DNSSEC vs. DNSCurve for Securing the Net

Daniel Raumer
Betreuer: Ralph Holz
Seminar Future Internet SS2011
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: daniel.raumer@mytum.de

KURZFASSUNG

DNSSEC und DNSCurve sind zwei Alternativen zur Absicherung des Domain Name Systems (DNS). Vom abgesicherten DNS wird erwartet, dass dieses eine Möglichkeit bietet, die momentan anfällige X.509 PKI abzulösen. Nach einer kurzen Einführung zu DNS und einiger Angriffszenarien, wird die Funktionsweise der beiden Protokolle vorgestellt. DNSSEC baut hierbei durch Signieren der im DNS verwendeten Resource Records eine für jeden Teilnehmer überprüfbare Vertrauenskette vom Rootserver bis zum zu überprüfenden Record auf, während CurveCP durch Verschlüsselung zusätzlich auch Vertraulichkeit schafft und so bei direkter Kommunikation mit einem Nameserver Vertrauen in dessen Antwort gesichert werden kann. Beide Ansätze werden miteinander verglichen und in Bezug auf Performanz, Schwachstellen und sich eröffnende Möglichkeiten, wie die Ablösung der existierenden X.509 PKI, bewertet. Dabei werden unter Anderem Argumente aus der laufenden Debatte zwischen Dan Kaminsky und Daniel J. Bernstein, für und gegen die jeweiligen Protokolle, vorgestellt.

Schlüsselworte

Domain Name System Security, DNSSEC, DNSCurve, Authentisierungsprotokoll

1. MOTIVATION FÜR DNSSEC

Das 1983 erfundene und 1986 standardisierte Domain Name System (DNS) [11, 14, 15] liefert ein Mapping von menschenlesbaren Adressen auf in Netzwerken gebräuchliche Internet Protokoll (IP) Adressen. In seiner Basisversion gibt es keinen Mechanismus zur Sicherstellung von Authentizität und Integrität dieser Informationen, so dass bereits nach kurzer Zeit (1990) die ersten Angriffe beschrieben wurden [2]. Das Vertrauen eines Benutzers in die Authentizität Anderer beruht meistens nicht auf der IP-Adresse, sondern auf dem Domainnamen. Dies gilt oft auch wenn Zertifikate verwendet werden, da beispielsweise beim erstmaligen Beziehen der Zertifikate oft das DNS verwendet wird. Angriffe, die das DNS oder dessen Antworten manipulieren, umgehen darauf folgende Sicherheitsmechanismen, wenn der sichere Kanal nicht mit dem eigentlichen Ziel aufgebaut wird. Der 1995 beschriebene erste[1] Ansatz zur Absicherung von DNS, DNS Security (DNSSEC) [8], wurde auf Grund zu aufwändiger Schlüsselverwaltung jedoch nicht angenommen. Daher wurde zehn Jahre später eine neue überarbeitete Fassung veröffentlicht [1]. In der aktuellen Fassung wird DNSSEC von einer zunehmenden Anzahl an Nameservern unterstützt. Dennoch ist der Einsatz bis heute noch sehr gering. Eine Verbreitung von DNSSEC eröffnet neue Möglichkeiten und Potentiale bei der Nutzung des DNS.

Nach einer kurzen Einführung über DNS, soll DNSSEC zunächst in aktueller Fassung beschrieben werden. Anschließend wird mit DNSCurve ein alternativer Ansatz vorgestellt. Diese Ansätze werden bewertet. Danach werden Argumente aus der Performanzdebatte zwischen Dan Kaminsky und Daniel J. Bernstein um DNSSEC und die Alternative, DNSCurve, vorgestellt.

1.1 DNS - eine kurze Übersicht

Das DNS [14, 15] ist eine verteilte, hierarchisch aufgebaute Datenbank zur Übersetzung zwischen für Menschen verstehbaren Hostnamen und IP-Adressen. Abbildung 1 zeigt

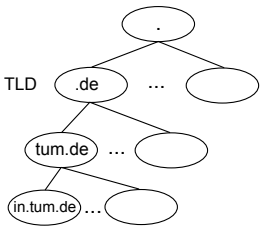

Abbildung 1: DNS Domains

dabei den hierarchischen Aufbau einer DNS Domain und damit der dahinter liegenden Anordnung der Server, durch die eine zur Absicherung nötige Public Key Infrastruktur (PKI) vorgegeben wird. Eine PKI ist ein System, das neben Benutzern, Zertifikaten und Speicher- beziehungsweise Abruforten für Zertifikate, eine hierarchisch angeordnete Struktur von Zertifizierungsstellen beinhaltet. Dabei signiert eine teilnehmende Zertifizierungsstelle (CA) der höheren Ebene, die öffentlichen Schlüssel (Public Keys), der in der Hierarchie unter ihr liegenden Teilnehmer. An oberster Stelle steht die sogenannte Wurzel, welche der oberste Vertrauensanker einer PKI ist. Vertrauensanker bezeichnet hierbei eine Autorität, deren Vertrauenswürdigkeit nicht mehr durch eine Signatur auf eine andere vertrauenswürdige Autorität zurückgeführt wird. Auch Certificate Revocation Lists (CRLs), die gesperrte Zertifikate auflisten, so wie eine Registration Authority (RA) sind oft Teil einer PKI [16].

[1]TSIG kann auf Grund der Nichtpraktikabilität in großen Netzwerken im WWW nicht verwendet werden und wird daher in dieser Arbeit vernachlässigt

doi: 10.2313/NET-2011-07-2_08

1.1.1 Resource Records

Im DNS werden Webadressen auf die sogenannten **Resource Records** (RRs), oft auch nur Records genannt, abgebildet. Diese können verschiedene Typen haben:

A: Der A Record wird genutzt, um einen Hostnamen auf eine IPv4-Adresse zu mappen. Zum Mapping auf IPv6-Adressen wurde das AAAA Record eingeführt.

CNAME: Der CNAME Record definiert ein Alias zu einem anderen Namen.

MX: Der MX Record bildet einen Namen auf einen Mailserver ab.

NS: Der NS Record verweist auf den in der Hierarchie darunter liegenden Nameserver, wenn der angefragte Nameserver den angefragten Bereich delegiert hat. Somit liefert es den Anfragenden, die dieses Record erhalten an einen speziellere Nameserver weiter.

TXT: Das TXT Record erlaubt beliebige, anwendungsspezifische Texte im DNS abzulegen.

Mit DNSSEC eingeführte RRs werden im weiteren Verlauf der Arbeit vorgestellt.

