2012

Providing public key certificate authorization and policy with DNS

Matthew Lidestri

Providing Public Key Certificate Authorization and Policy with DNS

By

Matthew Lidestri

Thesis submitted in partial fulfillment of the requirements
for the degree of
Master of Science in
Computing Security and Information Assurance

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

February 28, 2012

Committee:
Charles Border (Chair)
Daryl Johnson
Yin Pan
Bill Stackpole

Abstract

Public Key Infrastructure (PKI) instills trust in certificates commonly used to secure email, web traffic, VPNs, file transfers, and other forms of network communication. Due to a number of successful attacks against certificate authorities, malicious parties have illegitimately acquired trusted certificates for widely used online services, government agencies, and other important organizations. These incidents, and the potential for future attacks of a similar nature, present notable risk to PKI and global security as a whole.

The proposed Certificate Policy Framework (CPF) offers a mechanism for organizations to control which certificates are authorized to authenticate their services. This DNS-based protocol allows organizations to publish an access control list for any given hostname, where each entry in the ACL identifies a certificate and indicates whether the certificate should be blocked, warned upon, or permitted. Similarly, any CPF-compatible application can query DNS for CPF records to verify the integrity of the certificate from an authoritative viewpoint. In this work, we review limitations in PKI and certificate-based security and review existing work in this area. We will also discuss CPF in greater detail and demonstrate how it can be used to augment PKI to strengthen this widely adopted technology.

Table of Contents

Introduction

We continue to enjoy the benefits of modern networks, which provide a medium for the fast and reliable exchange of information. In conditions where security is desired or essential, we rely upon encryption to ensure the confidentiality and integrity of data while in transit. In many cases, public key cryptography is used to establish a secure connection over unsecured networks. This crypto-system utilizes two unique, mathematically related keys – a private key and a public key. The public key is often bound with identification information to form a certificate for authentication. However, in order to use a certificate in this manner, we need to trust that the identity information is accurate – that is, that the peer is who it claims to be. Without establishing trust, data could be unintentionally shared with an unauthorized party impersonating the intended peer.

Public Key Infrastructure (PKI) has been widely implemented as a foundation for instilling trust in certificates [19]. PKI is a centralized trust model comprised of one or more certificate authorities (CAs) run by trusted organizations, third parties, and governments. Operating systems and third party applications support a number of CA certificates by default, thereby trusting them and any certificates that they sign (endorse). Reputable certificate authorities abide by a set of policies for verifying the information associated with a certificate prior to signing it; in some cases these checks are initiated by a registration authority (RA) before being passed to the CA for signing. These verification checks are in place to ensure that certificates will only be issued to authorized requestors, providing credibility to the system.

While PKI has proven to be a scalable and manageable trust model, security professionals have expressed concern over weaknesses [16][13]. In the past, highly regarded CAs have issued certificates representing specific services to individuals having no affiliation with the domain owners [9]. In addition to the risks associated with validation failures, computing resources maintained by certificate authorities have also been compromised in the past. As a result of these breaches, attackers were able to fraudulently acquire trusted certificates for numerous services. These certificates could be used to intercept traffic destined for the target services via a man-in-the-middle (MITM) attack, enabling the malicious party to view and manipulate data being exchanged.

These issues demonstrate a major problem with PKI – certificate authorities are trusted unconditionally [10]. The organizations whose online services have been negatively impacted by breaches at CAs had no way of preventing the certificates from being generated or trusted. Each client application dictates which CAs are considered trustworthy, and will inherently accept any certificate issued by a trusted CA. Due to this broad level of trust, the issuance of an illegitimate certificate will allow a malicious party to falsely impersonate a service or organization without warning.

In addition to the flaws with PKI, there are issues with certificate-based security. Application warnings caused by certificate validity errors are often simple for users to bypass, providing little protection. Also, if a service is compromised and the public/private key pair is exposed, the attacker could launch a transparent MITM attack. These situations further challenge the ability for PKI and certificates to properly secure traffic.

In order for PKI to be a viable trust model in the future, organizations must have the ability to manage which certificates are authorized to identify its services. Additionally, client applications must be able to access this information and enforce it. We believe that this can be accomplished using a protocol designed to distribute certificate authorization rules and policy. This work presents a new protocol called certificate policy framework (CPF), which allows organizations to advertise authorized certificate signatures for a given hostname via the domain name service (DNS). In addition, it will allow organizations to provide client applications with guidance on whether to permit, warn, or block connections based on the certificate that it receives. The enhancements introduced by CPF will complement the trust offered by certificate authorities, reducing risk and strengthening PKI.

Background

Public key cryptography is commonly applied to networked services in order to conceal data being exchanged between peers and to prevent the modification of data while in transit. This is commonly achieved using X.509 certificates, a standard format for combining identity information with a unique public key. Certificates are an integral part of encryption protocols like Transport Layer Security (TLS) and its predecessor, Secure Socket Layer (SSL), and are heavily leveraged to tunnel unencrypted protocols like SMTP (email), FTP (file transfers), and HTTP (web browsing). Internet users most commonly interact with certificates while web

browsing, as nearly everyone visits websites and web applications protected by HTTPS. Common examples include online banking sites, stock/401k portfolio management sites, and webmail services. Without TLS/SSL protecting these plaintext protocols, it would be trivial to eavesdrop on the traffic. Given the right access, it would also be possible to intercept the traffic in transit and manipulate it transparently.

Eavesdropping can occur using passive monitoring as data passes through a network. Network taps, port mirroring, and wireless sniffing are common ways to capture data transparently as it traverses network devices and transmission media. Packet capture tools like Wireshark or TCPDump can be used to save and analyze pure network traffic streams, while more security-driven tools like Cain & Able[1] can be configured to sniff traffic and capture specific information like account credentials. When encryption protocols are applied and comply with best practices, the application data is concealed from exposure as cipher-text and cannot be accessed without possessing the appropriate encryption key(s).

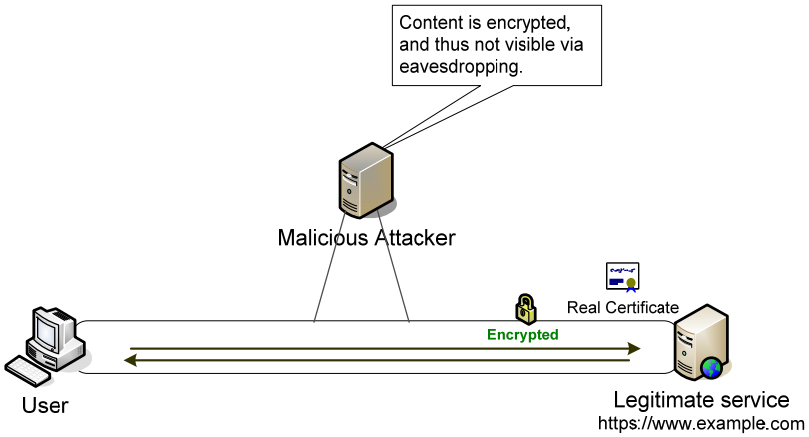


Figure 1: Example of third party eavesdropping on encrypted communications.

A MITM attack is an active effort by a malicious party to intercept network connections by posing as the intended peer on both sides. In order to successfully orchestrate a MITM attack, the network traffic must flow through a system controlled by the attacker. This can be done at OSI layer 2 via an ARP poisoning attack, by receiving routed traffic, or by poisoning DNS to direct traffic to a different IP address. By maintaining separate connections to the client application and server and acting as a pass-through, the attacker gains the ability to both eavesdrop on communications and manipulate the data in-transit. Encrypted protocols such as TLS/SSL and

[1] http://www.oxid.it/cain.html

SSH are also susceptible to MITM attacks, although it's extremely difficult to intercept this traffic undetected.

SSH uses a host key by default, which is the public key. The first time an SSH client like Putty[2] connects to an SSH server, it stores the server's host key locally as a reference for future connections. If the client ever connects to the same resource and is presented with a different host key, it will suspect a MITM attack and notify the end-user. This validation technique employs "trust on first use" (TOFU) where the client assumes that the first time they connect to an endpoint it is legitimate [12]. If the connection is being intercepted on first use, then the client will remain unaware that the encrypted connection has indeed been compromised.

