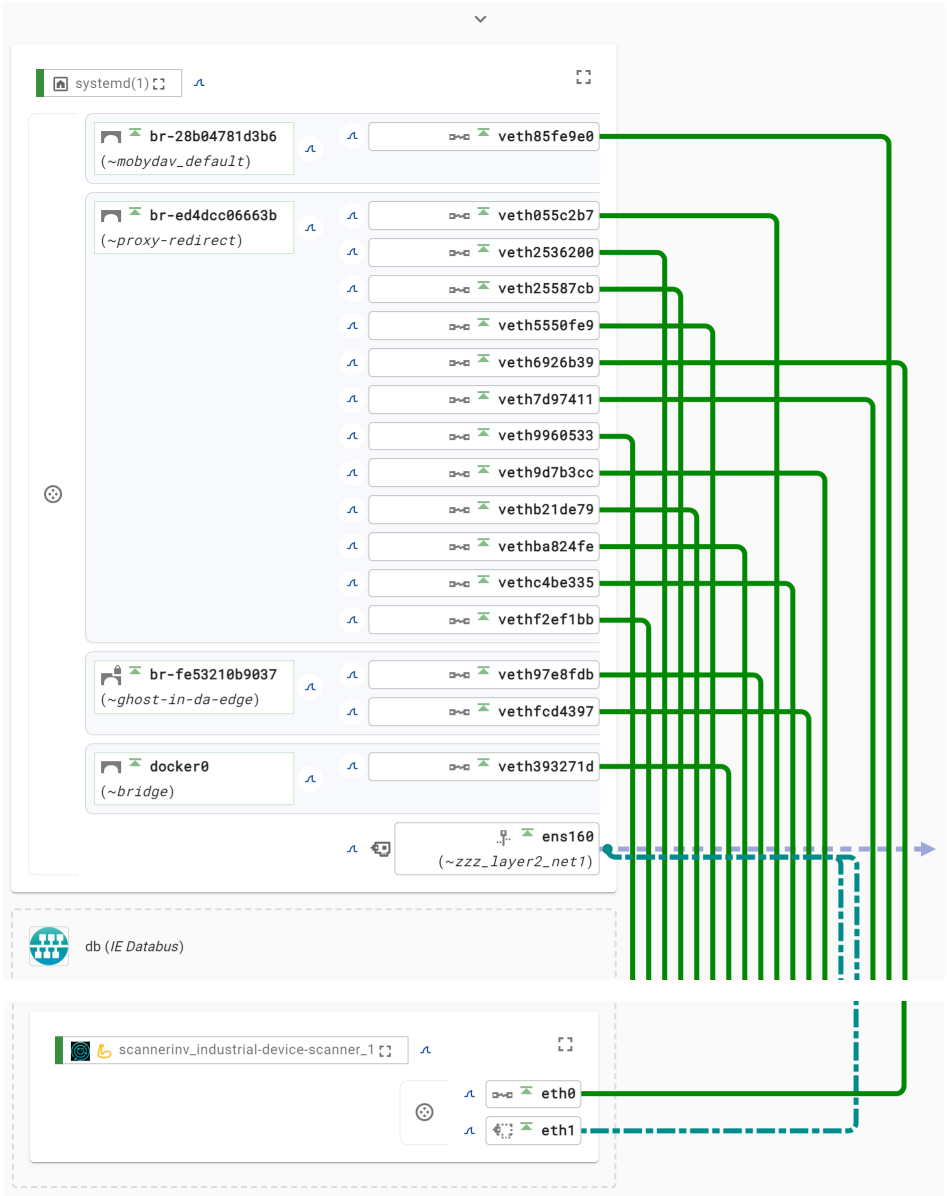

Abb. 3: Einfacher visueller Einstieg inklusive Navigationshilfen in die teilweise komplexen Kommunikationsbeziehungen in Container-Systemen.

