Mehrstufiges Firewall-Konzept

(VoIP-Sicherheit: IP-Telefonie mit Session-Border-Controllern schützen; *iX* 11/2018, S. 112)

Vielen Dank für diesen tollen Artikel in der Zeitschrift. Ich beschäftige mich aktuell intensiv mit der Thematik Session Boarder Controller und mache gerade Tests mit einem AudioCodes Mediant SBC. Beim Lesen des Artikels hat sich bei mir in Bezug auf die Integration des SBC in die IT-Infrastruktur eine Frage gestellt.

Der SBC hat eine integrierte Firewall und ist laut Hersteller ein entsprechend gehärtetes Device. Kann ein Design mit einem AudioCodes SBC direkt im „Transfernetz“ vom Provider mit integrierter aktiver Firewall und dahinter erst eine separate interne Firewall als ein mehrstufiges Firewall-Konzept mit externer und interner Firewall angesehen werden? Der Vorteil der integrierten Firewall ist wie auch im Artikel beschrieben, dass der SBC die Ports für die Audioverbindungen zuverlässig freigeben und blockieren kann und auch kein NAT notwendig ist. Es kommt nur VoIP vom Provider, die Firewall im SBC routet keinen anderen Daten-Traffic. Der SBC und die angeführten VLANs in der Zeichnung sind nur für VoIP vorgesehen.

ANDREAS SCHRÖCKER, VIA E-MAIL

Nach Einschätzung des Autors des Artikels handelt es sich um ein mehrstufiges Konzept, jedoch sieht er ein dediziertes „filterndes Device“ an der Netzübergabeschnittstelle als sinnvoller an, um bei einer eventuellen Kompromittierung des SBC einen zusätzlichen Paketfilter (ggf. auch recht grob) davor zu haben.

In diesem Fall könnte dann auf dem Layer-2-Switch im TF-Netz vom Provider eine Port-ACL (Paketfilter auf physischer Portebene) auf die SIP- und RTP-Ports sowie die externe IP-Adresse des SBC gelegt werden. Damit hätte man einen groben statuslosen Paketfilter. Die dynamische Öffnung der RTP-Ports übernimmt dann der SBC selbst.

Die benötigten Adress- und Portinformationen können bei Audiocodes Mediant SBCs im Normalfall aus dem IP-Interface, dem SIP-Interface und dem Media-Realm ausgelesen werden.

Keine ARM-Systeme erhältlich

(*iX* extra Embedded Systems: Trends auf der embedded world; *iX* 3/2020, S. 102)

Der Eindruck ist in der Tat, dass der ganze Embedded-Markt momentan einem großen Dreckhaufen entspricht. ARM-Server sind zum Beispiel für die Allgemeinheit nicht zu haben, dafür müsste man in Richtung High Performance irgendwelche Bastel-Boards oder Einplatinenrechner nehmen und daraus einen Cluster bauen, was offensichtlich viel schwieriger und ineffizienter wäre, als gleich zig Rechenkerne in einem System vereinigt zu kaufen.

ANDI GLAESER, VIA E-MAIL

Docker und Kubernetes als Windows Server

(*iX* extra Hosting: Container aus der Cloud; *iX* 4/2020, S. 119)

Docker und Kubernetes wirken *nicht* dem Vendor Lock-in entgegen. Ganz im Gegenteil, hier droht eine Monokultur zu entstehen, mit einer Landschaft von Systemen, welche alle dieselben Sicherheitslücken gemeinsam haben. Die Einfachheit von Docker mag für manchen attraktiv erscheinen. Meiner Ansicht nach ist es nur dann sinnvoll, wenn zum Beispiel ein Webhoster so etwas wie die berühmten Ein-Klick-Installationen für Kunden anbieten will.

Wollte ich mir Docker auf meinem Notebook installieren, weil es dort beispielsweise irgendeine Appliance fertig als

Container vorgebaut gibt, deren Aufsetzen mir sonst zu umständlich und zu schwierig wäre, dann würden dafür heute 293 MByte Speicherplatz verwendet werden. Docker ist einfach zu teuer und lohnt sich nicht für einzelne Anwender. In 300 MByte lässt sich beinahe eine komplette FreeBSD-VM unterbringen, in der man sich dann auch relativ einfach mithilfe von Webmin die gewünschten Dienste konfigurieren könnte.