TLS-based connections are not generally susceptible to the risks associated with TOFU due to the use of signed certificates and support for PKI. A TLS-enabled service authenticates itself to the client application using an X.509 certificate that has been signed by a highly accepted certificate authority, such as Symantec (VeriSign), Thawte, or Entrust. Client applications often trust these and other CAs by default, and will implicitly trust any certificate that a trusted CA has signed. Thus, the client is able to use its pre-shared trust relationship with the CAs to verify that the certificate representing the remote service is truly legitimate.

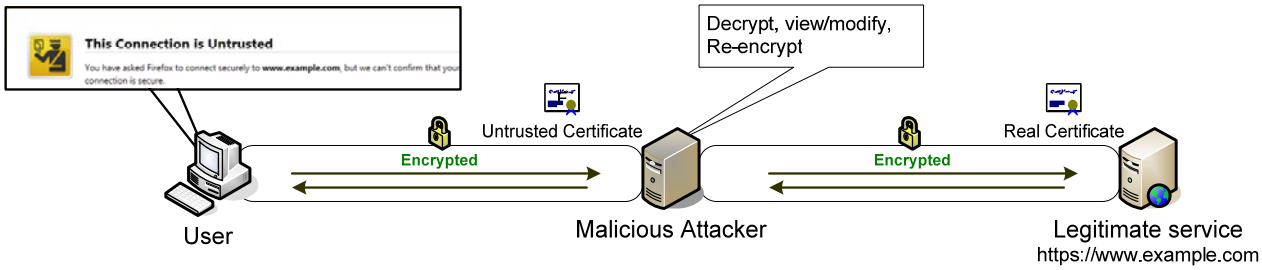


Figure 2: Example of a successful MITM attack on encrypted communications.

If the TLS connection is being proxied in a MITM attack, it's likely that the attacker is using a self-signed certificate or invalid certificate, which would not be trusted by default. This should cause the client application to present an error to the user or refuse the connection.

[2] http://www.chiark.greenend.org.uk/~sgtatham/putty/

Certificates and PKI, paired together, effectively address the limitations of simple public key exchanges and provide strong security. However, there are other challenges with certificate-based security and PKI trust that introduce risk and are cause for concern.

Certificate Trust Errors are Ignored

One of the key purposes of PKI is to prevent a MITM attack from occurring transparently. Certain applications, such as the Oracle Java Runtime Environment will fail when connecting to an untrusted peer by default. Other applications will warn the user and require confirmation to proceed. Although client applications may vary in their response to an untrusted certificate, they should universally prevent a connection from occurring in a completely trusted context under conditions that indicate otherwise.

Web browsers are among the most common applications that prompt the user to continue if a certificate's trust cannot be determined. Traditionally, browsers will generate warnings when presented with a peer certificate that has not been signed by a trusted CA, has been revoked, has expired and is no longer considered valid, or where the certificate common name does not match the hostname provided in the URL. While the majority of certificates presented to users are valid and trusted, there are situations where untrusted certificates are used to protect services and applications on intranets, development networks, and Internet-facing devices for the sole benefit of encrypting traffic. In these cases, the end-user must decide whether or not to trust the connection. Certificate trust is a complicated matter, and studies indicate that a high percentage of users don't fully understand certificate warnings or the risks associated with trusting a questionable certificate [18]. When a user decides to trust an unknown certificate and continue with the connection, they may be unintentionally passing their data to a malicious endpoint.

Certificate Validation Vulnerabilities

As mentioned previously, public key certificates are the combination of a public encryption key and fields containing identification information. Fields of interest often include (but are not limited to) the "subject name", "validity period", "issuer name", "basic constraints", and "key usage" [7]. In order for a certificate to act as an authenticator, applications must be able to interpret this information and enforce security accordingly. However, there have been logic flaws

in various applications that prevented them from properly applying certificate security, putting their users at risk.

In 2009, a null prefix attack was publicly disclosed and impacted a number of applications, such as popular web browsers, instant messaging applications, and email clients [11]. Due to a validation bug when interpreting the subject name of a certificate, the inclusion of a "\0" character would cause the application to treat this string as the end of the field content. For instance, if a person owned the domain "example.com" and purchased a certificate for "www.rit.edu\0.example.com", a CA would legitimately issue an X.509 certificate as a subdomain for "example.com", but vulnerable applications would interpret the common name as www.rit.edu.

In 2011, a similar vulnerability was disclosed affecting various devices produced by Apple Inc. [8][14]. Apple's data security component did not properly validate the "basic constraints" field of all certificates in the certificate chain. Due to this bug, a certificate specifically designated for server authentication could be used to sign another certificate just like a CA. When working properly, software should check to ensure that the certificate used to sign another is a CA – if it is not, then the certificate should not be trusted.

These types of application vulnerabilities provide attackers with an opportunity to circumvent certificate security and intercept encrypted traffic in a completely trusted context. While code flaws are unavoidable in applications, a defense-in-depth approach can significantly reduce the impact of these deficiencies. Rather than relying solely on certificate validation, it would be beneficial to have another mechanism in place to verify the legitimacy of the certificate.

PKI Reliability Issues

Certificate authorities and registration authorities are the catalysts for trust in PKI. They are the bodies that are responsible for investigating the legitimacy of certificates and endorsing them. By having a pre-shared trust relationship with specific CAs, the integrity of any given server certificate can be verified easily by client applications. While CAs are an essential component for trust in the digital world, vulnerabilities have been discovered that introduce major security implications when successfully exploited.

In the past, we have observed successful attacks against CAs by way of weaknesses in domain validation processes. On December 29th, 2008 a registration authority associated with the Comodo CA mistakenly issued a certificate for mozilla.com to an unauthorized researcher [9]. Comodo confirmed that this issue resulted from a failure in processes and had taken steps to remediate the issue. However, this particular incident is important because the technical controls proved to be ineffective, as opposed to being maliciously circumvented.

More recently, we have witnessed sophisticated attacks against CAs by way of bypassing controls to fraudulently acquire certificates for important services and organizations. In March of 2011 an attacker was able to acquire valid certificates for popular online services managed by seven different companies including Google, Microsoft, the Mozilla Foundation, and Skype via a registration authority associated with Comodo Group. The fraudulent certificates were quickly detected and revoked by Comodo, and browser vendors deployed application updates to blacklist the malicious certificates. The login credentials for the registration authority had been compromised, granting attackers access to generate certificates [22].

In July of 2011, StartCom's StartSSL CA suffered a similar security attack that led them to take their online services offline for a short period of time. StartCom confirmed that no fraudulent certificates were issued as a result of the attack and their CA private keys were not exposed. [4] While no specific details were posted about the attack, it raises concerns as attackers continue to target PKI vendors.

In September of 2011, Dutch certificate vendor DigiNotar B.V. disclosed a major security breach affecting its systems. As a result, attackers were able to generate 344 fraudulent certificates successfully targeting domains for a variety of services. Impacted organizations included Google, Microsoft, Facebook, Twitter, Yahoo, the United States Central Intelligence Agency (CIA), and other major certificate authorities. An independent investigation indicates that the attackers had full administrative control over numerous certificate servers operated by DigiNotar, and were believed to be active for well over two months [15]. Computer forensics uncovered several malicious applications resident on DigiNotar's servers, developed specifically for use on their systems. Several weaknesses were also discovered in DigiNotar's "secure" computing environment; the domain administrator password was identified as weak, facilitating compromise on a larger scale. In addition, no antivirus or anti-malware solutions were installed, applications were not patched, and critical services were not separated in accordance with best practices.

These incidents raise major concerns regarding the security provided by certificate authorities. As of July 2010, there are 1,482 certificate authorities run by 651 organizations worldwide that are trusted by popular web browsers [3]. While this provides a wide degree of competition and flexibility when choosing a PKI vendor, client applications are placing full trust in an extraordinary number of organizations. Most certificate authorities are independently managed and differ in policies and security posture. In the case of the DigiNotar breach, the company's PKI infrastructure lacked the most basic and essential security controls. With hundreds of organizations running trusted certificate authorities, we must consider the possibility that other PKI environments may not adhere to industry security standards and best practices.

