HARDFIRE – ein Firewall-Konzept auf FPGA-Basis

Andreas Fießler[1], Sven Hager[2], Björn Scheuermann[2], Alexander von Gernler[1]

Firewalls sind heutzutage unverzichtbare Komponenten in Netzwerken. Große Herausforderungen stellen dabei die stetig steigende Netzwerkbandbreite und das wachsende Datenvolumen dar. Es ist nicht zu erwarten, dass sich dieses Wachstum in absehbarer Zeit verlangsamt. Softwarebasierte Firewalls stoßen selbst mit neuesten Rechnersystemen und einer Vielzahl von Optimierungen derzeit an ihre Grenzen und werden dadurch zum Flaschenhals in der Infrastruktur. Um die steigenden Anforderungen an die Übertragungsgeschwindigkeit bei gleichbleibendem Sicherheitsniveau bedienen zu können, ist der Einsatz von Hardwarebeschleunigung zwingend notwendig. Bisher werden im Hochgeschwindigkeitsbereich vor allem spezialisierte Hardware-Firewalls eingesetzt. Diese sind in der Entwicklung jedoch sehr aufwendig und zudem wenig flexibel. Nicht unterschätzt werden darf auch das Sicherheitsrisiko durch mögliche Hintertüren in der Hardware, da sich im Gegensatz zu Software Funktionen und Abläufe in fertigen Hardwarekomponenten weder einfach prüfen noch zertifizieren lassen. In diesem Beitrag wird die grundsätzliche Problematik von Firewalls im Hochgeschwindigkeitsbereich diskutiert und ein neuartiger Lösungansatz vorgestellt. Das Forschungsprojekt HARDFIRE[3] befasst sich mit der Entwicklung einer zustandshaltenden Firewall auf Basis von rekonfigurierbaren, logischen Schaltkreisen (Field Programmable Gate Array, FPGA). Im Gegensatz zu bisherigen Hardware-Firewalls wird hierbei eine für den jeweiligen Anwendungsfall maßgeschneiderte und optimierte Konfiguration angestrebt. Im Rahmen des Projekts werden außerdem Risiken durch Hardware-Hintertüren und Vorgehensweisen zu deren Vermeidung berücksichtigt.

Stichworte: Firewall, FPGA, Hardwarebeschleunigung, IT-Sicherheit, Hintertür

1. Einleitung

Die Entwicklung von Firewalls steht derzeit an einem Wendepunkt. Während lange Zeit die steigenden Anforderungen an die Übertragungsgeschwindigkeit durch den Einsatz von erhöhter Rechenleistung bedient wurden, gerät das Wachstum dieser Größen vermehrt in ein Missverhältnis. In Abbildung 1 werden die Wachstumskurven nach Interpretation von *Moore's Law* [1] und *Butter's Law* [2] visualisiert. Es wird deutlich, wie die benötigte Verarbeitungsgeschwindigkeit zunehmend von der verfügbaren divergiert.

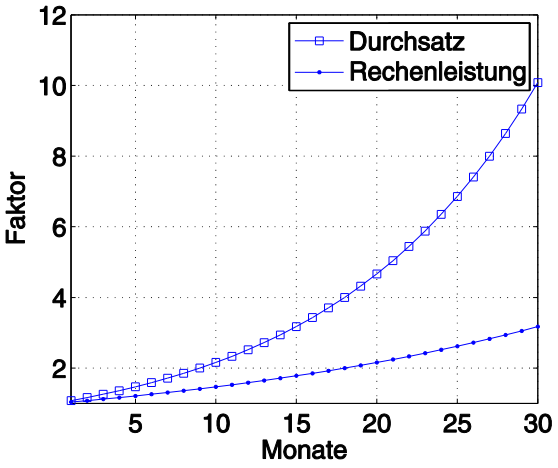

Abbildung 1: Wachstumskurven von Rechenleistung und Datendurchsatz nach *Moore's Law* und *Butter's Law.*

Im Hochgeschwindigkeitsbereich werden diese Probleme adressiert, indem diverse Möglichkeiten der Hardwarebeschleunigung genutzt werden. Die Bandbreite reicht dabei von einfachen *Offloading-Engines* (z.B. für Prüfsummen) bis hin zu komplexen Spezialprozessoren, welche auch höher-

[1] genua Gesellschaft für Netzwerk- und Unix-Administration mbH, Kirchheim bei München
[2] Humboldt-Universität zu Berlin, Berlin
[3] http://www.hardfire.org

schichtige Funktionen übernehmen. Die Hardwareentwicklung birgt im Vergleich zur Softwareentwicklung allerdings eine Vielzahl von Schwierigkeiten, wie hohe Einmalkosten und lange Entwicklungszyklen. Spätestens mit der Integration komplexer Firewall-Funktionalität muss zudem das selbstverständliche Vertrauen, das Hardwarekomponenten in der Praxis entgegengebracht wird, infrage gestellt werden [3].

2. Firewalls im Hochgeschwindigkeitsbereich: Probleme und derzeitiger Stand der Technik

Die Anforderungen an Firewall-Systeme sind seit ihrer Entstehung in den späten 80er Jahren stetig gestiegen [4]. Während die ersten Firewalls lediglich statische Paketfilter auf Basis von Netzwerkadressen, Transportschicht-Ports und -protokoll implementierten, ist dies heute bei Weitem nicht ausreichend, um ein Netzwerk vor den simpelsten Angriffen z.B. auf Basis von IP-Spoofing zu schützen [4]. Daher verwenden moderne Firewall-Systeme eine Reihe von weiteren Techniken, welche von zustandshaltender Paketfilterung [5] über Deep Packet Inspection (DPI) [6] bis hin zu spezialisierter Applikationskontrolle mithilfe von Application Level Gateways (ALG) reichen.