1.1.2 Nameserver

RRs werden von sogenannten **Nameservern** verwaltet. Einen Nameserver, der für einen bestimmten Bereich nicht mehr auf einen in der Hierarchie darunter liegenden Nameserver verweist, nennt man einen autoritativen Nameserver. Ein Anfrager gibt an, welchen RR er haben möchte. Auf Grund des Mappings zusammengehörende RRs werden als Set bezeichnet. In einem Set besteht keine Ordnung. Jedoch werden die RRs oftmals nach einer Round Robin Strategie abgewechselt, um Lastbalanzierung zu erreichen. Die Gesamtheit aller Adressen, für die ein Nameserver zuständig ist, wird Zone genannt. Eine angefragte Domain wird von rechts nach links abgearbeitet und so aufgelöst. Die einzelnen Hierarchieebenen werden dabei durch einen Punkt getrennt. Diese erste Ebene wird als Top-Level-Domain (TLD) bezeichnet.

1.1.3 Resolver

Das Auflösen einer Adresse wird vom **Resolver** durchgeführt. Der Resolver ist dabei ein Softwaremodul, das auf dem Rechner eines DNS-Teilnehmers beheimatet ist und die Informationen von Nameservern abruft. Er bildet also die Schnittstelle zwischen Anwendungen und Nameserver. Der Resolver löst eine Anfrage einer Anwendung auf, indem er sie um zur Auflösung nötige Informationen ergänzt und an einen normalerweise fest zugeordneten Nameserver, den sogenannten Rootserver, übermittelt. Das Auflösen kann rekursiv oder iterativ erfolgen. Beim rekursiven Auflösen leitet ein Nameserver die Anfrage so lange weiter, bis die Adresse des angefragten Server feststeht, während sich beim iterativen Auflösen der Client von Nameserver zu Nameserver in der Hierarchie nach unten arbeitet. Normalerweise arbeiten Nameserver rekursiv und liefern dem anfragenden Resolver die vollständige Antwort. Bei stark ausgelasteten Servern wie den Root-Servern ist die Rekursion jedoch deaktiviert um die Belastung für das weitere Auflösen einer Anfrage auf die anfragenden Resolver zu verteilen. In der Antwort wird dem Anfragenden dann mittels des Authoritative Response Flags mitgeteilt, ob ein Request autoritativ, also direkt aus einer lokalen Zonendatei, oder iterativ beziehungsweise rekursiv aufgelöst wurde.

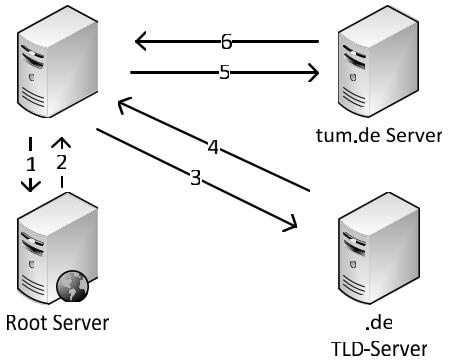

Abbildung 2: iterative Adressenauflösung über DNS

Abbildung 2 zeigt das schrittweise iterative Auflösen der Adresse www.tum.de unter der Voraussetzung, dass noch keine Adressen gecached sind. Zuerst wird vom Rootserver die Adresse des TLD-Servers zu „.de“ erfragt. An diesen Server wird dann die Frage nach dem tum.de Server gerichtet, der dann wiederum eine IP Adresse für „www.tum.de“ zurücksendet. Gewöhnlich werden Anfragen eine definierte Zeit vom Anfragenden selbst, aber auch vom Internet Service Provider gecached, um einerseits Datenverkehr aus dem Providernetz heraus zu verhindern und vor allem kürzere Latenzen zu ermöglichen. Auch kann es vorkommen, dass bestimmte, meist häufig angefragte, IP-Adressen bereits vom Server der höheren Hierarchieebene vollständig ausgeliefert werden.

1.2 Angriffszenarien

Dieser Abschnitt beschreibt Angriffszenarien, die Schwachstellen im DNS ausnutzen. DNS-Angriffe zielen meist darauf ab, durch Manipulation DNS-Nutzer auf falsche Webseiten zu lenken, um anschließend sensible Benutzerdaten durch Phishing zu erhalten. Aber auch Denial-of-Service-Attacken (DoS-Attacken) auf DNS-Server können Schaden anrichten, da wie beschrieben viele Anwendungen den Resolver und damit DNS Dienste nutzen und deren Funktionalität von diesen abhängt. Dieses Kapitel beschreibt **DNS-Spoofing**, **Cache Poisoning** und **DoS-Angriffe** mit, beziehungsweise auf, das DNS.

1.2.1 DNS-Spoofing

Als DNS-Spoofing wird das Manipulieren einer DNS-Anfrage und damit das Umlenken des Anfragers an eine andere Website bezeichnet. Möglich ist dies, indem beispielsweise eine gefälschte DNS-Antwort noch vor der offiziellen DNS-Antwort beim Anfrager ankommt. Da im normalen DNS kein Authentizitätsnachweis besteht, verarbeitet und behandelt dieser die Antwort wie die offizielle. Ein Benutzer kann so ohne dass er es merkt mit einem schädlichen Server kommunizieren. In Folge dessen werden viele der danach ablaufenden Sicherheitsmechanismen, wie SSL oder IPsec, ausgehebelt.

doi: 10.2313/NET-2011-07-2_08

1.2.2 *Cache Poisoning*

Als Cache Poisoning wird das „Verschmutzen“ eines Caches mit manipulierten Daten bezeichnet. Besonders wenn Integrität und Authentizität nicht überprüft werden, können manipulierte Daten somit verteilt werden. Ein alleiniges Sicherstellen der Authentizität eines Nameservers und der Aufbau eines sicheren Kanals hilft hierbei nicht, sofern dieser Nameserver bereits gefälschte Informationen enthält. Wird ein bereits beschriebener Spoofing Angriff auf die Anfrage hinter einem Cache durchgeführt merkt sich der Cache die falschen Daten. Handelt es sich hierbei jetzt nicht nur um einen Cache auf einem lokalen Rechner, sondern um einen größeren Cache, wie er beispielsweise beim Internet Service Provider (ISP) steht, erhalten auch alle anderen Anfragenden hinter dem betroffenen Cache die kompromittierte DNS-Antwort.

1.2.3 *DoS-Angriff*

Bei einem Distributed-Denial-of-Service-Angriff auf einen Nameserver (DDoS-Angriff) wird dieser durch einen hohen Datenstrom von DNS-Anfragen überlastet, so dass legitime Anfragen nicht mehr beantwortet werden können. Möglich wird dies vor allem dadurch, dass der Empfänger eines IP, beziehungsweise UDP-Pakets das Paket nicht annehmen muss, da dieses einfach geschickt wird. Diese Angriffe können nur schwer verhindert werden. Allerdings können sie durch ein Protokoll, das die Möglichkeit bietet, im Vergleich zur Anfragegröße viel größerer Antworten zu erzeugen, begünstigt werden. Auch der DNS-Amplification-Angriff ist ein Denial-of-Service-Angriff. Er macht sich die Vervielfachung der Datengröße durch einen DNS-Server zu Nutze. Das Opfer ist hierbei der Empfänger der DNS-Antworten. Die ausgenutzten DNS Server dienen hierbei lediglich als Verstärker des Angriffes.