With an increasing trend of attacks against certificate authorities, they may be exposed to greater risk than in the past. PKI services are of great interest to parties that would benefit by being able to intercept secure communications and possess the means to do so, such as state-sponsored attackers. By gaining access to generate trusted certificates on-demand, attackers also gain the ability to intercept secure communications on a wider scale undetected. In addition, the previously discussed weaknesses in existing PKI services indicate that this infrastructure may be more vulnerable than originally perceived. Due to the design of the PKI trust model, if any given CA issues a fraudulent certificate then it will impact all applications that trust the CA without confinement. In other words – the compromise of a certificate authority yields a high reward to the attacker, and nearly the same results can be achieved regardless of which CA is targeted. Essentially, PKI is only as strong as the weakest certificate authority or registration authority. These issues pose considerable risk to PKI and limit its reliability as a trust model.

Background Conclusion

A malicious attacker can overcome the protection provided by PKI and certificate-based security using a variety of methods that range in impact and scope, as shown in table 1. The most damaging attacks allow traffic to be intercepted in a trusted context, such that the end-user or system is not aware of the compromised connection and continues to exchange data that may be sensitive in nature.

Threat	Impacted Audience	Scope	Context
Certificate Errors Ignored by User/App	Specific peers (users)	Limited	Untrusted
Certificate Validation Vulnerabilities in Apps	Specific applications	Limited	Trusted
Improper Certificate Issuance by CA	Specific service	Limited	Trusted
Compromised CA	Multiple services	Broad	Trusted

Table 1: Summary of threats that negatively impact the security provided by PKI.

All of these issues pose a threat to users, organizations, and systems. Fortunately this area continues to attract more attention by the academic and security communities, and previous work has provided potential solutions to address one or more of these threats.

Existing Work

As previously discussed, one of the challenges with public key certificates is that trust warnings are not easily understood by users. The work of Sunshine et al [18] found that a significant percentage of users do not understand the risks associated with SSL certificate warnings in browsers, and often opt to ignore the potential danger. This behavior was most common with warnings that are cryptic and easily disregarded (Firefox 2.x) or capable of being bypassed with little effort (IE7). The Firefox 3 web browser introduced an SSL warning that requires multiple steps to bypass, further improving user behavior. Although the design of the certificate warning impacts its effectiveness, the authors of this study ultimately concluded "the best avenue we have for keeping users safe may be to avoid SSL warnings altogether and really make decisions for the users – blocking them from unsafe situations and remaining silent in safe situations". We agree with this assertion – it would be ideal to prevent users from accessing services protected by certificates or public keys that pose a significant risk.

In order for applications to proactively protect users, they need to support alternative mechanisms beyond PKI for assessing the risk associated with a certificate. Two works have examined the concept of verifying the legitimacy of public keys by comparing them to other vantage points on the Internet. Stone-Gross et all propose the VeriKey solution [17] to detect MITM attacks against websites that use self-signed certificates. Wendlandt, Anderson, and Perrig propose a very similar solution, Perspectives [20], to address the "trust on first use" issue with public keys to prevent MITM attacks. These solutions work on the fundamental premise that all client devices should be presented with the same public key when connecting to a service, regardless of

geographic location or Internet Service Provider (ISP). If a particular client receives a public key certificate that differs from everywhere else, there is an elevated risk that the traffic is being intercepted. When connecting to a remote service that supports public key cryptography, the client application queries a set of geographically distributed "verification servers" for their view of the service's public key. Upon receiving the hostname, a verification server connects to the target service, receives the public certificate, and returns it to the requesting client for comparison. If the public key presented to the verification server matches the public key seen by the application, then the key is legitimate. However, if they do not match, then it's likely that the traffic is being intercepted.

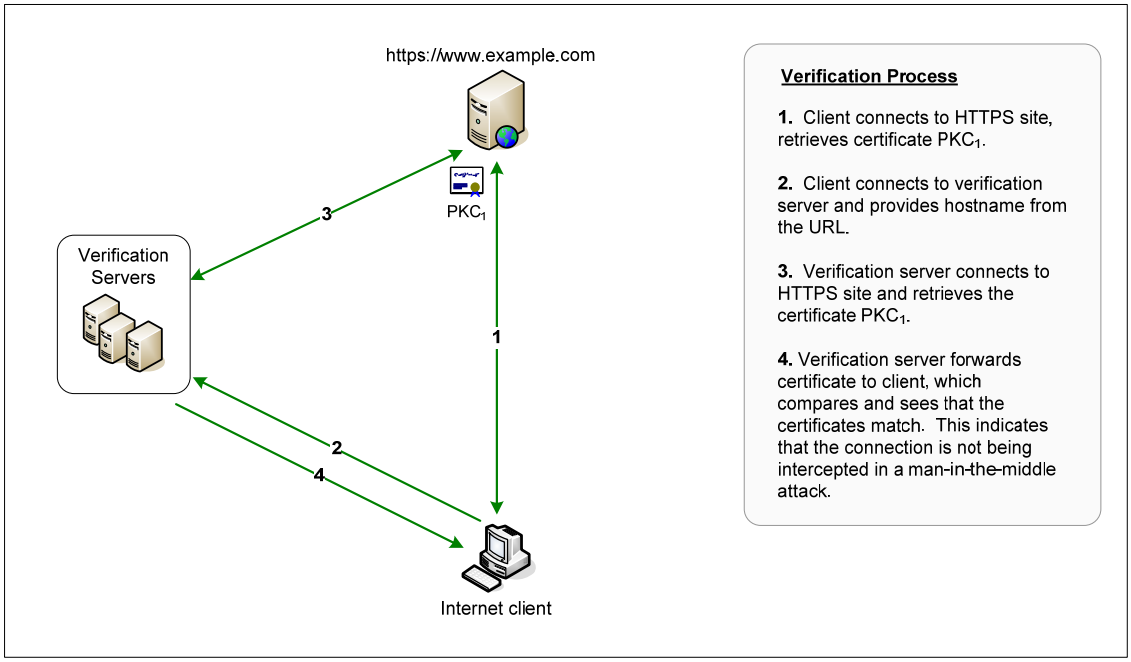


Figure 3: Verification process for legitimate connection using an external public key comparison service.

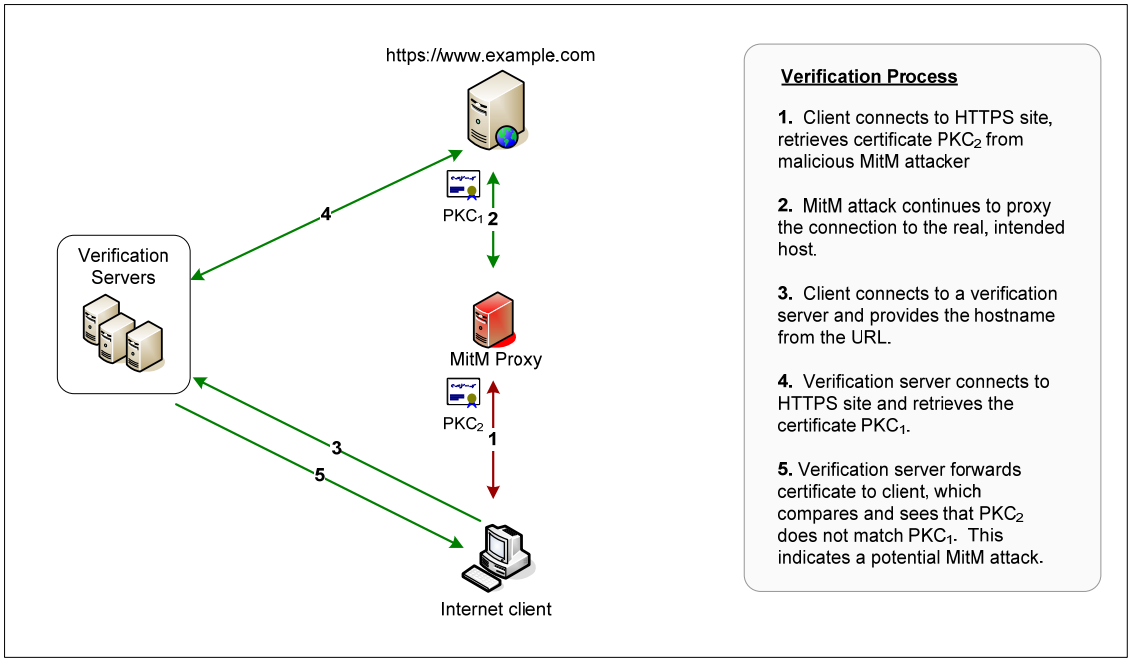


Figure 4: Verification process under MITM attack conditions using an external public key comparison service.