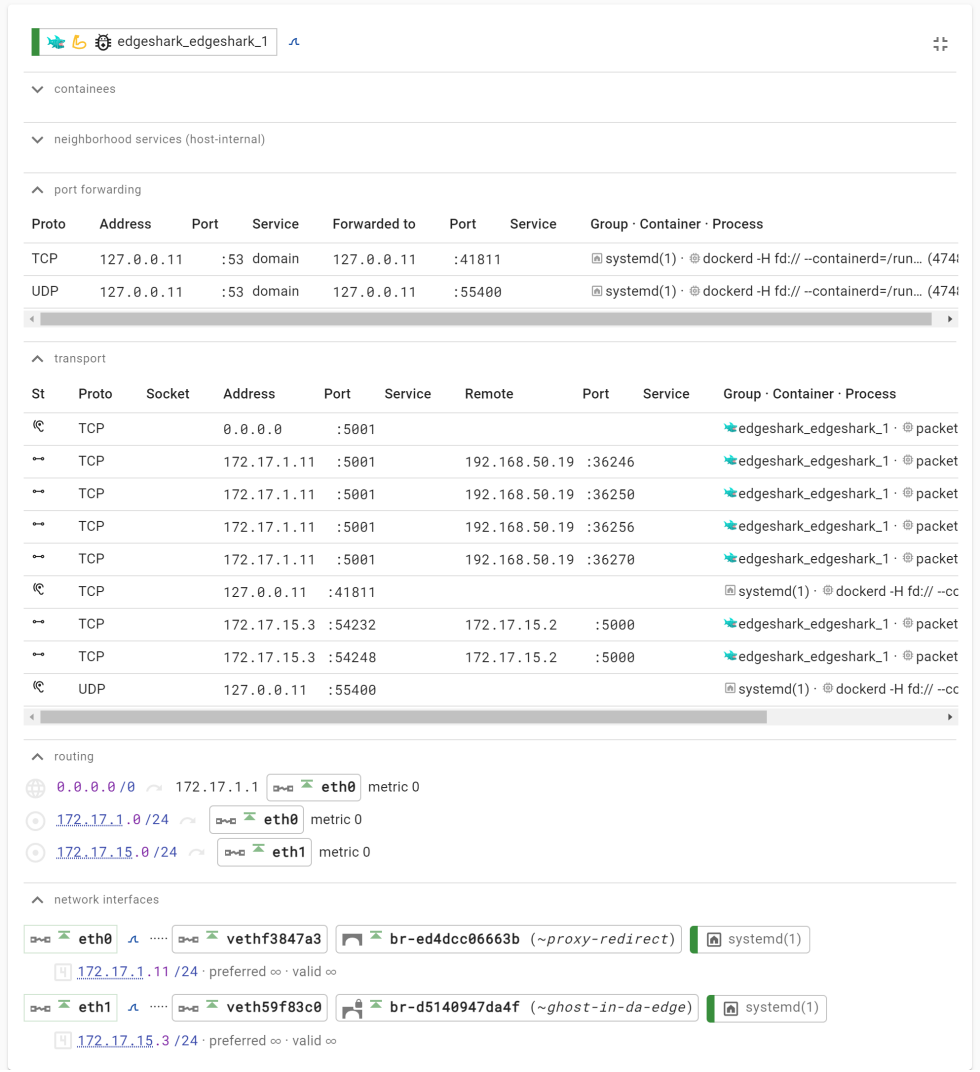

Abb. 4: Detaillierter Einstieg in die aktuell wirksame Kommunikationskonfiguration eines Containers.

Nachdem diese Informationen dem JSON-Datenhaushalt des Discovery-API-Endpunktes entstammen, können derartige Informationen auch automatisiert ausgewertet werden – beispielsweise zur Prüfung von Containern hinsichtlich ihrer exponierten Endpunkte, zum Überwachen bestimmter Kommunikationsflüsse sowie vieles mehr.

Im Gegensatz zu den bekannten CLI-Werkzeugen wie beispielsweise “netstat” [netstat8] und “show socket” [Shows8] enthält das Ghostwire-Informationsmodell nicht nur alle Endpunkte über alle virtuellen Netzwerkstacks eines Hosts hinweg, sondern auch deren Zuordnung zu Containern und ggf. Composer-Projekten, Kubernetes-Pods, und dergleichen.

6 Packetflix Capture Streaming-Dienst

Von etlichen Heimroutern – wie beispielsweise den „Fritz!box“en – ist die Funktion bekannt, auf Wunsch einen Mitschnitt des Netzwerkverkehrs aufzeichnen und dann später auf den eigenen Rechner herunterladen zu können. Nun ist eine derartige Download-Funktion nicht grundsätzlich falsch, erscheint jedoch in ihrer konkreten Ausführung etwas aus der Zeit gefallen.