Während dieses Konglomerat aus verschiedenen Funktionalitäten (im Folgenden als Sicherheitssystem bezeichnet) in seiner Gesamtheit die Sicherheit eines abgeschirmten Netzes stark erhöhen kann, so erfordert es auch einen beträchtlichen Rechenaufwand, welcher den erreichbaren Netzwerkdurchsatz einschränken kann. Dies gilt insbesondere dann, wenn das Sicherheitssystem auf Basis einer herkömmlichen CPU in Software implementiert wird [7, 8].

Ungeachtet dieser Tatsache besteht der überwiegende Teil heutzutage im Einsatz befindlicher Firewall-Systeme aus einer (oftmals quelloffenen) Softwarekomponente, welche unter einem gängigen Betriebssystem auf einer herkömmlichen Hardwareplattform ausgeführt wird. Einige weitverbreitete Systeme sind beispielsweise iptables[4] und pf[5] sowie die Intrusion Detection Systeme (IDS) Snort[6] und Bro[7]. Die Bandbreite der zugrundeliegenden Hardware-Plattformen reicht hierbei von kleinen Heimroutern bis hin zu leistungsfähigen Serversystemen. Die o.g. Systeme treffen die Filterentscheidungen für jedes Paket über die Software auf der CPU. Dies bedeutet, dass jedes eingehende Netzwerkpaket über Busse, Speicher und verschiedene Betriebssystemkomponenten verarbeitet und sequentiell mit einem mitunter sehr großen Regelsatz verglichen bzw. die zugehörige Netzwerkverbindung in einem Verbindungsspeicher nachgeschlagen werden muss. Bei einer Übertragungsgeschwindigkeit von beispielsweise 100 Gbit/s bleiben dem System hier im Extremfall fünf Nanosekunden pro Paket (64 Byte) für all diese Operationen. Erschwerend hinzu kommt, dass viele softwarebasierte Systeme trotz der Existenz effizienter Verfahren [9, 10, 11] einen naiven Klassifikationsalgorithmus verwenden, um eingehende Pakete der entsprechenden zugehörigen Regel zuzuordnen [12]. Aktuelle Ergebnisse aus der Forschung zeigen, dass selbst

[4] The netfilter.org project. http://www.netfilter.org/
[5] OpenBSD Packet Filter. http://www.openbsd.org/faq/pf/
[6] Snort IDS. https://www.snort.org/
[7] Bro IDS. https://www.bro.org/

schnelle softwarebasierte Paketklassifikationssysteme wie Open vSwitch[8] auch unter guten Bedingungen kaum Datendurchsätze von 10 Gbit/s erreichen [13].

Weitere Resultate belegen, dass die Implementierung eines schnellen Klassifikationsalgorithmus in dedizierter Hardware zu einem um zwei Größenordnungen höheren Durchsatzfaktor im Vergleich zu einer CPU-Implementierung führen kann [8]. Somit kann den Performance-Limitierungen für Softwarelösungen mit verschiedenen Arten von Hardwarebeschleunigung begegnet werden. In höheren Geschwindigkeitsbereichen kommen spezielle Hardware-Firewalls zum Einsatz, die oftmals auf einem dedizierten, anwendungsspezifischen Schaltkreis (Application-Specific Integrated Circuit, ASIC) basieren. Besonders die in Hardware mögliche hochgradige Parallelisierung, wie sie zumeist in Assoziativspeichern (Ternary Content Addressable Memories, TCAMs) genutzt wird, sorgt dabei für den benötigten Performancegewinn [14]. ASICs weisen allerdings einige Nachteile auf: Sie sind nur wirtschaftlich, wenn eine gewisse Stückzahl in der Produktion erreicht wird oder physikalische Beschränkungen keine Alternativen zulassen. Die hierfür notwendige, universelle Auslegung steht wiederum im Gegensatz zur optimalen Lösung für ein bestimmtes Szenario. So ist beispielsweise das Abbilden von negierten Feldern oder Intervallen in Firewall-Regeln auf TCAMs problematisch, da diese Funktionalitäten aufgrund ihrer universellen Struktur nicht unterstützt werden [15]. Darüber hinaus bieten TCAMs im Vergleich zu herkömmlichen RAM-Bausteinen nur sehr wenig Speicherplatz und sind sowohl teuer als auch energieineffizient. All dies wirft die Frage nach alternativen Klassifikationsmöglichkeiten auf [16]. Ein weiterer großer Nachteil von kommerzieller dedizierter Hardware ist, dass die verwendeten Komponenten nicht einsehbar sind und somit potentiell Hintertüren beinhalten können.

Filterungsschaltkreise auf FPGAs zu implementieren ist eine Möglichkeit, diese Nachteile zu vermeiden. Diese Option wurde in den letzten Jahren insbesondere im Hinblick auf zustandslose Klassifikation mit großem Interesse verfolgt [17, 18, 19]. Während die existierenden Ansätze vielversprechend sind und hohe Durchsätze für zustandslose Paketklassifikation ermöglichen, so werfen sie jedoch zwei fundamentale Fragen auf:

- Wie lässt sich eine effiziente, zustandshaltende Komponente auf einem FPGA realisieren, welche das Parallelisierungspotential auf dem FPGA optimal ausnutzt?
- Die existierenden Ansätze verwenden FPGAs hauptsächlich als für geringe Stückzahlen kosteneffiziente Implementierungsplattform. Die verwendeten Algorithmen könnten prinzipiell als ASIC realisiert werden. Wie viel Performance lässt sich durch das Spezialisieren der auf dem FPGA implementierten Schaltung auf einen gegeben Regelsatz gegenüber einer universellen Auslegung gewinnen?