The concepts presented by Perspectives have been adopted and enhanced in Convergence[3], a new project that's available to the general public. While the goal of previous solutions was to improve authenticity for self-signed certificates and public keys, Convergence has been developed with an expanded purpose – to replace certificate authorities as the trust point for services using public key cryptography [12]. If successful, the replacement of PKI would lead to the elimination of CA trust for server certificate authentication. Companies and organizations could simply apply a self-signed certificate to their services in order to enable encryption, and Convergence would provide verification that a client's connection is truly secure. This notion has appeal and highly visible upside, but introduces several concerns.

PKI has existed for decades and is integrated into many different operating environments as a security standard. In addition to global PKI, companies and organizations may run an internal certificate authority as a method of authenticating local services. In order to eliminate PKI, many network applications will need to be updated. Web browsers and SSH clients come to mind immediately, and we would expect these to have the greatest impact. However, many other applications would also need to adopt Convergence - SMTP servers, FTP clients, VPN software, and a variety of third-party applications.

[3] http://www.convergence.io

In certain cases, network visibility will be a hurdle for Convergence. It's straightforward for notary servers to verify the public keys on Internet-facing services so long as they are publicly accessible. However, if a device is not available to the Internet, then the notary servers will not be able to verify the public key. An alternative may be for companies to run an internal notary server, which presents its own issues. A dedicated notary server is another service to maintain, which can be inconvenient for smaller organizations which are limited in funding and technical resources. In addition, notary servers using the preferred Perspectives backend require network connectivity to any devices running secure services in order to verify the public key. This may pose a dilemma for organizations enforcing strict network access controls, since the notary server could be viewed as a central point to circumvent firewall restrictions.

It is also noteworthy that Convergence focuses its verification process on public keys, which are the unique id and true identifier. However, X.509 certificates also include field information that's currently validated by certificate authorities (albeit the degree of review varies between CAs and their certificate products). It is useful for certificates to include information like the "organization name", providing it is accurate. By ignoring these fields, it greatly reduces the ability for X.509 certificates to act as a true certificate of identity.

Rather than replacing certificate authorities, other efforts have explored opportunities to strengthen PKI. Hallam-Baker, Stradling, and Laurie propose allocating a Certificate Authority Authorization (CAA) resource record in DNS; this would allow domain owners to publish the object identifier (OID) of approved certificate authorities for their domains [5]. In turn, whenever a CA receives a certificate signing request, a CAA record check would be required as part of its verification procedures. If the record exists but the CA's OID is not listed, then the request should be rejected due to insufficient authorization. In effect, this will allow organizations to indirectly control which CAs can sign certificates for its services. It also imposes a new layer of accountability on CAs; if a certificate is signed illegitimately even though a CAA record is present, then this would indicate bigger issues with the offending CA's security controls and may warrant global revocation of trust.

A relatively new IETF working group, DNS-based Authentication of Named Entities (DANE), has also begun working on approaches to provide greater trust in networked communications [2]. The purpose of this group is to provide DNS administrators with the ability to bind identification

information with specific services securely using DNSSEC, for the purposes of distribution and authentication. They are currently overseeing the development of a protocol for linking certificates and certificate associations with domain names for use in TLS connections [6]. The initial specifications allow certificates, public keys, and CAs to be linked to a service via DNS for trust. The details of this protocol are still under development, but show promise and are bound to be a significant contributor in this area. Conceptually, it shares similarities with the CPF protocol, although it is more geared towards the publication of trust points rather than distributing detailed policies. As a result, the protocol can dictate what is considered trusted but cannot offer any greater granularity or guidance to a client application.

Overview of CPF

The recent attacks against certificate authorities are a clear indication that we cannot continue to rely upon these entities as the sole proprietors of online trust. Furthermore, application-specific X.509 certificate validation vulnerabilities, certificate warning ineffectiveness, and similar issues have contributed to the reduced effectiveness of certificate-based security. Foreseeing this as an opportunity to develop a more comprehensive solution, Certificate Policy Framework (CPF) is intended to address a number of these limitations.

The CPF protocol defines a lightweight means of publishing, retrieving, and interpreting certificate policies for a network service. CPF policy is comprised of a set of authorization rules for certificates and public keys. Each policy entry defines a certificate based on its hash digest and indicates how client applications should react if matched. The policies are published in DNS as text-based resource records, and are specific to each fully-qualified hostname as it appears in DNS and in either the certificate common name or subject alternative name.

CPF empowers organizations with the ability to control the authorization of certificates representing its services. This approach establishes dual-control shared between the CAs and authoritative organizations, effectively limiting trust in CAs. The use of PKI and CPF together promote greater security than either can provide independently. CAs will still be responsible for verifying the legitimacy of a certificate signing request and then signing the certificate, thereby providing a level of assurance from a third party. CPF offers a certificate verification mechanism from the perspective of the authoritative entity, which is inherently the most knowledgeable party and is ultimately responsible for the security of its services.

CPF Design

The foundation of CPF is derived from Sender Policy Framework (SPF), a widely implemented DNS-based protocol that allows domain owners to publish a list of mail servers that may send email on behalf of a given domain [21]. The use cases for SPF and CPF are very similar; a given $Peer_2$ receives information from $Peer_1$, but $Peer_2$ is not confident that $Peer_1$ is trustworthy. In turn, $Peer_2$ contacts the DNS server controlled by the organization responsible for the alleged $Peer_1$ to verify legitimacy and authorization (if available). In the case of SPF, an SMTP server is interested in verifying that the client server's IP address is authorized to forward mail on behalf of the sender's domain. Similarly, a CPF-compatible client application is interested in verifying the legitimacy of a peer's public key before exchanging data.

The full specifications for the CPF protocol have been documented as an RFC-style draft included in the appendix of this work. The RFC discusses the CPF protocol and how it should be applied and interpreted in great detail. The full specification was used to architect CPF and has been incorporated in our experiments. This section discusses key aspects of CPF in a detailed albeit less comprehensive manner. We've broken down the overview of CPF into three sections – defining the CPF record format, publishing the policy, and accessing/interpreting policy.

CPF Record Format

CPF records adhere to a standardized format to ensure that CPF-compliant client applications can parse the information properly and consistently. The record is a single, case-insensitive string of text that is processed from left to right with fields delimited by spaces. The first field in the record indicates the protocol version, and is followed by one or more directives. Consider the example below:

```
v=1 hash_sha1:938ca8e9a284355ce1a7ff7621c1d2d876ab2543 ~all
\_/ \_____________________________________________/ \___/
 |                       |                           |
Version      Directive 1 – hash mechanism     Directive 2 – "all"
             with default qualifier of "+"    mechanism with "~"
                                                 qualifier
```

Each directive is a single rule comprised of both a mechanism and a qualifier. Each mechanism is an identifier for a certificate, or for a supported DNS resource record that may identify a

certificate. If matched, the client application should enforce the policy much like an access control entry (ACE) in a firewall. Supported mechanism types are as follows:

hash: Defines a hash of a target certificate or public key. The directive is officially denoted as *hash_<algorithm>*, followed by a ":" and the full hash value. Supported hash algorithms include SHA1, SHA256, and SHA512.

include: Used to refer to another CPF record for policy. This results in an additional DNS query to fetch the target CPF record.

all: Matches everything, and is used to define the default policy when previous directives do not match.

The qualifier is a single symbol at the beginning of the directive that indicates the action to execute if the directive is matched. If no qualifier is specified, then the default qualifier "+" is applied. The list of acceptable qualifiers and their associated actions are as follows:

+	**Pass:**	Permit the connection.
-	**Fail:**	Block the connection, and do not offer the user with a means to override
~	**SoftFail:**	Warn the user, but allow them to override the error at their discretion
?	**Neutral:**	Permit the connection and log the result. This qualifier is provided for testing purposes, and does not indicate trust. When the connection is allowed based on a "?" qualifier, it is treated with the same level of trust as a service that does not have a CPF record published.