Viele Menschen fahren heute nicht mehr umständlich in eine Videothek [VT], um sich dort audiovisuelle Medien wie Blu-rays oder gar VHS-Kassetten [VHSKa] auszuleihen, zuhause anzusehen und dann (eventuell) wieder zurückzubringen. Vielmehr ist es inzwischen gängig, audiovisuelle Medien bequem zuhause oder jederzeit unterwegs per „streaming“ zu konsumieren, wie Netflix, Spotify und YouTube zeigen (um nur eine kleine Auswahl an Streaming-Diensten beispielhaft zu nennen).

Der „Packetflix“-Dienst überträgt somit konsequenterweise das heutige Streaming-Modell auf das Mitschneiden und Analysieren von Netzwerkverkehr insbesondere bei Containern. Eine einfache Websocket-API nimmt dabei beispielsweise einfach nur den Namen des „anzuzapfenden“ Containers entgegen und beginnt daraufhin, die mit geschnittenen Netzwerkpakete an die Gegenstelle zu übertragen.

Intern kümmert sich der Packetflix-Dienst darum, den Container-Namen unter Zuhilfenahme des Ghostwire-Dienstes in die benötigten Ressourcen-Referenzen auf der Ebene der Linux-API abzubilden. Damit startet er danach das Wireshark-Werkzeug „dumpcap“ [WSdc], wobei er es zudem in den Kontext des Netzwerkstacks des betreffenden Containers einblendet. Wie in Abb. 1 angedeutet, wird im Dienst das vergleichsweise schlanke dumpcap verwendet, nicht jedoch ein vollumfänglicher Wireshark selbst. Tatsächlich nutzt Wireshark dumpcap selbst in verschiedenen Anwendungsszenarien als externe Proben-Entnahmestelle.

Als besondere Funktion schreibt Packetflix zusätzliche Informationen in den Paketdatenstrom hinein, die den untersuchten Container eindeutig identifizieren (Name, ID, …). Damit wird das heute übliche Problem vermieden, dass die gängigen Werkzeuge dumpcap, tcpdump [Tcpdmp1] und Wireshark keinerlei Wissen über Container besitzen und es somit ansonsten schwerfallen würde, nachträglich die Quelle der Paketdaten zu identifizieren: ohne Nachhilfe hinterlegt dumpcap wenig aussagekräftige Informationen wie Schnittstelle „eth0“ des Hosts „52c001de42bef1“.

Der Zugriff auf den Packetflix-Dienst erfolgt im einfachsten Fall aus der Weboberfläche heraus, kann aber auch über Kommandozeilenprogramme erfolgen (Wireshark einbezogen). In der Weboberfläche stehen hierfür Schaltflächen bereit, die eine stilisierte „Haifischflosse“ in Anspielung auf Wireshark zeigen, siehe auch Abb. 5.

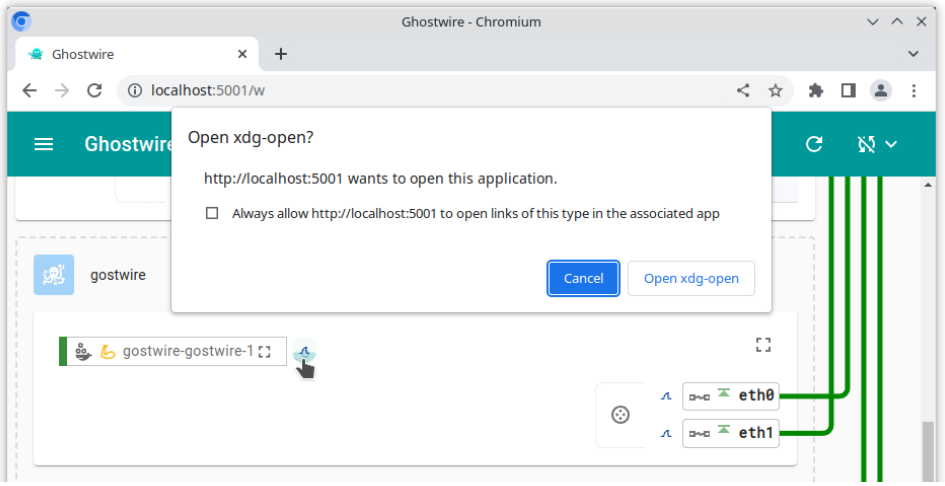

Abb. 5: Einfacher Wechsel zum Wireshark-Tool zwecks Live-Analyse von Paketen.