[8] http://openvswitch.org/

Das Ziel des HARDFIRE-Projektes ist es, diese beiden Fragen zu beantworten und so eine transparente Plattform zu schaffen, welche den Anforderungen an effiziente und sichere Filterungshardware genügt.

3. Vertrauenswürdige FPGA-Entwicklung

Lange Zeit fand das Potenzial für Sicherheitsrisiken durch Hardware wenig Beachtung. Aktuelle Untersuchungen und Enthüllungen bestätigen allerdings sowohl die Möglichkeit solcher Angriffe als auch die reale Gefahr. Beispielsweise können durch Kill-Switches Hardwarekomponenten irreversibel deaktiviert oder mit versteckten Schnittstellen unberechtigt geheime Informationen ausgelesen werden [20, 21, 3]. Mit höherer Komplexität und umfangreicherer Funktionalität steigen diese Risiken weiter an, da in Software vergleichsweise gut kontrollierbare Abläufe an geschlossene Hardware ausgelagert werden. Abbildung 2 zeigt einen exemplarischen Fall, in dem eine einfache Netzwerkkarte – vom Betriebssystem unbemerkbar – Speicherinhalte mit sensiblen Informationen ausliest. Die Daten werden anschließend von der bösartigen Netzwerkkarte eigenständig in einem unauffälligen Paket versteckt und an den Angreifer gesendet. Ein solcher Angriff ist durch den etablierten Speicherdirektzugriff (DMA) in nahezu jedem modernen Rechnersystem ohne aktive I/O Memory Management Unit (IOMMU) möglich.

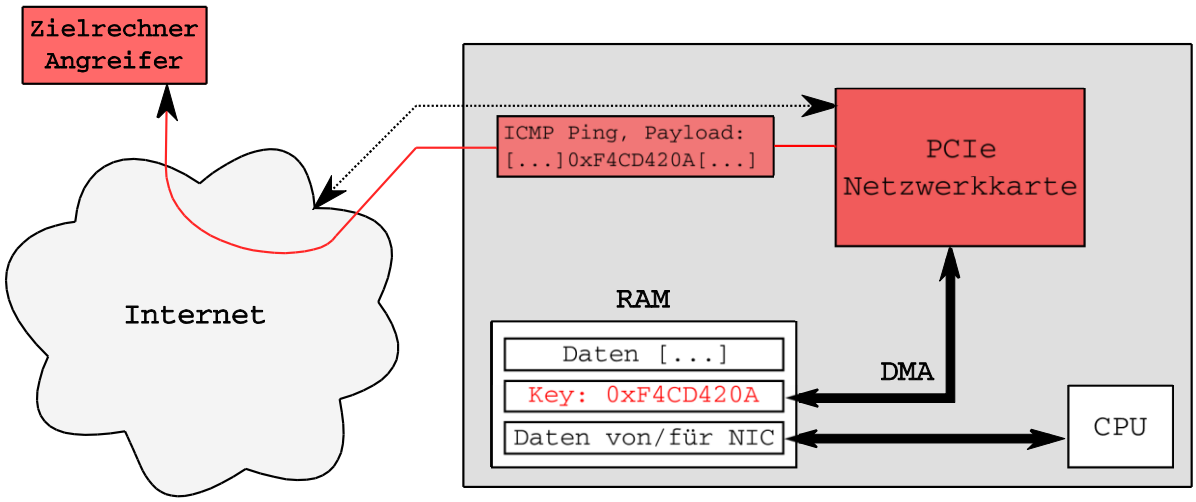

Abbildung 2: Angriffsszenario mit einer präparierten Netzwerkkarte per DMA.

Als Gegenmaßnahme können relevante Teile der Hardwarekomponenten selbst entworfen und implementiert werden – beispielsweise durch Eigenentwicklung auf Basis von FPGAs. Dies gewährleistet eine zertifizierbare Funktionalität auf dieser Ebene. Restlose Sicherheit kann allerdings auch mit der Verwendung eines FPGAs nicht garantiert werden. Beispielsweise bergen geschlossene, vorgefertigte Funktionsblöcke (sog. IP-Cores[9]), welche verbreitet zur Reduktion des Entwicklungsaufwands eingesetzt werden, vergleichbare Risiken wie integrierte Schaltkreise. Im Vorfeld des For-

[9] Im Kontext von IP-Cores steht die Abkürzung IP für Intellectual Property.

schungsprojekts wurden hierzu bereits verschiedene Möglichkeiten erarbeitet, um größtmögliche Sicherheit bei vertretbarem Mehraufwand in der Entwicklung zu erreichen. Ziel ist dabei, IP-Cores mit unbekanntem Vertrauensstatus weiterhin verwenden zu können.