One of the most important concepts of CPF is the distinction between the " - " and " ~ " qualifiers. The " ~ " indicates a "SoftFail" condition where the certificate is not trusted, but may be permitted by the client application if the user chooses to bypass certificate warnings. The " -" instructs the client application to hard-block the connection without allowing the typical user to override. This feature allows organizations to prevent users from bypassing certificate security for its services, reducing the risk of unintended data exposure. We anticipate that the " - " will be commonly used to protect connectivity with online banking services, trading and brokerage sites, healthcare sites, hosting providers, remote access solutions, and other services where a high level of security is critical.

Publishing Policy

DNS is used to distribute CPF information since it is a natural fit in many ways. DNS infrastructure is already established and highly scalable, with the proven capacity to handle high volumes of queries quickly and efficiently. The CPF policy text is likely to be very short for most situations, thus it can be exchanged easily in a standard DNS UDP packet. In addition, a legitimate server certificate should almost always correlate to a fully qualified hostname such as "host.domain.tld", and thus will be resolvable in DNS. It is also logical to manage CPF information in the same service as the hostname rather than relying upon completely independent systems. The use of an external system would require new infrastructure and dependencies, whereas DNS servers are already deployed and maintained with little overhead. DNS also supports the DNSSEC protocol for digital signing of DNS responses, which can prevent MITM attacks against DNS queries and responses [1].

CPF information is published in DNS as TXT records for the purposes of this work. In the event that this protocol is adopted on a wider scale, it will be ideal to use a dedicated resource record such as "CPF" to identify certificate policy records rather than the generic "TXT" record. The RFC requires the use of a dedicated "CPF" record type, thus only one CPF record may be published for each hostname. The "include" mechanism may be used to refer to another CPF record, which will result in additional DNS queries. The RFC imposes a maximum limit of ten DNS queries when resolving the CPF policy for a hostname; this is in place to prevent abuse by blocking CPF records with an excessive number of includes, or recursive includes that may result in an infinite loop.

Other special conditions to consider:

- *Wildcard certificates* represent any hostname under a domain, not a specific hostname. The reserved hostname "*_wcc_cpf*" is used by CPF when looking up hashes for a wildcard certificate. For example, if a CPF-compatible application is presented with a certificate for "*.example.com" but the URL hostname does not match, then the application will request the CPF information for "_wcc_cpf.example.com".
- *Subject Alternative Names* are used for services accessed via multiple hostnames. This attribute allows a single certificate to identify both the common name and alias addresses. The CPF record for the subject alternative name will likely be the same as the CPF record

for the common name, thus the “include” mechanism may be used to link one to the other for simplified administration.

Accessing and Interpreting Policy

Most applications support DNS resolution, which means that they can easily access CPF records. A CPF-compatible application simply needs to query DNS for the CPF record associated with a remote service and then be able to interpret the policy and enforce it. The application must first determine the hostname to query, the source of which may vary depending upon the circumstances.

1. Generally the application will connect to a URL like https://www.example.com and the hostname will match the certificate common name or a subject alternative name. In this case, the application will query for the hostname specified in the URL.

2. On occasion, the hostname in the URL does not match any of the names assigned to a certificate. This typically occurs if the wrong certificate has been assigned to a service, if the service is being accessed by IP address rather than hostname, or if the hostname in the URL is an alias that is routed properly via DNS but does not match the certificate information. This condition results in the generation of a certificate hostname mismatch error. Most applications warn the user when this occurs and should continue to do so. However, the application should also query for the hostname specified in the certificate common name as a precaution. In the event that a certificate public/private key pair has been compromised, the domain owner could potentially advertise that the certificate is no longer trusted and to block it using the “-“ qualifier. This will allow the client application to detect the issue and protect the user from the potentially malicious resource.

3. Wildcard certificates are formatted as “*.example.com” where “*” can represent any hostname. As long as the hostname in the URL is represented correctly by the certificate, the CPF lookup should be conducted against the URL hostname. If the certificate does not match the URL hostname, then it will be handled as a certificate mismatch error. In this case, the client application will query the CPF record for the hostname “_wcc_cpf” rather than “*”.

When processing the CPF information, one of seven possible CPF results will be returned:

- *None* – no CPF records were published for the target hostname or that the peer address is not eligible for CPF lookups. Ineligible addresses include IP Addresses and unqualified hostnames. The client software cannot ascertain whether or not the peer is authorized using CPF.
- *Pass* – the domain owner has authorized the certificate to represent the services hosted at the given hostname. The client application can trust that the certificate is deemed legitimate from the perspective of the organization operating the service.
- *Neutral* – The domain owner has explicitly acknowledged the existence of a certificate and its association with a specific hostname, but cannot or does not want to assert whether or not the certificate is officially authorized. This is primarily allocated for testing purposes, and should be treated like "None".
- *SoftFail* – Similar to "Fail" but with the distinction that the client application may decide how to handle this condition. It may be configured to log the failure and continue, warn the end-user and require approval to continue, etc. A client application MUST reflect the reduced level of trust in a connection when the "SoftFail" result is returned.
- *Fail* – an explicit statement that a certificate is not authorized to represent the peer hostname. The client application must abort the connection.
- *TempError* – the client application encountered a transient error while performing the DNS query. Checking software can choose to accept or temporarily reject the connection.
- *PermError* – occurs when the CPF record for the hostname could not be correctly interpreted. This signals an error condition that requires manual intervention to be resolved by the domain owner, as opposed to the TempError result. The client application should treat this as either "Fail" or "SoftFail".

When the client application receives a response to the DNS query, it must parse the results into a list of directives much like an access control list (ACL). Each directive is treated as an access control entry (ACE), and they are processed from first to last and executed on a "first hit" basis. The application must compare the certificate's hash value to each entry for matching criteria. If there is a match, then the qualifier's corresponding CPF result is returned to the client application. If a match is not found, the default response of SoftFail (~) is applied.

Evaluation

In order to demonstrate the usage and effectiveness of the CPF protocol, we have provided a basic text-based web browser that adheres to the CPF RFC. The browser was subjected to twenty-four tests to calculate compatibility and effectiveness with CPF, and to determine any delay incurred due to processing and network performance overhead. Each of the tests is run in two phases – three trials at LAN speed, and three trials in an Internet-simulated environment.

CPF-Compatible Browser Design

The proof-of-concept (POC) web browser is programmed using Java 1.6.x and Perl 5.12. The Java application "CPF-POC.jar" accepts a URL as a command-line argument, connects via HTTPS, and fetches the server's public key certificate. It then extracts the common name and subject alternative names from the certificate and compares them to the hostname specified in the HTTPS URL. Once the application verifies the validity of the certificate, it will pass the appropriate hostname to a Perl-based CPF resolver.

The custom Perl resolver "cpf-fetch.pl" leverages Net::DNS to conduct a series of DNS queries related to CPF, with additional logic to follow CPF-specific mechanisms like "includes". It also validates the final CPF information to ensure that errors are properly detected and handled. Finally, it prepares the policy in an ACL format, notes any errors, and returns the results back to the Java browser for interpretation and enforcement.

The browser's CPF interpreter parses the output from the Perl script and determines whether the CPF ACL needs to be processed. If no CPF record was present or an error was returned during the CPF resolution/validation process, then the CPF interpreter will issue a result immediately. Otherwise, the hashes of the peer's certificate will be compared to each ACL entry. When a final result is determined, the application will act on this information by allowing the connection, blocking the connection without the ability to override, or warning the end-user and prompting for approval to continue.

Environment Configuration

The test environment is composed of three Linux virtual machines – a workstation (vPC1), a web server (THS-Web1), and a DNS server (THS-DNS1). The web server hosts a sample website via HTTPS using Apache2, and the DNS server acts as both a recursive and authoritative nameserver for our tests using Bind 9. The hosts are all directly connected over a 1 Gbps link to an Ethernet switch. In the first phase of each test, all three devices are connected via the same network switch for a "LAN-based test" as shown in figure 5. This environment minimizes latency and delays, allowing us to observe protocol performance under optimal conditions.