Docker und Kubenetes drohen zum neuen Windows Server zu werden: zu teuer und zu monolithisch. Oft fehlt das Fachwissen, wie es eigentlich funktioniert – und obwohl es theoretisch Open Source ist, ist der Code viel zu umfangreich.

ANDI GLAESER, VIA E-MAIL

Falsches HTML-Beispiel

(RegEx-Katastrophen: Wenn reguläre Ausdrücke Software lahmlegen; *iX* 5/2020, S. 110)

Zwei Anmerkungen zu dem aufschlussreichen Hintergrundartikel: Zum einen wäre es aus meiner Sicht sinnvoll gewesen, die Eigenschaften und Möglichkeiten von regulären Ausdrücken und die konkrete Implementierung der Parser, die häufig zum Einsatz kommen, zu trennen. So kommt diese wichtige Erkenntnis, dass man das Ganze auch anders implementieren kann, lediglich im letzten Absatz des Artikels vor.

Zum Zweiten halte ich den Hinweis bezüglich der Nutzung von regulären Ausdrücken am Beispiel von HTML für sachlich so verkürzt, dass er genau genommen falsch ist. Richtigerweise wird darauf hingewiesen, dass der Versuch naiv ist.

Allerdings wird hier leider eine ganz fundamentale Tatsache nicht erwähnt, nämlich dass es sich bei HTML um eine kontextfreie Sprache handelt, deren Ausdrucksmöglichkeiten mit regulären Ausdrücken beweisbar nicht erfasst werden können; Stichworte sind hier die Chomsky-Hierarchie im Allgemeinen sowie das „Pumping-Lemma für reguläre Sprachen" im Speziellen. Die direkte Konsequenz ist, dass jeder Versuch, reguläre Ausdrücke zum Erkennen oder Filtern von kontextfreien Sprachen (oder noch ausdrucksstärkeren) zu verwenden, zwangsweise gleichzeitig zu eng und zu weit bemessen ist – also falsche Erkennungen (falsch positiv) und falsche Nicht-Erkennungen (falsch negativ) liefert.

Als praktische Konsequenz ist zum Beispiel der Versuch, HTML-Injection (was zu Cross-Site Scripting führen kann) oder SQL Injection mit regulären Ausdrücken zu erkennen und zu verhindern, zum Scheitern verurteilt. Was damit in der Praxis sogar gefährlich ist, weil man sich in falscher Sicherheit wiegen könnte.

ULRICH KÜHN, VIA E-MAIL

Fast unmöglich zu beherrschen

(RegEx-Katastrophen: Wenn reguläre Ausdrücke Software lahmlegen; *iX* 5/2020, S. 110)

Der RegEx-Artikel spricht mir aus dem Herzen: Mächtige Werkzeuge verlangen ein fast unmenschlich großes Maß an Selbstkritik, um sie beherrschen zu können.

HELLFRIED SABATHY, VIA E-MAIL

Ergänzungen und Berichtigungen

Storage: HPEs Primera im Test; *iX* 5/2020, S. 72

Der Kopf des Artikels zeigt irrtümlich den mitgelieferten und als Lasttreiber genutzten Server HPE ProLiant DL380 Gen10. Gegenstand des Tests waren aber die beiden Storage-Arrays Primera A630 und A670.

Zwei Arrays von HPEs Primera-Serie spielten die Testkandidaten, hier das Modell Primera A670.

Hosting für Entwickler: Docker und Entwicklerwerkzeuge vom Hoster; *iX* 4/2020, S. 12

gridscale wird in der Übersicht nur als Anbieter von Platform as a Service gelistet. Das ist nicht korrekt, da der Provider unter anderem auch ein Managed-Kubernetes-Angebot im Portfolio hat.

Die *iX*-Redaktion behält sich Kürzungen und auszugsweise Wiedergabe der Leserbriefe vor. Die abgedruckten Zuschriften geben ausschließlich die Meinung des Einsenders wieder, nicht die der Redaktion.