Ein relativ einfach umsetzbares Verfahren ist die Verschlüsselung der im IP-Core verarbeiteten Daten (Scrambling). In vielen Fällen benötigt ein IP-Core zur Verarbeitung nur einen sehr kleinen Teil der Metadaten, beispielsweise Beginn und Ende eines Datenpakets. Bei einer effektiven Verschlüsselung der restlichen Daten können sowohl das Auslösen von Hintertüren in IP-Cores durch spezielle Sequenzen als auch der beschriebene DMA-Angriff verhindert werden, da durch die Verschlüsselung Auslösesequenzen nicht erkennbar sind bzw. ungültige Daten entstehen. Die Anwendung dieses Verfahrens auf einen kompromittierten IP-Core in einer FPGA-basierten Netzwerkkarte ist in Abbildung 3 veranschaulicht. Hier wird treiberseitig und in der Netzwerkkarte nach dem DMA-IP-Core der Datenstrom ver- bzw. entschlüsselt. Versucht der bösartige DMA-IP-Core hier Daten aus dem Speicher zu lesen, ist dies zunächst weiterhin möglich. Vor dem Versenden eines analog zum letzten Beispiel generierten Pakets werden die Daten allerdings entschlüsselt – der IP-Core kann hierfür keine gültig verschlüsselten Pakete erzeugen. Somit entstehen nach der Entschlüsselung schlicht ungültige Daten. Auf diese Weise können vom DMA-IP-Core keine unauthorisierten Pakete erzeugt und versendet werden.

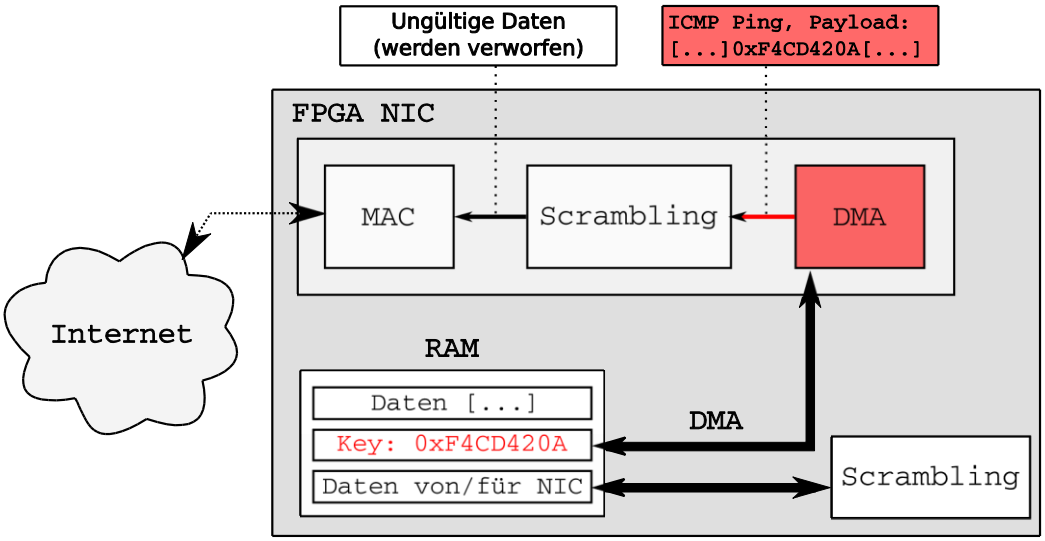

Abbildung 3: Verschlüsselung (Scrambling) schützt gegen DMA-Angriff.

Dieser Ansatz deckt allerdings nur einen Teil möglicher Angriffsvektoren in IP-Cores ab. Werden auf einem FPGA sensible Daten (z.B. Schlüsselmaterial) verarbeitet, besteht die Möglichkeit, dass ein IP-Core für Unbefugte direkt neue Zugriffswege auf diese Daten erlaubt. Realisierbar sind verschiedenste Formen, von undokumentierten, seriellen Schnittstellen bis hin zu Seitenkanälen über absichtliche Temperaturveränderung [22]. Um diesen Gefahren zu begegnen, können für IP-Cores im Vorfeld valide Zustände festgelegt werden. Eine Abweichung im Betrieb kann dann Hinweise auf versteckte Hintertüren geben – z.B. die Signalleitung, die zu deren Aktivierung dient. HAIDER et al. beschreiben hierfür ein geeignetes Toolkit [21].

3.1. In HARDFIRE verwendete Konzepte zur Hardware-Absicherung

HARDFIRE wird eine bis auf Hardware-Ebene nachvollziehbare und abgesichterte Implementierung bereitstellen. Die hierzu umgesetzten Maßnahmen umfassen:

- Einsatz von Scrambling, vor allem an der Schnittstelle zum Hostsystem
- Substitution von verschlüsselten IP-Cores durch quelloffene Versionen, sofern vorhanden oder der Entwicklungsaufwand dies zulässt
- Zertifizierbarkeit der zugänglichen Bestandteile
- Verschlüsseltes Update-Verfahren für die FPGA-Konfiguration, welches ausschließlich von einem dedizierten System (Rechner, der das FPGA-Modul enthält) zugänglich ist
- Umsetzung der in [21] vorgestellten Erkennungsmethoden für Hintertüren in IP-Cores

3.2. Betrachtung verbleibender Angriffsvektoren

Im Rahmen der konzeptionellen Arbeiten konnten weitere Risikofaktoren identifiziert werden:

- Die Design-Tools für FPGAs könnten an vielen Stellen die Funktion der erzeugten Hardware manipulieren. Vergleichbar hiermit ist ein bösartiger Compiler, der Software unbemerkt mit Hintertüren oder Schwachstellen versieht [23].
- Die Fertigung von ICs wird oft an externe Hersteller ausgelagert. Auch hier besteht ein Risiko nachträglicher Modifikationen. Zudem können Fälschungen der entsprechenden Schaltkreise in den Produktionszyklus gelangen. Gleichermaßen betrifft dies auch die neben dem eigentlichen FPGA verbauten Komponenten eines Systems.
- Ein Angreifer kann versuchen, unbemerkt eine eigene Konfiguration (Bitfile) in den FPGA einzuspielen. Am Markt etablierte FPGAs bieten hiergegen zwar Schutzmechanismen in Form von verschlüsselten Updates. Diese zeigten in der Praxis allerdings bereits Schwachstellen [24].