In the second phase of each test, we deployed a WAN emulation virtual appliance on the network running WAN-Bridge LiveCD[4]. This device acts as a layer-2 Ethernet bridge between the PC and servers as shown in figure 6, restricting bandwidth to 3072 Kbps and adding 20 ms of latency to each Ethernet frame that traverses it. This "Internet simulation" allow us to better gauge the end-user experience when using CPF over the Internet.

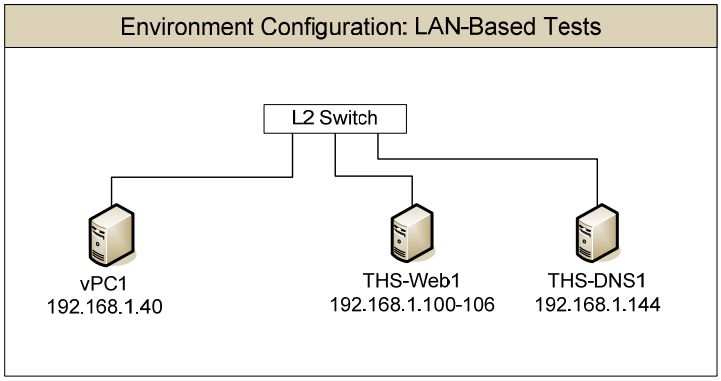


Figure 5: Network diagram of environment for LAN-based testing

[4] http://code.google.com/p/wanbridge/

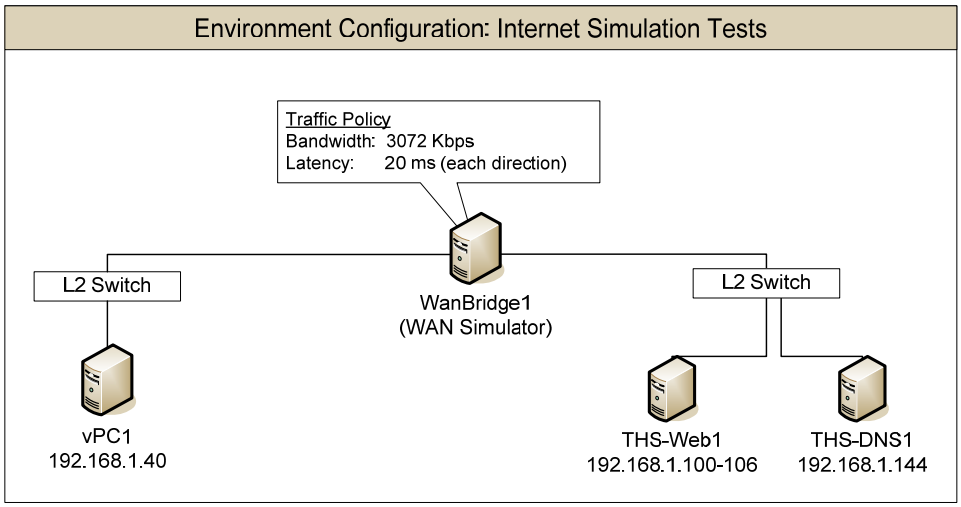


Figure 6: Network diagram of environment for Internet simulation testing

The DNS protocol does not natively provide authenticity or integrity, two key elements that prevent DNS traffic from being transparently manipulated. Although DNS is the only functional requirement for the CPF protocol, some form of digital signing or encryption must be applied in order for CPF to provide seamless protection under a MITM attack. The DNSSEC protocol extends DNS to offer these features, although it is not widely implemented to date.

A limited test has been devised to explore the potential performance impact when combining CPF with DNSSEC. The environment is similar to previous experiments, except that a dedicated DNS resolver "THS-DNSResolver" has been added (figure 7) to conduct recursive DNS queries and validate the DNSSEC responses. Server THS-DNS1 is running a cryptographically signed version of the zone used in experiment #7, as noted in table 2. In this experiment, the workstation vPC1 directs all DNS queries to THS-DNSResolver, which is configured to forward queries for covertpacket.com to THS-DNS1. The "named" service on THS-DNSResolver has been configured to use the public key of THS-DNS1 as a DNSSEC trust anchor, allowing it to validate the responses and detect unauthorized changes.

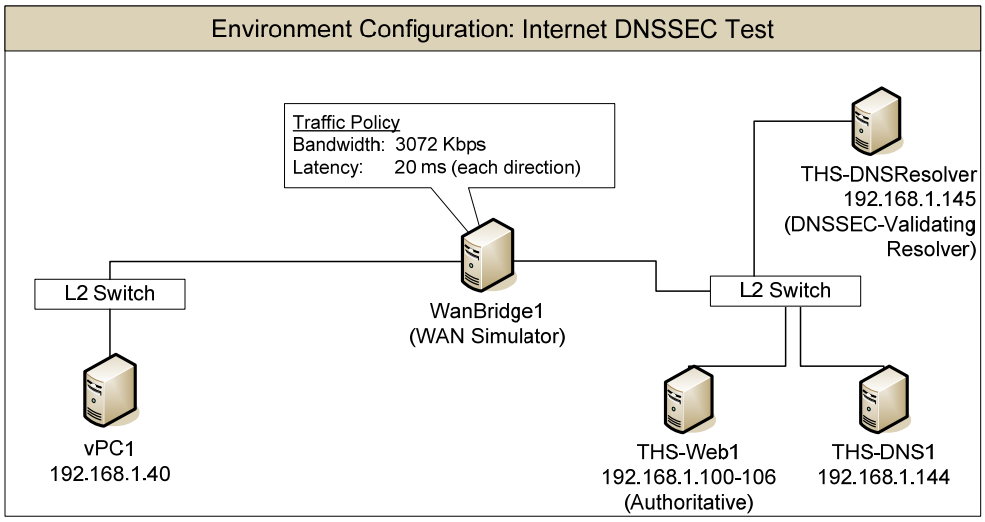


Figure 7: Network diagram of Internet simulation environment with DNSSEC applied to experiment #7

Reliability Results

Our web browser was subjected to twenty-four unique experiments listed in table 2, to confirm that it will behave in accordance with the CPF RFC. These tests target key features and functionality related to the detection and handling of errors, the ability to work with other protocols, successful interpretation of policy, and proper enforcement of results. We found that the web browser successfully completed all tests with the anticipated results. This data indicates that the CPF protocol can be implemented in third-party applications with relative ease and predictability.

#	Description	Result
1	Demonstrate that the "-" qualifier is properly interpreted and enforced per RFC	**Pass**
2	Demonstrate that the "~" qualifier is properly interpreted and enforced per RFC	**Pass**
3	Demonstrate that the default "~all" directive is enforced when a CPF record is returned but no directives are matched.	**Pass**
4	Demonstrate that the "+" qualifier is properly interpreted and enforced per RFC	**Pass**
5	Demonstrate that the hostname from the certificate common-name is used for the CPF lookup when a certificate hostname mismatch occurs.	**Pass**
6	Demonstrate that default qualifier "+" is applied on directives lacking an explicit qualifier	**Pass**
7	Demonstrate that the "-" qualifier will be enforced on the hash mechanism	**Pass**
8	Demonstrate that the "~" qualifier will be enforced on the hash mechanism.	**Pass**
9	Demonstrate that the "?" qualifier will be enforced on the hash mechanism.	**Pass**
10	Demonstrate that CPF will default to "~all" (SoftFail) when directives are available but none are matched.	**Pass**
11	Demonstrate that the CPF interpreter will detect a malformed hash mechanism	**Pass**
12	Demonstrate that CPF validates the protocol version	**Pass**
13	Demonstrate that the CPF interpreter will detect multiple CPF records for a hostname and	**Pass**

	result in protocol failure.	
14	Demonstrate that the CPF interpreter will detect an invalid mechanism	**Pass**
15	Demonstrate that SSL connections are permitted when a CPF record is not advertised	**Pass**
16	Demonstrate that CPF follows "include" mechanisms	**Pass**
17	Demonstrate that the CPF interpreter limits the number of DNS queries per RFC	**Pass**
18	Demonstrate that CPF resolver will conduct lookups based on the URL hostname when the service is identified by a valid wildcard certificate.	**Pass**
19	Demonstrate that an "include" can be used to link a host's CPF information to a corresponding wildcard certificate	**Pass**
20	Demonstrate that large CPF records (>512 bytes in size) will be resolvable via DNS using TCP	**Pass**
21	Demonstrate that CPF resolver will conduct lookups based on the URL hostname when the service is identified by a certificate subject-alternative-name.	**Pass**
22	Demonstrate that CPF resolver will conduct lookups based on the certificate common name when a hostname mismatch occurs between the URL hostname and the certificate	**Pass**
23	Demonstrate that the CPF resolver checks for a corresponding A-record before conducting a CPF lookup.	**Pass**
24	Demonstrate that CPF resolver will conduct lookups based on the wildcard certificate's domain when the remote service's hostname is not valid for a wildcard certificate.	**Pass**

Table 2: Summary of tests executed by CPF-compatible browser, and whether or not the final result complied with RFC specifications.