Ziel eines sicheren DNS muss es also sein, diese Angriffslücken zu schließen, oder zumindest nicht zu vergrößern, ohne dabei neue Angriffe zu ermöglichen. Zusätzlich spielt der Preis an Komplexität und Einrichtungsaufwand eine entscheidende Rolle für die Akzeptanz einer neuen Technik. Sicherheit darf ein hoch skalierbares System, wie das DNS, nicht zu stark verlangsamen oder dessen Skalierung verschlechtern.

2. DNSSEC

DNSSEC kann den Ursprung und die Integrität von DNS-Daten sicherstellen. Es enthält ebenso Mechanismen, die die Nichtexistenz von DNS Daten überprüfen. Erreicht wird dies durch diverse Veränderungen und Erweiterungen des DNS-Protokolls. Es werden vier neue RR-Typen eingeführt: Resource Record Signature (RRSIG), DNS Public Key (DNSKEY), Delegation Signer (DS) und Next Secure (NSEC). Zusätzlich werden 2 neue Bits im Messageheader eingeführt: Checking Disabled (CD) und Authenticated Data (AD). DNSSEC macht dabei von bereits bekannten Erweiterungen wie Extended DNS Gebrauch, um auch DNS Pakete mit mehr als 512 Bytes senden zu können. Beim Design von DNSSEC wurde zu Gunsten der Performanz darauf geachtet, dass die Anzahl der kryptographischen Operationen eines Nameservers nicht mit der Zahl der Anfragen steigt [1].

2.1 Authentizität und Integrität von DNS Informationen

Ursprung und Integrität von DNS-Informationen werden sichergestellt, indem DNSSEC eine Digitale Signatur der DNS RRs anlegt und diese in einem neuen RR, dem RRSIG Record speichert. Wenn ein Resolver DNS-Informationen überprüfen will, kann er dies über die Signatur machen, indem er eine normale DNS Auflösung macht und dabei die Signatur anfrägt. Dieser kann er dann direkt vertrauen oder eine Vertrauenskette zu einer bereits vertrauenswürdigen Autorität bilden. Dafür wird der öffentliche Schlüssel im neuen DNSKEY RR gespeichert und kann so vom entsprechenden Nameserver selbst ausgeliefert werden. Dieses Record muss zum Aufbau einer Vertrauenskette wiederum von der höheren Instanz signiert sein. Momentan ist hierbei der Einsatz von RSA Verschlüsselung vorgesehen. Dennoch gibt es Entwicklungen modernere, kleinere und damit schnellere ECC[2] für DNSSEC zu verwenden [10]. DNSSEC basiert nicht auf gesicherten Verbindungen zwischen den Instanzen sondern lediglich auf gesicherten Objekten [1].

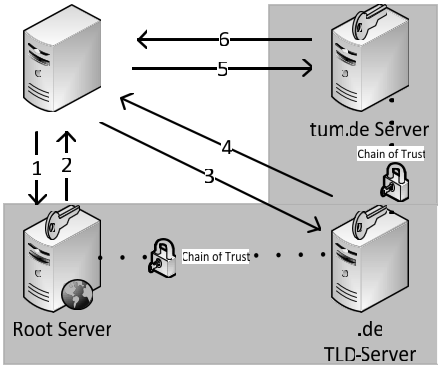

Abbildung 3: DNS mit DNSSEC

Abbildung 3 erweitert das Szenario aus Abbildung 2 um die zur Absicherung nötigen DNSSEC Elemente: Die im grauen Kasten enthaltenen Server benötigen jeweils einen öffentlichen und einen privaten Schlüssel. Durch gegenseitiges Signieren der Keys entsteht eine Vertrauenskette (gepunktete Linie), durch welche es dann möglich ist die RRs bezüglich Authentizität und Integrität auf einen Server höherer Ebene zurückzuführen und, bis hin zu einer Vertrauenswürdigen Autorität, zu überprüfen.

Auch delegierte Anfragen können mittels des DS RR-Types von DNSSEC überprüft werden, indem ein signiertes DS Record wiederum einen neues DNSKEY Record signiert. Dadurch werden auch komplexere Authentisierungspfade möglich, die es zum Beispiel auch ermöglichen, zusätzliche Ebenen und Pfade in der PKI einzuschieben.

2.2 Sicherstellen der Nichtexistenz von Namen oder Typen

Nachdem nun existierende RRsets signiert werden können, muss DNSSEC auch in der Lage sein, die Nichtexistenz eines

[2]Elliptic Curve Cryptography (ECC) bezeichnet asymmetrische Kryptosysteme, die Operationen auf elliptischen Kurven über endlichen Körpern verwenden.

doi: 10.2313/NET-2011-07-2_08

Namen oder RR Typs sicherzustellen.

2.2.1 NSEC Resource Record

Da ohne eine überprüfbare Negativantwort das unterdrücken einer Nachricht einen Angriff darstellen würde, muss es möglich sein auch Negativantworten zu verifizieren. Die bisher verwendete NXDOMAIN Nachricht kann hierbei nicht verwendet werden, da diese lediglich eine Negativantwort ist, die keinen eindeutigen Bezug zur Anfrage hat. Sie fällt daher auf jede Anfrage nach einer nicht existierenden Domain gleich aus. Daher ist der neue RR-Type NSEC eingeführt worden. Durch eine einheitliche Darstellung und Anordnung der Domainnamen in den Zonen kann ein Nameserver die ebenfalls signierte Information über den leeren Namensbereich schicken, indem er die nächsten existierenden Namen schickt [1]. Abbildung 4 zeigt NSEC am Beispiel eines Na-

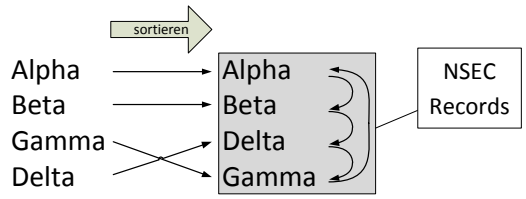

Abbildung 4: NSEC Beispiel

meservers, der die Zonen „Alpha“, „Beta“, „Gamma“ und „Delta“ hält. Hat Alice beispielsweise die nicht existierende Domain „Epsilon“ angefragt, bekommt sie das NSEC Record „Delta“ und „Gamma“ als Antwort, da dies die nächsten existierenden Zonen sind.