Die ersten beiden Punkte erfordern auf Seite des Angreifers sehr detailliertes A-Priori-Wissen über den speziellen Anwendungsfall. Da sowohl Design-Tools als auch FPGAs für ein breites Spektrum an Aufgaben konzipiert werden, kann das Risiko für solche Angriffe als gering eingestuft werden. Bösartige Konfigurationsupdates können hingegen speziell für ein fertiges System ausgelegt werden. Zu dieser Problematik sei auf Abschnitt 5 verwiesen.

4. Konzept HARDFIRE

Das Ziel von HARDFIRE ist, neue Wege aus den im Bereich der Hochleistungs-Firewalls bestehenden Problemen aufzuzeigen und dabei eine Alternative sowohl zu reinen Softwarelösungen als auch zur aufwendigen ASIC-Entwicklung zu bieten. Hierfür wird ein speziell entwickelter Workflow eingesetzt. Die HARDFIRE-Toolchain liest einen vom Systemadministrator vorgegebenen Regelsatz ein und erzeugt daraus, wie

in Abbildung 4 veranschaulicht, einen spezialisierten, regelsatzspezifischen Schaltkreis. Dieser wird zunächst in der Hardwarebeschreibungssprache VHDL generiert, aus der dann wiederum eine FPGA-Konfiguration synthetisiert werden kann.

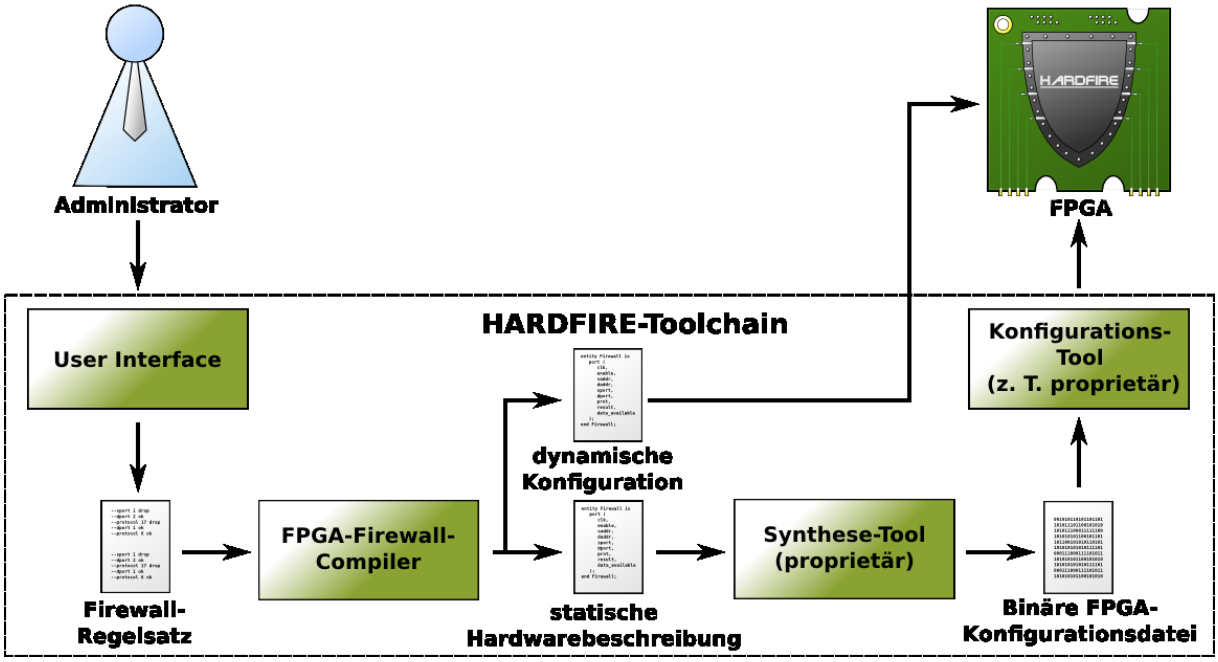

Abbildung 4: Workflow eines HARDFIRE-Systems.

Der FPGA wird so als Spezialprozessor genutzt, der nicht nur an die generelle Aufgabe („Firewalling“), sondern darüber hinaus an die spezifische Probleminstanz, also den konkreten Regelsatz, angepasst ist. Hierdurch gewinnt HARDFIRE einen entscheidenden Vorteil gegenüber ASIC-basierten Firewalls: Bei HARDFIRE ist im Vorfeld der Erzeugung der logischen Schaltung der Regelsatz bekannt. Deshalb kann der genutzte Schaltkreis sehr detailliert an den Regelsatz und damit den konkreten Einsatzort und -zweck angepasst werden. Einzelne Filterkriterien, beispielsweise Adressmasken oder Portbereiche, müssen bei der Paketverarbeitung nicht jeweils aus einem Regelspeicher gelesen, sondern können unmittelbar in der Verarbeitungslogik integriert werden. Des Weiteren kann die Verarbeitung selbst großer Regelsätze hochgradig parallel erfolgen. Die statische Rekonfiguration des FPGA bei umfangreichen Regelsatzänderungen wird ergänzt um Möglichkeiten, dynamisch punktuelle Konfigurationsaktualisierungen ohne eine vollständige, zeitaufwendige Neusynthese vorzunehmen.