Performance Results

In order for CPF to be a practical solution, it must improve security without significantly hindering the end-user's browsing experience. Since the protocol builds onto existing browser functionality, the minor increase in network and computational overhead will have some impact on content delivery. Our tests measure the time it takes to resolve and process CPF records under a variety of common circumstances. These measurements exclude the time required to load the application into memory, as this will vary between applications.

In order to resolve a basic CPF record consisting of a single SHA1 hash and an "all" mechanism, approximately 495 bytes of data is exchanged as indicated in tests 7-9. The time needed to acquire, validate, and process CPF information ranges between 6 ms and 8 ms in a LAN environment, and 93 ms to 106 ms in the Internet-simulated environment. These delays and bandwidth needs are negligible, and would not be noticed by end-users in either situation. The timings observed in CPF are similar to those experienced when using VeriKey, which range between 204 ms and 885 ms for lookups against notary servers in the same geographic region [17].

Overall bandwidth needs increase when fetching CPF records that are larger in size or employ the "include" mechanism. In addition, numerous DNS queries extend the total duration of time needed to fetch and process CPF policy due to the compounding effects of network latency. For example, test 17 follows a recursive CPF "include" ten times before reaching the maximum lookup count, resulting in a total of 11 DNS queries. This resulted in the exchange of 2,733 bytes of data, and averaged 505 ms to process in the Internet-simulated environment. Similarly, test 20 specifies a CPF record with 13 different SHA1 hashes, which exceeds 512 bytes in length and forces DNS to renegotiate transport protocols from UDP to TCP. In this unusual scenario, 1,330 bytes are exchanged and processed in an average of 186 ms. Both of these tests complete in well under one second, despite the uncommon data requirements and communications needs associated with the CPF policies used in these particular tests.

The performance results of CPF are promising in both the LAN environment and Internet-simulated environment, and the tests demonstrate that a fast response can be expected in both common-sized policies and large policies. This was expected, since a CPF record is generally comparable in size to a query for a CNAME or a zone's MX records, which are routine queries. Based on these results, we are confident that CPF can be implemented with little impact to the end-user experience.

Performance Impact from DNSSEC

When DNSSEC is applied in the LAN environment, the average DNS resolution time is 12 ms compared to 6 ms without DNSSEC. Similarly, in the Internet-simulation environment, we found that the average DNS resolution time is approximately 105 ms as opposed to 97 ms without DNSSEC. Based on these results, the average delay incurred by using DNSSEC is 8 ms – a minimal delay. The DNS payload size for CPF with DNSSEC is substantially larger at 1,211 bytes, from 495 bytes without DNSSEC. This increase can be mainly attributed to the inclusion of the signed hashes for both the target resource record (TXT or A) and the NS resource record in each DNS response.

The true overhead introduced by DNSSEC may vary based upon a number of variables. The size of the DNS data exchange is most dependent upon the length of the DNSSEC zone-signing key (ZSK) and the length of the domain name. The DNS resolver may introduce additional latency when fetching DNSSEC-related resource records, validating the chain of trust, and returning the

DNS response. However, it can also leverage caching to improve performance. It's also worth noting that DNSSEC is required to secure all DNS responses, not just CPF records. Therefore, similar overhead would be experienced when resolving common resource record types, such as A-records and MX records. The overhead of DNSSEC on CPF records would be comparable to other records. While all the conditions related to DNSSEC are difficult to predict, our testing generally suggests that DNSSEC does not substantially impact response time. We do not believe that it will significantly affect DNS queries for CPF records in a manner that would be noticeable to the end-user.

Future Work

CPF Mechanism Expansion

The current list of mechanisms supported in CPF accommodates most common situations and use cases. It would be worthwhile to explore the expansion of mechanism definitions to include other DNS record types. For example, if the CAA record continues to mature and gain industry support, it would be beneficial to use this information to verify certificate authorization based on the signing CA. This will provide a more flexible approach to authorization, with reduced maintenance requirements as compared to publishing hashes for specific public key certificates.

Application Support

Applications, services, and operating systems will need to be updated to support CPF. This work provides the necessary information and a proof-of-concept example to aid in the development process. In addition, the use of DNS as a distribution model simplifies the integration process. It will be beneficial to add CPF support to any application that exchanges public keys or public key certificates, such as web browsers, SSH clients, proxy appliances, and SMTP servers. DNS services will also need to be updated to support a new "CPF" resource record type.

Integration with Convergence

CPF and Convergence approach certificate verification in different ways, both of which have advantages and disadvantages. CPF offers an authoritative answer as specified by the service owner, which we consider to be a more trustworthy indicator. Likewise, Convergence allows the

user to select notaries for verification, which assures consistency of the public keys provided by a service. One of the advantages of Convergence is that it provides end-users with enhanced security for public services regardless of the service owner's support, whereas CPF requires the service owner to maintain a set of records for certificate authorization. However, CPF offers a way to authenticate public keys without direct network access to the service by verification/notary servers.

Convergence notaries have the ability to support multiple verification backends, such as Perspectives, DNSSEC, and CA PKI [12]. It may be worthwhile to develop a CPF backend for Convergence; this would allow Convergence end-users to benefit from CPF when it is available for a hostname, but have the ability to fall-back on an alternate backend when CPF records have not been published. As an additional benefit, the CPF lookups would be conducted from an independent network, ensuring consistency. While DNSSEC is recommended with the use of CPF, verifying the public key via a CPF backend in Convergence would offer additional security for unsigned DNS responses. Since DNSSEC has not been widely adopted thus far, this would be a welcome alternative.

Certificate Assurance Grading

Most applications identify a certificate as being either "trusted" or "untrusted". However, in the real world people tend to grant trust in varying levels based on their knowledge of an individual and his or her actions, appearance, credentials, background, etc. As new verification methods continue to progress and we move away from a system solely managed by CAs, it may be beneficial to assign connections a "trust grade" based on the application's level of assurance that the connection is secure and authentic. This would allow users to judge the level of trust when conducting different actions online. For example, a basic 2048-bit SSL certificate signed by a CA may be trusted, but offers a low degree of assurance. This may be acceptable for a website's contact form or a specialized web application, but users may exercise greater caution when exchanging sensitive information. In comparison, a 2048-bit certificate that's fully validated by the CA, only accepts strong TLS/SSL ciphers, and has a published CPF record offers a high degree of assurance that is more appropriate for use with webmail applications, remote access solutions, and online banking.

Conclusion

Digital trust is an important aspect of network security that millions of people depend upon on a regular basis. PKI plays a vital role in this area, and we believe that the Internet community will continue to rely upon this trust model for the foreseeable future. However, in order for PKI to be a reliable solution, the vulnerabilities related to trust confinement must be resolved.

In this work, we present CPF as a unique and effective approach that allows network service owners to publish certificate authorization policies using the existing DNS infrastructure. As a result, organizations gain the ability to control authentication for its services, creating a second viewpoint external to CAs that counteracts many of the negative repercussions resulting from a compromised CA or from fraudulent certificates. In addition, the CPF protocol allows for greater granularity to protect end-users transparently, enhancing the security of certificates and public keys on a broad level.