2.2.2 NSEC3 Resource Record

Allerdings ermöglicht das schrittweise Abfragen der NSEC RRs, also der leeren Namensbereiche, einen Rückschluss auf alle existierenden Namen der Zone. Da dieses Verhalten unerwünscht sein kann, wurde der NSEC3 RR eingeführt. Der NSEC3 RR ist ein Hash des nächsten existierenden Namens. Der Empfänger kann dadurch überprüfen, wenn der Hash des angefragten Namen kleiner ist als der empfangene [3]. Abbildung 5 verdeutlicht NSEC3 am Szenario aus Abbildung 4. Als Antwort auf Alices Frage nach der nicht existierenden Domain „Epsilon“ mit Hash „bx42“ erhält sie nun den Hash „bb3y“ und „c56b“ als Antwort. Sie kann nachprüfen, dass der hash von Epsilon dazwischen liegt

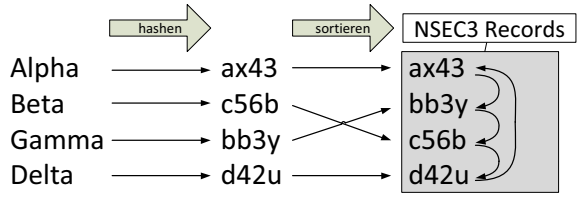

Abbildung 5: NSEC3 Beispiel

Hashwerte dieser Art können bei ausreichend kurzen Eingaben zurückgerechnet werden. Daher ist es dennoch mit genügend Aufwand möglich, die Namen zu erraten [7]. Da DNS öffentliche Informationen ohne Zugangsbeschränkungen bereitstellt, müssen geheime Informationen hinter einer Firewall versteckt werden. Das Problem kann dennoch durch sogenannte NSEC3 „White Lies“ gelöst werden. Ein Unsichtbarer Name wird beim Hash, den der Anfrager eines nichtexistierenden Namens erhält, einfach nicht berücksichtigt: Angenommen Domain „Alpha“ soll nur von Alice gesehen werden. Bob will wissen, ob die Domain „Epsilon“ mit dem Hash „ab42“ existiert. Da diese nicht existiert, antwortet der Nameserver mit einem NSEC3 Record. Bob wird nun aber nicht den Hash von „Alpha“ erhalten, sondern den Hash des nächst höheren Namens, „bb3y“.

2.3 Schlüsselmanagement in DNSSEC

Um eine PKI entlang der Domainhierarchie zu etablieren, nutzt DNSSEC pro Zone zwei verschiedene Keys, die in Tabelle 1 aufgelistet sind.

Tabelle 1: Schlüssel in DNSSEC

Schlüsselname	Schlüssellänge	Gültigkeitsdauer
Key-Signing-Key:	2048 Bit	2-4 Jahre
Zone-Signing-Key:	512 Bit	1-2 Monate

Dabei signiert der durch den eigenen Key-Signing-Key (KSK) signierte, kurzlebige Zone-Signing-Key (ZSK) die DNS-Einträge der eigenen Zone. Der KSK signiert, neben dem eigenen ZSK, KSKs der darunter liegenden Zonen und ist wiederum von der übergeordneten Zone signiert, beziehungsweise bei der Root-Zone von sieben Personen, die nach einem Schema zur Geheimnisteilung zusammen den Schlüssel besitzen. Die Indirektion von KSK und ZSK bringt für den kürzeren ZSK einen verminderten Rechenaufwand zur Überprüfung und weniger zu übertragende Daten bei DNSSEC-Anfragen. Da kurze Schlüssel weniger sicher sind, wird für den langlebigeren KSK eine höhere Schlüssellänge gewählt. Zusätzlich wird durch die Aufteilung offline Signing ermöglicht. Der mächtigere Schlüssel kann in einem sicheren Bereich aufbewahrt werden.

Da offline Signaturen auch Nachteile mit sich bringen, wird ihre Verwendung an DNSSEC auch kritisiert [7]. Offline Signaturen, wie sie in DNSSEC verwendet werden, seien nicht in der Lage dynamisch Links zu erzeugen, müssten zwischengespeichert werden und seien auf Grund der vergleichsweise langen Lebensdauer anfällig für Angriffe wie beispielsweise das Finden einer zu einem Hash passenden Signatur. Während in DNSSEC sich sehr frequent ändernde Links, auf Grund der ständig nötigen Neuzertifizierung, tatsächlich Probleme bereiten, seien laut Dan Kaminsky die anderen Punkte nicht haltbar [13], da mit DNSSEC auch online Signing möglich ist. So sei zum Beispiel on demand online Signing, also das sofortige automatische Absichern einer Domain, durchaus möglich für DNSSEC (RFC4470, RFC4471), wie es das auch für alle gängigen Protokolle und sogar DNSCurve sei.

Resolver können sich bereits verifizierte ZSKs cachen, damit diese nicht bei jeder Anfrage überprüft werden müssen. Beim Cachen kommt neben der alten Time to Live (TTL) nun noch ein neues Verfallsdatum hinzu. Während die TTL beim normalen DNS vom Cachezeitpunkt abhängt, muss diese neue Zeit ein absoluter, vom Zertifikat beziehungsweise Schlüssel abhängiger Wert sein, der gecachete

doi: 10.2313/NET-2011-07-2_08

Daten mit abgelaufenen Zertifizierungen entfernt. Im Resolver kann als Vertrauensanker also entweder ein KSK oder ein ZSK gewählt werden. Ersteres authentisiert damit nur die DNS-Antworten dieses Servers, während Zweiteres die ganze Zone authentisiert. Ist dem Resolver nur der Hash eines Schlüssels bekannt, kann der Schlüssel über DNS erhalten werden. Sofern Platz in der DNS Antwort ist, werden die Signaturen einer Zone, die zum Authentisieren der öffentlichen Schlüssel nötig sind, automatisch mitgesendet um die Zahl der Roundtripps klein zu halten.

2.4 DNSSEC als Nachfolger der X.509 PKI

Die X.509 PKI [4] wird zur Absicherung der meisten verschlüsselten Kommunikationen im Web verwendet. Ein Zertifikat wird dabei meist ausschließlich auf Seite des Servers eingesetzt, um den Schlüsselaustauschpartner zu authentisieren. Clients verfügen hingegen nur selten über Zertifikate. Immer wieder kommt es dabei vor, dass eine CA die sich unterhalb eines vertrauten Knotens befindet kompromittiert wird. Dadurch können bösartige Webseiten dem Browser als gültige, gesicherte Webseiten angezeigt werden. So zwang beispielsweise Mitte März eine kompromittierte CA und die fehlerhafte Umsetzung der CRL Prüfung den Browserhersteller Mozilla zum Nachbessern.