Der zunächst vorhandene, bauartbedingte Nachteil von FPGAs gegenüber ASICs – ein identischer Schaltkreis arbeitet auf einem FPGA deutlich langsamer als auf einem ASIC [25] – kann durch die regelsatzspezifische Hardware-Generierung kompensiert werden: Die regelsatzspezifische, in HARDFIRE erzeugte Schaltung ist deutlich spezialisierter, da sie nur die für die jeweilige Aufgabe unbedingt notwendige Logik enthält. Deshalb ist sie wesentlich kleiner als eine generische ASIC-Lösung mit vergleichbarer Verarbeitungsparallelität. Die Kompaktheit der auf dem FPGA zu konfigurierenden Logik erhöht die Performanz beträchtlich. Somit wird eine sowohl flexible als auch hochperformante Firewall-Lösung erreicht. Die Generierung der Hardwarebeschrei-

bung ist dabei integraler und vollständig automatisierter Bestandteil der Toolchain. Der Administrator benötigt daher keine speziellen Hardware-Kenntnisse, um HARDFIRE einzusetzen. Die prinzipielle Anwendbarkeit des geschilderten Wegs wurde von den Autoren in Vorarbeiten bereits für zustandslose Paketfilter unter Beweis gestellt [7]. Im HARDFIRE-Projekt wird nun, darauf aufbauend, eine vollständige Firewall-Lösung entstehen. Diese wird auch komplexere Funktionalität wie beispielsweise Connection Tracking bieten und somit die Grundlage für einen Einsatz in Produktivumgebungen legen. Abbildung 5 veranschaulicht die Bestandteile und den internen Datenfluss in der HARDFIRE-Verarbeitungslogik.

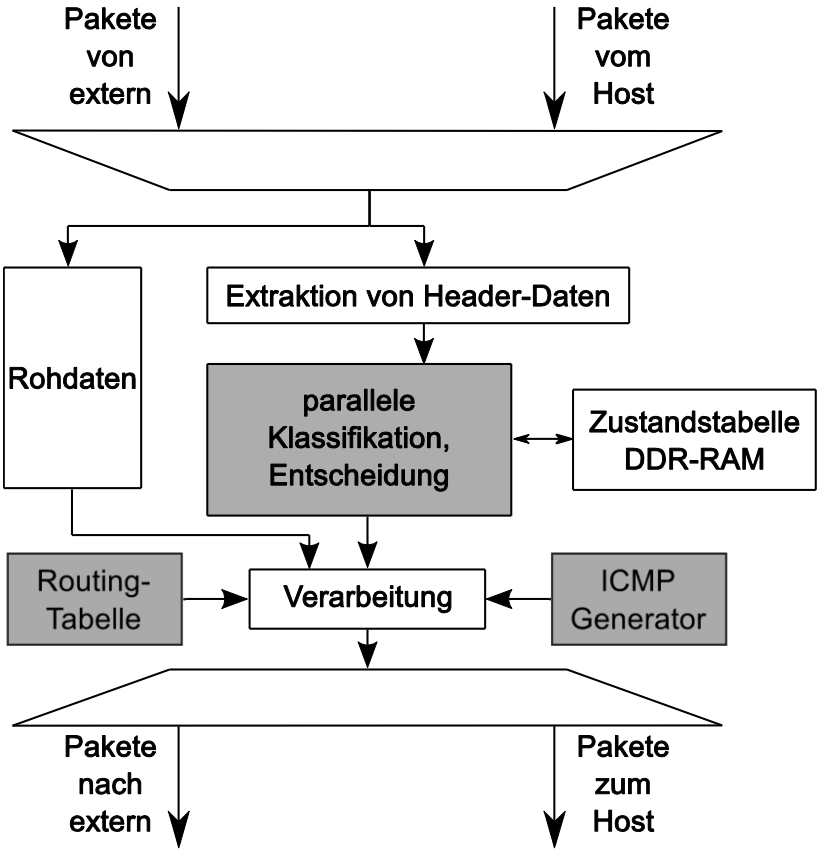

Abbildung 5: Interner Datenfluss in HARDFIRE. Die weißen Bereiche sind statisch, während die grauen Bereiche dynamisch für das Einsatzszenario generiert werden.

Bereits untersucht werden konnte, welche Update-Verfahren sich für welche Art von Konfiguration eignen:

- Vollständige Synthese eines Bitfiles: Erfordert umfangreiche Software des FPGA-Herstellers und einen mehrstündigen Zeitaufwand zur Erzeugung des Bitfiles und kommt daher für die optimierte, maßgeschneiderte Erstkonfiguration und umfassende Änderungen zum Einsatz.
- Partielle Rekonfiguration: Hiermit können Teile der im FPGA implementierten Logik während des Betriebs ausgetauscht werden. Diese erfordert weiterhin die Entwicklungssoftware und einen zwar kürzeren, aber dennoch signifikanten Zeitaufwand. Problematisch kann zudem sein, das System trotz Update in ei-

nem konsistenten Zustand zu halten. Dieses Verfahren ist daher derzeit nicht in HARDFIRE vorgesehen.