Antippen oder Anklicken startet nach Rückfrage beim Anwender den lokal auf dem eigenen Rechner installierten Wireshark. Dieser verbindet sich daraufhin aufgrund der ihm übermittelten „Kontaktdaten“ mit dem Packetflix-Dienst und zeigt danach ohne weiteres Zutun die live übertragenden Netzwerk-Pakete an. Der Packetflix-Dienst besitzt hierbei selbst keine Wireshark-Komponente.

Die Verbindung zwischen Wireshark und dem Packetflix-Dienst stellt ein sogenanntes „external capture plugin“ [Extcap] her; hierbei handelt es sich um ein eigenständiges kleines Binärprogramm, das von Wireshark automatisch erkannt und bei Bedarf herangezogen wird.

7 Zusammenfassung

Mit dem OpenSource-Projekt „Edgeshark“ [ES] steht ein einfach nutzbares und erweiterbares Werkzeug zur Verfügung, um die virtuelle Kommunikationsstrukturen von Containern darzustellen, zu diagnostizieren und auszuwerten. Edgeshark soll dabei aber lediglich ein Einstiegspunkt sein: weitere, neuartige Anwendungen können beispielsweise direkt auf den Datenhaushalt in JSON-Kodierung über ein REST API zugreifen. Die Codebasis steht unter der MIT-Lizenz und nutzt aktuell die weit verbreiteten Sprach- und Ablaufplattformen Go [Go] sowie React [React] in Verbindung mit Typescript.

Zukünftige Container-APIs können bei Bedarf über Erweiterungspunkte zugerüstet werden.

Literaturverzeichnis

[Caps7] Capabilities(7) man page, https://man7.org/linux/man-pages/man7/capabilities.7.html, Stand 02.06.2023.

[CS] Cloudshark, https://www.qacafe.com/analysis-tools/cloudshark/qa-cloudshark-personal-saas/, Stand 02.06.2023.

[DoDo] Docker Documentation, https://docs.docker.com/, Stand 02.06.2023.

[ES] Edgeshark Project auf Github, https://github.com/siemens/edgeshark.

[Extcap] extcap(4) manual page, https://www.wireshark.org/docs/man-pages/extcap.html, Stand 02.06.2023.

[Go] https://go.dev/, Stand 02.06.2023.

[KinD] Kubernetes-in-Docker, https://github.com/kubernetes-sigs/kind, Stand 02.06.2023.

[Namsp7] namespaces(7) man page, https://man7.org/linux/man-pages/man7/namespaces.7.html, Stand 02.06.2023.

[Netstat8] netstat(8) man page, https://man7.org/linux/man-pages/man8/netstat.8.html, Stand 02.06.2023.

[Podman] https://podman.io/, Stand 02.06.2023.

[React] https://react.dev/, Stand 02.06.2023.

[Shows8] show sockets manpage, https://man7.org/linux/man-pages/man8/ss.8.html, Stand 02.06.2023.

[SIE] Siemens Industrial Edge, https://www.siemens.com/de/de/produkte/automatisierung/themenfelder/industrial-edge.html, Stand 02.06.2023.

[SO] Stack Overflow, https://stackoverflow.com/, Stand 02.06.2023.

[Tcpdmp1] tcpdump(1) man page, https://man7.org/linux/man-pages/man1/tcpdump.1.html, Stand 02.06.2023.

[VHSKa] Video Home System Kassetten, https://de.wikipedia.org/wiki/Video_Home_System#Kassetten, Stand 02.06.2023.

[VT] Videothek (Wikipedia), https://de.wikipedia.org/wiki/Videothek, Stand 02.06.2023.

[WS] Wireshark Organisation, https://www.wireshark.org, Stand 02.06.2023.

[WSdc] Wireshark Organisation, https://www.wireshark.org/docs/man-pages/dumpcap.html, Stand 02.06.2023.