In einer durch DNSSEC aufgebauten PKI kann dagegen eine CA, auf Grund der nun existierenden, strukturbedingten Bindung von CAs und deren öffentlichen Schlüsseln an die Domainnamen, lediglich in ihrem Namensbereich zertifizieren. Das dadurch entstehende Konzept der Zertifizierung relativer Namen wurde bereits vor über 30 Jahren in [9] beschrieben. Es wird die Gefahr durch eine kompromittierte Autorität minimiert. Auch das Verwenden von Aliasen ist in einer solchen Zertifizierungsstruktur möglich. Will ein Benutzer also beispielsweise weder dem „.de“ TLD Nameserver noch dem obersten Knoten vertrauen, so kann er, falls diese Domain auch unter einer „.com“ Endung existiert, auch dem „.com“ TLD Nameserver vertrauen und so eine Vertrauenskette zum unter der gewünschten Domain registrierten Server aufbauen.

Zum Erhalten des öffentlichen Schlüssels, beziehungsweise des eigentlichen Zertifikats gibt es verschiedene Ansätze: DNS kann den öffentlichen Schlüssel im DNSKEY RR speichern oder kann lediglich einen Hash davon enthalten. Da X.509 Zertifikate [4] sehr viele, meist überflüssige, Verwaltungsinformationen enthalten, kann es sinnvoll sein ganz auf Zertifikate zu verzichten und nur den öffentlichen Schlüssel zu verwenden. Werden, zum Beispiel durch bereits für X.509 Zertifikate entwickelte Software, X.509 Zertifikate benötigt, können diese über die Website erhalten werden und deren Authentizität über das DNS gesichert werden. Aber auch den im X.509 Zertifikat eingekapselten, öffentlichen Schlüssel kann man im DNS mittels des CERT RRs hinterlegen. Für die Ablage neuer Informationen spielt die Größe eine entscheidende Rolle. DNS verwendet meistens UDP, muss aber bei größeren Datenmengen auf TCP ausweichen, da die Wahrscheinlichkeit, dass die Antwort bei Verwendung von UDP nicht ankommt, mit der Zahl der für eine Nachricht nötigen Pakete steigt. Hierbei sind vor allem die Größen 512 Bytes und 1500 Bytes relevant. Erstere ist die maximale Größe, bei der die Fragmentierung verhindert werden kann, während Zweiteres die Größe ist ab der auf jeden Fall fragmentiert wird. Mit DNSSEC wächst die Auffassung, man könne das DNS auch zur Auslieferung neuer, größerer Daten nutzen. Daher ist es, da es die einzige Möglichkeit ist, vertretbar bei Größen die UDP überfordern würden, auf TCP auszuweichen und den mit TCP verbundenen Overhead in Kauf zu nehmen. Ebenfalls kann durch die Verwendung von TCP die Verstärkungswirkung durch DNS Server eingeschränkt werden, da hier bereits nach dem vergleichsweise wenig Bandbreite benötigendem Handshake die Verbindung abgebrochen wird, sofern Anfragesteller und Antwortziel nicht identisch sind [13].

Als Kritik an einem vollständigen Ersatz der X.509 PKI sei angeführt, dass DNSSEC lediglich Domains, jedoch keine direkten Personen authentisiert. Zusätzlich weist die Infrastruktur der DNS-Anbieter oft keine mit den X.509 Zertifizierungsstellen vergleichbare RA auf.

2.5 Nutzen und Schwachstellen von DNSSEC

DNSSEC ermöglicht es sicherheitsbewussten Nameservern und Resolvern, die Verlässlichkeit von DNS-Anfragen zu validieren und als sicher, unsicher, gefälscht oder unvertraut einzustufen. Unvertraut meint hierbei, dass die Anfrage zwar überprüfbar ist, aber eine Kette zu einem Vertrauensanker nicht existiert. Wie dann beispielsweise mit einer unvertrauten Antwort umgegangen wird, hängt von der jeweiligen Sicherheitspolitik ab. Als Beispiel kann ein Resolver der DNSSEC unterstützt, aber rekursiv hinter einem anderen, nicht DNSSEC unterstützendem Resolver ist, eine Antwort nur als unsicher einstufen. Wenn DNSSEC von allen an einer Anfrage beteiligten Server und vom Client unterstützt wird, kann DNSSEC Spoofing und Poisoning-Angriffe verhindern. DNSSEC kann also als Security-Enabler dienen. Es kann neue Sicherungsmechanismen ermöglichen, stellt jedoch alleine keinen Garant für Sicherheit dar.

Für DNSSEC wird eine globale, hierarchische PKI mit relativen Namen aufgebaut, die auch für andere Szenarien eingesetzt werden kann. Der Vorteil dieser PKI ist, dass DNS in Verbindung mit DNSSEC eine einzige Instanz, die über ein überall genutztes System authentisierte Zertifikate verteilen kann, darstellt. Jeder Nameserver kann hierbei nur Aussagen in seinem Namensbereich machen. Dies löst ein bekanntes Problem der älteren X.509 PKI, bei dem es durch Vertrauen einer CA vorkommen kann, dass weiter unten in der Hierarchie eine weniger vertrauenswürdige CA falsche Zertifikate erstellt und diesen dann vertraut wird.

DNSSEC war ursprünglich angedacht, um die Kommunikation unter Nameservern abzusichern. Der Benutzer sollte ursprünglich nur durch ein Bit erfahren dass die Adresse sicher ist. Dennoch konnten Daten dann am Ende verfälscht werden. Daher war schnell klar, dass auch Endgeräte die Ergebnisse der DNS Anfrage überprüfen können sollten. Allerdings findet DNSSEC zur Zeit vor allem auf vielen Endgeräten und Routern noch nicht die für eine lückenlose Kette nötige Verbreitung. Daher liefert DNSSEC oft noch keine Ende zu Ende Sicherheit. DNSSEC sieht aber dennoch vor, dass auch Endbenutzer DNS Informationen überprüfen können.

DNS wurde ursprünglich in der Annahme entwickelt, jedem Anfragenden auf eine identische Anfrage die gleiche Ant-

doi: 10.2313/NET-2011-07-2_08

wort zu liefern. DNS-Daten sind also für jeden Anfragenden sichtbar, weswegen DNSSEC keine Vertraulichkeit, Zugriffsschutz oder differenzierte Antworten für unterschiedliche Anfragende schafft. DNSSEC bietet keine Schutzmaßnahmen gegen DoS-Angriffe, wie sie bereits bei DNS ohne DNSSEC möglich waren. Durch die Verursachung von Rechenaufwand für kryptographische Operationen entstehen sogar weitere Möglichkeiten für DoS-Angriffe. Kritiker warnen, dass DNSSEC Amplification Angriffe ermögliche, da teilweise mit sehr großen Datenmengen geantwortet wird [7]. Dies ist, wie bereits erwähnt, ein generelles IP- beziehungsweise UDP-Problem. Pakete können einfach gesendet werden ohne dass sie angenommen werden müssen. Auch DNS ermöglicht schon die Wirkung als Verstärker. Laut Dan Kaminsky liefert DNSSEC, gemessen an DNS, etwa die doppelte Verstärkung. Größere Antworten müssen vom DNS per TCP verschickt werden. Die dadurch vergrößerte Anfälligkeit für Amplification Angriffe werde aber laut Dan Kaminsky ohnehin durch effektiveres HTTP Flooding in den Schatten gestellt [13].