- Dynamische Konfiguration: Diese erfolgt direkt am Hostsystem über eine definierte Schnittstelle zur HARDFIRE-Hardware. So kann z.B. für kleine Regeländerungen eine schnelle Anpassung ermöglicht werden.

5. Sicherheitsaspekte des Konzepts

Neben den in Kapitel 3 beschriebenen Vorteilen durch eine tiefgreifendere Kontrolle über die Abläufe in der Hardware müssen bei HARDFIRE zusätzliche Details beachtet werden, um eine sichere Konfiguration zu gewährleisten. Ausschlaggebender Unterschied ist die Rolle des Firewall-Herstellers. Dieser stellt bei klassischen Firewalls das System und generische Software bzw. Firmware-Updates bereit. Bei HARDFIRE ist eine Instanz notwendig, die mittels der proprietären FPGA-Design-Software spezifische Bitfiles generieren kann. Dies kann beim Administrator selbst erfolgen, wozu dieser allerdings die entsprechende, teure Entwicklungsumgebung des FPGA-Herstellers benötigt. Alternativ ist eine Aufteilung durch Integration des Firewall-Herstellers denkbar. Dieser Ablauf ist in Abbildung 6 dargestellt. Hier würden einfache Änderungen, welche kein neues Bitfile erfordern, direkt durch den Administrator am System durchgeführt. Komplexere Modifikationen können standardisiert an den Hersteller kommuniziert werden, welcher daraufhin ein neues Bitfile für den FPGA erzeugt.

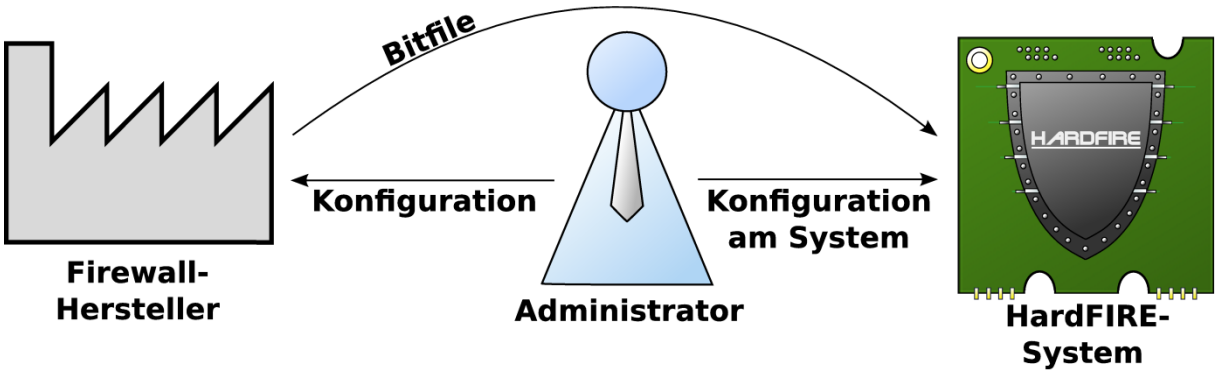

Abbildung 6: Mögliches Beziehungsmodell der in HARDFIRE beteiligten Parteien, hier ohne die Notwendigkeit der komplexen Design-Software beim Endanwender.

Problematisch ist hierbei die Tatsache, dass einer weiteren Partei (dem Firewall-Hersteller) detaillierte Informationen über die Netzwerkinfrastruktur übermittelt werden müssen und der Administrator nur schwer überprüfen kann, ob das Bitfile tatsächlich seinen Anforderungen entspricht. Daher erfordert dieses Vorgehen eine Vertrauensbeziehung zwischen Hersteller und Administrator, bzw. dem Endkunden. Zusätzlich sind folgende Maßnahmen notwendig und vorgesehen:

- Asymmetrische Verschlüsselung und Signatur der Konfigurationsdaten zur Übermittlung an den Firewall-Hersteller.
- Symmetrische Verschlüsselung des Bitfiles für den im FPGA fest konfigurierten Schlüssel, zudem zusätzliche asymmetrische Verschlüsselung und Signie-

rung des verschlüsselten Bitfiles. Auf diese Weise wird der Kommunikationsweg bis zum HARDFIRE-System selbst abgesichert.

In weiteren Schritten ist zudem denkbar, das FPGA-Bitfile nicht wie allgemein üblich verschlüsselt in einem ROM abzulegen, sondern durch ein weiteres System beim Start aus einer signierten Datei zu lesen, die Integrität zu prüfen und erst dann auf den FPGA zu laden.

6. Fazit und Ausblick

Die Performancegewinne einer hardwarebasierenden Lösung werden durch den Einsatz eines rekonfigurierbaren FPGAs mit der Flexibilität einer Softwarelösung kombiniert. Da bereits in der Konzeption durch den Workflow eine einfache, transparente Administrierbarkeit vorgesehen ist, kann ein HARDFIRE-System mit geringem Aufwand in bestehende Strukturen integriert werden. Mit der Eigenentwicklung der Kernfunktionalität auf Hardware-Ebene und dem Einsatz der beschriebenen Gegenmaßnahmen wird zudem sichergestellt, dass keine versteckten Hintertüren die Firewall kompromittieren können.