Ebenfalls sieht DNSSEC keine Schutzmaßnahmen für Zonenübertragungen oder dynamisches Updates vor. Da DNSSEC, wie beschrieben, die Objekte und nicht die Verbindung schützt, besteht die Möglichkeit, veraltete Informationen, so lange der ZSK gültig ist, wieder einzuspielen. Somit kann ein nicht mehr gültiger Verweis auf eine IP-Adresse, die nicht mehr unter Kontrolle des ehemaligen Besitzers ist, noch verwendet werden. Da diese Adresse nun beispielsweise für das Betreiben einer Phishingwebsite genutzt werden könnte, stellt dies ein Sicherheitsrisiko dar.

3. DNSCURVE

DNSCurve ist im Vergleich zu DNSSEC eine deutlich jüngere Erweiterung des DNS. Es wurde vom Kryptologen Daniel J. Bernstein (DJB) vorgestellt, mit dem Ziel, Schwachstellen von DNSSEC zu beheben. DNSCurve soll im Gegensatz zu DNSSEC auch Vertraulichkeit schaffen. DNSCurve sichert die Verbindungen ab und dadurch nur indirekt die Objekte. DNS-Anfragen werden dabei mit CurveCP gesichert. Der Name bezieht sich auf die Verwendung von ECC auf Curve25519, im Gegensatz zur in DNSSEC verwendeten RSA-Verschlüsselung. ECC bietet den Vorteil, mit deutlich kürzerer Schlüssellänge bereits vergleichbare Sicherheit zu erreichen. Die kürzere Schlüssellänge ermöglicht performantere kryptographische Berechnungen. CurveCP ist ein ebenfalls von DJB entwickelter und vorgestellter Ansatz zur Absicherung des Internetverkehrs. CurveCP verschlüsselt Pakete der Anwendungsschicht und versendet diese per UDP. Es ersetzt TCP und baut eine verlässliche Verbindung innerhalb von CurveCP nach.

Abbildung 6 veranschaulicht den Ablauf einer DNS-Anfrage mit CurveCP. Jeder DNSCurve Teilnehmer besitzt einen öffentlichen und einen privaten Schlüssel. Dadurch können einem DNSCurve-Teilnehmer Pakete verschlüsselt geschickt werden, wenn dem Sender der öffentliche Schlüssel bekannt ist. Ein DNSCurve Client schickt eine Anfrage an einen Nameserver, indem er die DNS-Anfrage mit dem öffentlichen Schlüssel des Nameservers, dem eigenen privaten Schlüssel und einer Nonce zu einer „kryptographischen Box“ verschlüsselt. Anschließend wird die Nonce und der eigene öffentliche Schlüssel dazu gepackt. Dadurch kann der CurveCP-fähige

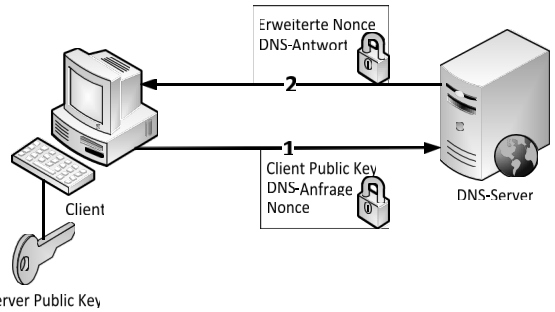

Abbildung 6: CurveCP: Beispiel einer Anfrage

Nameserver über den verwendeten privaten Schlüssel des Senders dessen Identität sicherstellen. Durch die Verwendung des öffentlichen Schlüssels des Nameservers wird sichergestellt, dass nur dieser in der Lage ist die Nachricht zu entschlüsseln. Durch die Kombination von Signatur und Verschlüsselung in einen Ablauf, kann dies mit weniger Rechenaufwand erreicht werden. Eine exakte Beschreibung der zugrunde liegenden Kryptographie und des Aufbaus der „kryptographischen Box“ kann in [6] nachgelesen werden. Dieses Paket wird anschließend per UDP an den Nameserver geschickt. Zunächst wird das Paket vom DNSCurve-fähigen Nameserver als CurveCP Paket behandelt. Es wird entpackt und dabei auf Authentizität und Integrität überprüft. Wenn dies fehlschlägt, wird das Paket wie ein normales DNS-Paket behandelt, damit auch weiterhin normale DNS-Anfragen möglich sind, wenn DNSCurve von einem Server verwendet wird. Wenn die Prüfung erfolgreich war, wird auf das entpackte DNS-Paket geantwortet. Die Antwort wird mit einer erweiterten Nonce und dem selben Schlüssel, der zum Entpacken des CurveCp-Pakets benutzt wurde, wieder in eine „kryptographische Box“ gepackt. Das entstehende Paket wird anstatt des DNS Pakets an den Client geschickt.

Wenn der wartende Client ein nicht gültiges DNSCurve-Paket, ein Paket mit ungültiger Nonce oder eine anderweitig ungültige kryptographische Box erhält, verwirft er diese. Erreicht ihn eine gültige Antwort, wird diese geöffnet und ist damit als original DNS Antwort des Server identifiziert [5].

3.1 Schlüsselmanagement in DNSCurve und CurveCP

Der folgende Abschnitt beschreibt wie der öffentliche Schlüssel eines Nameservers an den Anfrager ausgehändigt wird.

3.1.1 Selbstzertifizierende URLs

Wie beschrieben, hat jeder DNSCurve-Client und Server einen geheimen Schlüssel und den dazugehörigen öffentlichen Schlüssel. Server verteilen ihren öffentlichen Schlüssel, indem sie ihn in den Servernamen, wie er im normalen NS RR steht, einbetten. Hierbei wird die Idee selbstzertifizierender URLs verwendet, wie sie bereits in [12] beschrieben wurden. Die Schlüssel der TLD Nameserver müssen dem Resolver als Vertrauensanker bekannt sein. So müsste beispielsweise zur Einführung von DNSCurve für `www.tum.de` lediglich folgendes RR an den .de Nameserver gemeldet werden: `www.tum.de CNAME 1238675309.tum.de`. 123 ist in diesem verkürzten Beispiel ein Schlüsselwort zur Erkennung,

doi: 10.2313/NET-2011-07-2_08

dass es sich um einen öffentlichen Schlüssel für DNSCurve handelt. Der Client kann daher den Schlüssel 456789 extrahieren. Die Verwendung von selbst zertifizierenden URLs ermöglicht es bestimmten Anwendungen selbst zu definieren, welchen Schlüssel sie verwenden. Ihren eigenen öffentlichen Schlüssel fügen Clients dagegen einfach in ihre Anfragen ein, um sie für die Antwort dem Server mitzuteilen.