Die im Forschungsprojekt gewonnenen Erkenntnisse und Ergebnisse dienen zudem als Grundlage für eine Vielzahl von verwandten Problemstellungen. Hervorzuheben sind hier besonders die im Projekt behandelten Teilbereiche Stateful Matching sowie die verschiedenen Möglichkeiten zur dynamischen Rekonfiguration. Auf deren Basis können Anwendungen wie Datendioden, komplexe Offloading-Karten, Last- und Hochverfügbarkeitssysteme auf standardisierter FPGA-Hardware realisiert werden.

7. Danksagung

Das Forschungsprojekt wird im Rahmen des Förderprogramms ZIM-KF durch das Bundeswirtschaftsministerium finanziert.

Literaturverzeichnis

[1] M. Kanellos, „CNET News," 10. Februar 2003. [Online]. Available: http://news.cnet.com/2100-1001-984051.html. [Zugriff am 9. Januar 2015].

[2] G. Robinson, „EETimes," 26. September 2000. [Online]. Available: http://www.eetimes.com/document.asp?doc_id=1142186. [Zugriff am 9. Januar 2015].

[3] B. Gellman und E. Nakashima, „U.S. spy agencies mounted 231 offensive cyber-operations in 2011, documents show," The Washington Post, 30. August 2013. [Online]. Available: http://tinyurl.com/l2fak3y. [Zugriff am 8. Januar 2015].

[4] K. Ingham und S. Forrest, „A History and Survey of Network Firewalls," University of New Mexico, 2002.

[5] M. Gouda und A. Liu, „A Model of Stateful Firewalls and its Properties," in *DSN '05: Proceedings of the 35rd International Conference on Dependable Systems and Networks*, 2005.

[6] T. AbuHmed und A. N. D. Mohaisen, „A Survey on Deep Packet Inspection for Intrusion Detection Systems," in *arXiv preprint arXiv:0803.0037*, 2008.

[7] S. Hager, F. Winkler und B. Scheuermann, „MPFC: Massively Parallel Firewall Circuits," in *LCN '14: Proceedings of the 39th IEEE Conference on Local Computer Networks*, 2014.

[8] M. Varvello, R. Laufer, F. Zhang und T. Lakshman, „Multi-Layer Packet Classification with Graphics Processing Units," in *CoNEXT '14: Proceedings of the 10th International Conference on Emerging Networking Experiments and Technologies*, 2014.

[9] F. Baboescu und G. Varghese, „Scalable Packet Classification," in *SIGCOMM '01: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001.

[10] V. Srinivasan, S. Suri und G. Varghese, „Packet Classification Using Tuple Space Search," in *SIGCOMM '99: Proceedings of the 1999 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 1999.

[11] B. Vamanan, G. Voskuilen und T. Vijaykumar, „EffiCuts: Optimizing Packet Classification for Memory and Throughput," in *SIGCOMM '10: Proceedings of the 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2010.

[12] S. Hager, S. Selent und B. Scheuermann, „Trees in the List: Accelerating List-based Packet Classification with Controlled Rule Set Expansion," in *CoNEXT '14: Proceedings of the 10th International Conference on Emerging Networking Experiments and Technologies*, 2014.

[13] P. Emmerich, D. Raumer, F. Wohlfahrt und G. Carle, „Performance Characteristics of Virtual Switching," in *CloudNet '14: Third IEEE International Conference on Cloud Networking*, 2014.

[14] K. Lakshminarayanan, A. Rangarajan und S. Venkatachary, „Algorithms for Advanced Packet Classification with Ternary CAMs," in *SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2005.

[15] Y.-K. Kim, J.-G. Ki, S.-S. Lee und Y.-S. Kim, „Extended TCAM for Adopting Range and Negation Rules in Packet Classification," in *APCC '06: Proceedings of the 12th Asia-Pacific Conference on Communications*, 2006.

[16] F. Baboescu, S. Singh und G. Varghese, „Packet Classification for Core Routers: Is there an alternative to CAMs?," in *INFOCOM '03: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, 2003.

[17] W. Jiang und V. Prasanna, „Large-scale Wire-speed Packet Classification on FPGAs," in *FPGA '09: Proceedings of the ACM/SIGDA 17th International Symposium on Field Programmable Gate Arrays*, 2009.

[18] W. Jiang und V. Prasanna, „Scalable Packet Classification on FPGA," in *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 20 (2012), sep, Nr. 9*, 2012.

[19] H. Song und J. Lockwood, „Efficient Packet Classification for Network Intrusion Detection using FPGA," in *FPGA '05: Proceedings of the ACM/SIGDA 13th International Symposium on Field Programmable Gate Arrays*, 2005.

[20] S. Skorobogatov und C. Woods, „Breakthrough silicon scanning discovers backdoor in military chip," in *CHES '12: Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems*, 2012.

[21] S. K. Haider, C. Jin, M. Ahmad, D. M. Shila, O. Khan und M. v. Dijk, *HaTCh: Hardware Trojan Catcher,* 2014.

[22] J. Brouchier, T. Kean, C. Marsh und D. Naccache, „Temperature Attacks," in *IEEE Security and Privacy 7*, 2009.

[23] K. Thompson, „Reflections on Trusting Trust," in *Commun. ACM, 27(8)*, 1984.

[24] S. Skorobogatov und C. Woods, „In the blink of an eye: There goes your AES key," in *IACR Cryptology ePrint Archive, 2012:296*, 2012.

[25] I. Kuon und J. Rose, „Measuring the Gap Between FPGAs and ASICs," in *FPGA '06: 2006 Fourteenth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2006.