3.1.2 Kritik am Schlüsselmanagement in DNSCurve

Dan Kaminsky kritisiert Bernsteins DNSCurve, da es nicht möglich ist sichere, dezentrale und menschenlesbare URLs zu haben [13]. Bernstein benutzt URLs wie beschrieben als Vertrauensanker, präsentiert aber eine Möglichkeit, diese lesbar zu machen und dadurch durch Benutzer handhabbar. Das von Bernstein vorgestellte System ist als problematisch anzusehen, wenn sich IP-Adressen ändern. Kaminsky befürchtet, dass, falls sich Keys ändern, das Internet bald voll sein wird mit nicht funktionierenden Links. Bernsteins Lösung, hierbei mit einem noch nicht genauer spezifizierten P2P DNS Abhilfe zu schaffen, würde dann Sicherheit ausschließen. Ohne eine ausgearbeitete Lösung für das Keymanagement sei DNSCurve jedoch nicht voll funktionstüchtig. Weiter kritisiert Kaminsky die Tatsache, dass DNSCurve nur online Nachrichten unterzeichnen kann, während DNSSEC sowohl online als auch offline Signieren ermöglicht [13]. Der Vorteil von offline Signaturen ist, dass sie erlauben, geheime Schlüssel an einem sichereren Platz aufzubewahren. Beim online Unterzeichnen müssten beispielsweise alle Server einer TLD den privaten Schlüssel haben, was ein Sicherheitsrisiko darstellt.

3.2 DNSCurve als Nachfolger der X.509 PKI

Auch DNSCurve soll, sofern es eingesetzt wird, zur Absicherung der gängigen für X.509 Zertifikate aufgebauten PKI dienen. Eine klassische PKI kann mit DNSCurve jedoch nicht aufgebaut werden, da DNSCurve lediglich Verbindungen absichert. Wenn alle Verbindungen abgesichert und der Client entweder iterativ alle Daten selbst abholt oder allen rekursiv arbeitenden Nameserver vertraut, ist jedoch ein sicherer Kanal mit dem authentisierten Server aufgebaut. Auch wenn Zertifikate in diesem Fall dann auf Grund des sicheren Kanals meist nicht mehr nötig sind, können diese vom Server einfach über den sicheren Kanal geschickt werden. Die Existenz einer ganzen PKI ist in diesem Fall nicht mehr nötig. Jeder Server kann sich sein Zertifikat selbst erstellen und über den bereits existierenden sicheren Kanal selbst verteilen.

3.3 Nutzen und Schwachstellen von DNSCurve

DNSCurve bietet im Gegensatz zu DNSSEC nicht nur Integrität und Authentizität, sondern durch Verschlüsselung auch Vertraulichkeit. Es nutzt ECC und ist dadurch deutlich performanter. DNSCurve ist eine minimalistische Lösung und daher einfach in bestehende Systeme zu implementieren. Es muss nur ein DNSCurve-Modul auf dem Server beziehungsweise dem Client installiert werden. Manipulierte Pakete werden sehr effizient verworfen, was gut gegen DoS-Angriffe ist.

Als problematisch kann der erhöhte kryptographische Aufwand für die Server angesehen werden, da die Ver- und Entschlüsselungen, im Gegensatz zu DNSSEC, bei jeder Verbindung durchgeführt werden muss. Misst man DNSCurve an dem Ziel, Vertraulichkeit zu schaffen, wird dieses nicht voll erreicht, da Sender, Empfänger und die Paketlänge sichtbar bleiben.

DNS-Caches müssen erweitert werden um DNSCurve zu unterstützen. Auch die Caches müssen Ver- und Entschlüsselungen für jede Anfrage durchführen. Hierfür wird für effizienteres Weiterverarbeiten ein geheimer Schlüssel sc zwischen Caches c und Servern s eingesetzt. Ein Cache wird jedes Mal, wenn er einen neuen öffentlichen Schlüssel s eines Servers erhält, daraus einen neuen geheimen Schlüssel sc berechnen.

Nach herkömmlicher Art des Cachings im DNS kann DNSCurve nur Sessions cachen, während mit DNSSEC Records gecached werden. Cachen pro Session benötigt aber eine bedeutend größere Anzahl an Cacheinträgen, da die Anzahl der Sessions hoch ist und die Anfragelast pro Session eine deutlich niedrigere Varianz aufweist als die gesamte Anfragelast. Neben dem geringeren Nutzen eines solchen Vorgehens, führt dies zu einem deutlich erhöhten Aufwand für Caches.

Auch werden Firewallprobleme mit CurveCP durch das Tunneln von größeren Datenmengen über DNS-Port 53, ohne DNS zu emulieren, befürchtet. Daher wäre es Sinnvoll einen anderen Port zu verwenden [13].

4. PERFORMANZVERGLEICH

Der folgende Abschnitt behandelt die Performanzdebatte über DNSSEC und DNSCurve zwischen Dan Kaminsky [13] und Daniel J. Bernstein [7]. Dan Kaminsky ist ein Spezialist für Computersicherheit und Geschäftsführer des Penetration-Testing-Unternehmens IOActive. Er arbeitete unter anderem für Cisco, Avaya und Microsoft. Seit 2010 ist er, als einer der sieben Schlüsselträger des DNS-Rootkeys, der amerikanische Repräsentant. Daniel Julius Bernstein ist Professor an der University of Illinois in Chicago. Er ist unter anderem Autor der DNS Serversoftware djbdns und entwickelt zurzeit DNSCurve und CurveCP, um damit das aufkommende, seiner Meinung nach nicht ausreichende, DNSSEC und die bestehenden Ansätze wie HTTPS zu ersetzen.

Wie in den vorangegangenen Abschnitten beschrieben, versuchen sowohl DNSSEC als auch DNSCurve durch sichere Authentisierung, über DNS eine Grundlage für sichere Kommunikation im Netz zu legen. DNSSEC ermöglicht einen Vertrauensanker für TLS oder IP-Sec, während DNSCurve mit der dazugehörigen Lösung CurveCP funktionieren soll. DNSCurve und CurveCP gleicht dabei noch mehr einem Entwurf, während DNSSEC bereits auf eine immer größer werdende, installierte Basis zurückgreifen kann.

4.0.1 Einsatz von Eliptic Curve Kryptographie

In der Analyse von DNSCurve kann Kaminsky den von Bernstein festgestellten Performanzvorteil von Curve25519 bestätigen [13]. Curve25519 ist etwa um Faktor 4 bis 8 schneller als RSA1024. Jedoch sei zu bedenken, dass RSA besser unterstützt ist und auf modernen Rechnern daher dank besserer Hardware einer durch Software berechneten Curve25519 überlegen sein wird. Kaminsky nimmt das bei gleicher Schlüssellänge niedrigere Level an Sicherheit von RSA

doi: 10.2313/NET-2011-07-2_08

dabei in Kauf. Als weitere Kritik am Einsatz von Curve25519 in DNSCurve muss angesehen werden, dass ein Beweis gegenüber einer dritten Person nicht wie mit mit RSA signierten Nachrichten in DNSSEC möglich ist. Das führt dazu, dass es in DNSCurve keinen Mechanismus gibt, eine ganze Vertrauenskette in einer einzigen Anfrage zu überprüfen. Der Client muss daher zu jedem Beteiligten eine Verbindung aufbauen.

4.0.2 Caching abgesicherter DNS-Pakete

Als größte Schwäche von DNSCurve nennt Kaminsky die Unmöglichkeit, DNSCurve-Nachrichten für andere Benutzer zu cachen [13]. Dies würde, im Gegensatz zu DNSSEC, den ISP Cache komplett aushebeln. Kaminsky schätzt hierbei eine deutlich erhöhte Belastung für die autoritativen Nameserver, während Bernstein von einer 1,15 fachen Belastung ausgeht. Dies würde jedoch nur eine Trefferrate pro Cache von gemittelt 6% bedeuten. Providerstatistiken lassen eine Trefferrate der ISP-Caches von etwa 80% -90% vermuten - also 5x-10x Belastung [13]. Bernstein argumentiert, dass lokales Caching dennoch möglich sei und Cachen in zweiter Ebene irrelevant sei. Kaminsky erwidert, dass lokales Caching bereits gemacht werde und alleine nicht ausreichen würde. Dabei stützt er sich auf Messwerte großer Caches, während Bernstein sich auf ein seiner Meinung nach unrealistisches Laborexperiment beziehe. Schließlich sei Caching zweiter Ebene auch für die Latenz meistens besser als den autoritativen Nameserver anzufragen.

4.0.3 CurveCP vs. SSL

Bernstein nennt Performanzprobleme als Grund für die nicht volle Verbreitung von HTTPS und damit als Motivation das performantere CurveCP zu verwenden [7]. Er führt dabei Google an, das nicht alle Daten mittels HTTPS verschlüsselt. CurveCP dagegen sei performanter, da Verschlüsselte Pakete nicht in einem TCP-Stream, sondern per UDP versendet werden. Verlässlichkeit wird durch das Imitieren einer TCP Verbindung innerhalb des CurveCP Protokolls erreicht. Laut Kaminsky, der sich unter Anderem auf Aussagen von Verantwortlichen bei Google stützt, ist Performanz jedoch keiner der Gründe die gegen HTTPS sprechen. HTTPS sei vor allem aufwändig „anzuschalten“, da die meisten Webseiten aus einem Konglomerat diverser Komponenten, die durch Links zusammengehalten werden, bestehen. Dabei ist vor allem der Aufwand zur Absicherung jeder einzelnen Komponente ein großes Hindernis. Hier kann eine Absicherung des DNS einen Vertrauensanker schaffen und, zum Beispiel durch wegfallende Arbeit mit einer PKI Struktur, den Einstiegsaufwand senken.

5. FAZIT

DNSSEC bietet die Möglichkeit, die Authentizität und Integrität von allen Informationen im DNS zu sichern. Neben den im DNS enthaltenen IP-Adressen kommen öffentliche Schlüssel, oder zumindest Hashes davon, hinzu. Dies motiviert dazu, das überall verfügbare und, zum Beispiel durch das TXT Record, sehr flexible DNS für weitere Einsatzszenarien als dem bloßen Ablegen von IP-Adressen zu verwenden. Zusätzlich können dank DNSSEC Schwachstellen bisheriger X.509 PKIs überwunden werden. DNSSEC kann den Verbindungsaufbau für IP-Sec oder TLS erheblich vereinfachen, absichern und auch beschleunigen. Auch DNSCurve mit CurveCP verfolgt das Ziel, das komplette Internet abzusichern. Es soll hierbei zusätzlich zur Authentizität auch Vertraulichkeit geschaffen werden, ohne dabei auf weitere, bereits existierende Protokolle angewiesen zu sein. Beide Ansätze besitzen Probleme mit gerade in letzter Zeit aufkommenden Vorschlägen für ein P2P DNS umzugehen.

DNSSEC, mit seiner langen Entwicklungsgeschichte, erfreut sich immer größerer Verbreitung. Der sehr einfach konzipierte Ansatz von DNSCurve stellt dagegen eine Alternative zur Verfügung, die allerdings noch diverse Tests bezüglich Einsetzbarkeit und Klärung einiger ungelöster Probleme benötigt. Anschließend müsste DNSCurve den Vorsprung von DNSSEC aufholen. Welche Entwicklungen am Ende von den diversen Stakeholdern angenommen werden, bleibt abzuwarten.

6. LITERATUR

[1] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose: *DNS Security Introduction and Requirements*, RFC 4033, März 2005

[2] S. M. Bellovin: *Using the Domain Name System for System Break-ins*, In Proceedings of the Fifth Usenix Unix Security Symposium, Seite 199-208, Salt Lake City, USA, Juni 1995

[3] D. Blacka: *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*, RFC 5155, Februar 2008

[4] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 5280, Mai 2008

[5] M. Dempsky: *DNSCurve: Link-Level Security for the Domain Name System*, Internet-Draft, Februar 2010

[6] D. J. Bernstein: *Cryptography in NaCl*, Networking and Cryptography library, 2009

[7] D. J. Bernstein: *High-speed high-security cryptography: encrypting and authenticating the whole Internet*, 27C3, Berlin, Deutschland, 2011

[8] D. Eastlake: *Domain Name System Security Extensions*, RFC 2535, März 1999

[9] M. Gasser, A. Goldstein, C. Kaufman, B. Lampson: *The digital distributed systems security architecture*, Proc. of the 12th National Computer Security Conference, Seite 305-319, NIST, Gaithersburg, USA, 1989

[10] P. Hoffman: *Elliptic Curve DSA for DNSSEC*, Internet-Draft, Dezember 2010

[11] M. Lottor: *Domain Administrators Operations Guide*, RFC 1033, November 1987

[12] M. Kaminsky, E. Banks: *SFS-HTTP: Securing the Web with Self-Certifying URLs.*, MIT Laboratory for Computer Science, Cambridge, USA, 1999

[13] D. Kaminsky's Blogg `http://dankaminsky.com`

[14] P. Mockapetris: *Domain Names - Concepts and Facilities*, RFC 1034, November 1987

[15] P. Mockapetris: *Domain Names - Implementation and Specification*, RFC 1035, November 1987

[16] A. S. Tannenbaum: *Computer Networks*, vierte Auflage, Prentice Hall, New Jersey, USA, 2003