In recent years, researchers have made significant progress identifying weaknesses in PKI and certificate-based security, and many have contributed thoughtful ideas and solutions for these issues. The CPF protocol was developed with many of the same issues in focus, and embraces the same common goal of improving online trust. We believe that the previous and ongoing efforts discussed earlier in this work offer valuable solutions, and respectfully propose CPF as a complementary solution rather than as an alternative. CPF is able to work collaboratively with Convergence/Perspectives, CAA records, TLSA records, and browser certificate inspection as an additional layer of security, offering greater overall protection.

We have discussed the goals, design, and implementation of CPF throughout this work, and demonstrated that it is capable of enhancing certificate-based security while maintaining an acceptable level of performance. Our work has the potential to benefit a great number of organizations and individuals. We hope that industry leaders will consider it as a solution or as a basis for further improvements.

References

[1] Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005, March 1). DNS Security Introduction and Requirements. *IETF Tools*. Retrieved July 5, 2011, from http://tools.ietf.org/pdf/rfc4033

[2] DNS-based Authentication of Named Entities (dane) - Charter. (n.d.). *IETF Datatracker*. Retrieved July 13, 2011, from http://datatracker.ietf.org/wg/dane/charter/

[3] Eckersley, P., Burns, J. (2010, July 30). An Observatory for the SSLiverse [PDF slides]. Retrieved August 14, 2011, from http://www.eff.org/files/DefconSSLiverse.pdf

[4] Goodin, D. (2011, June 21). Web authentication authority suffers security breach. *The Register*. Retrieved July 15, 2011, from http://www.theregister.co.uk/2011/06/21/startssl_security_breach/

[5] Hallam-Baker, P., Stradling, R., & Laurie, B. (2011, May 10). DNS Certification Authority Authorization (CAA) Resource Record. *IETF Tools*. Retrieved June 20, 2011, from http://tools.ietf.org/pdf/draft-hallambaker-donotissue-04.pdf

[6] Hoffman, P., & Schlyter, J. (2011, July 25). Using Secure DNS to Associate Certificates with Domain Names For TLS. *IETF Tools*. Retrieved July 29, 2011, from http://tools.ietf.org/id/draft-ietf-dane-protocol-09.txt

[7] Housley, R., Polk, W., Ford, W., & Solo, D. (2002, April). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *IETF Tools*. Retrieved October, 2011, from www.ietf.org/rfc/rfc3280.txt

[8] Keizer, G. (2011, July 27). Sniffer hijacks secure traffic from unpatched iPhones. *Computerworld*. Retrieved July 29, 2011, from http://www.computerworld.com/s/article/9218676

[9] Leyden, J. (2008, December 29). CA issues no-questions asked Mozilla cert. *The Register*. Retrieved July 17, 2011, from http://www.theregister.co.uk/2008/12/29/ca_mozzilla_cert_snaf/

[10] Lopez, J., Oppliger, R., & Pernul, G. (2005). Why Have Public Key Infrastructures Failed so Far? *Internet Research, 15*(5), 544–556.

[11] Marlinspike, M. (2009, July 29). Null Prefix Attacks Against SSL/TLS Certificates. Retrieved October 1, 2011, from www.thoughtcrime.org/papers/null-prefix-attacks.pdf

[12] Marlinspike, M. (August 2011). SSL and the Future of Authenticity [Video file] Retrieved on September 7, 2011, from http://www.youtube.com/watch?v=Z7Wl2FW2TcA

[13] Mike Burmester and Yvo G. Desmedt. 2004. Is hierarchical public-key certification the next target for hackers?. *Commun. ACM* 47, 8 (August 2004), 68-74. DOI=10.1145/1012037.1012038 http://doi.acm.org/10.1145/1012037.1012038

[14] National Vulnerability Database (NVD) National Vulnerability Database (CVE-2011-0228). (2011, August 29). *National Vulnerability Database Home*. Retrieved October 1, 2011, from http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-0228

[15] Prins, J. R. (2011). *DigiNotar Certificate Authority Breach - Operation Black Tulip*. The Netherlands: Fox-IT BV. Retrieved September 13th, 2011 from http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2011/09/05/diginotar-public-report-version-1/rapport-fox-it-operation-black-tulip-v1-0.pdf

[16] Schneier, B., & Ellison, C. (2000). Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure. *Computer Security Journal, 16*(1), 1-7. Retrieved May 8, 2011, from http://www.schneier.com/paper-pki.pdf

[17] Stone-Gross, B., Sigal, D., Cohn, R., Morse, J., Almeroth, K., & Kruegel, C. (2008). VeriKey: A Dynamic Certificate Verification System for Public Key Exchanges.

Detection of Intrusions and Malware & Vulnerability Assessment (pp. 44-63). New York: Springer Berlin / Heidelberg.

[18] Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., & Cranor, L. F. (2009). Crying Wolf: an Empirical Study of SSL Warning Effectiveness. *Proceedings of the 18th Conference on USENIX Security Symposium* (pp. 399 - 416). Montreal: USENIX Association.

[19] Weise, Joel (2001). Public Key Infrastructure Overview. *Sun Blueprints Online*. Retrieved May 2, 2011, from http://www.sun.com/blueprints/0801/publickey.pdf

[20] Wendlandt, D., Andersen, D., & Perrig, A. (2008). Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. *USENIX 2008 Annual Technical Conference on Annual Technical Conference (ATC'08)*, 321-334. Retrieved September 5, 2011, from http://www.cs.cmu.edu/~dga/papers/perspectives-usenix2008.pdf

[21] Wong, M. W., & Schlitt, W. (2006, June). RFC 4408 - Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. *IETF Tools*. Retrieved June 13, 2011, from http://tools.ietf.org/pdf/rfc4408

[22] Zetter, K. (2011, March 23). Hack Obtains 9 Bogus Certificates for Prominent Websites; Traced to Iran. *Wired.com*. Retrieved May 9, 2011, from http://www.wired.com/threatlevel/2011/03/comodo-compromise/

Appendices

- Certificate Policy Framework Request for Comment (RFC) Draft
- Logic Diagram for Proof-of-Concept HTTPS Browser

Certificate Policy Framework Request for Comment (RFC) Draft

UNASSIGNED
Request for Comments: DRAFT-UNASSIGNED
Category: Experimental

M. Lidestri

August 2011

Certificate Policy Framework (CPF) for Distributing Certificate Authorization/Policy, Version 1

Status of This Memo

This memo defines an experimental protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested.

Version

Ver 1.3 DRAFT

Table of Contents

1. Introduction

Public Key Infrastructure (PKI) has been widely implemented as a solution for instilling trust in public key certificates. However, one significant risk with PKI is the unconditional trust granted to certificate authorities (CAs). Applications and operating systems consider a CA as untrusted or completely trusted, with no further granularity. This means that any certificate issued by a trusted CA will be considered legitimate.

This document defines a protocol for publishing certificate controls to client applications and services using the domain name service (DNS). Domain owners may authoritatively offer certificate policy and authorization information for a given hostname using the CPF resource record. Likewise, client applications may acquire certificate policy by querying DNS for the CPF record of the hostname provided, which may then be compared to a certificate's attributes.

The CPF record is an access-control list for certificates; it may identify both specific certificates and certificates with certain attributes, and dictates how each entry should be handled. CPF policy information will enable applications to determine the legitimacy of a certificate authenticating a service from the perspective of the authoritative organization. This offers a greater level of assurance than that of a third party CA, which has no direct authority over the service. In addition, it allows the service owners to dictate how client applications should behave when presented with an untrusted certificate, further protecting themselves and their customers, clients, partners, and users.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Common Name – X509v3 certificate attribute which declares the primary identity. In the context of this document, the common name is always a fully qualified domain name (FQDN).

Subject Alternative Name – x509v3 certificate attribute which declares secondary/alias identities related to the common name.

Domain Owner – the individual(s) responsible for managing DNS zone information.

2. Operation

This section describes how CPF should be published and handled, at a high level. The details of the protocol syntax are further documented in sections 3 and 4.

2.1. Publishing Authorization

A CPF-compliant hostname must advertise a CPF record, as described in section 3. This record identifies certificates that may represent services located at a given hostname, and indicates how a client application should react to a given certificate.

If domain owners choose to publish CPF records, it is RECOMMENDED that they end in “-all” or “~all” so that a definitive determination of authorization can be made. If a final action is not defined, the default action should be enforced by the client application as defined in section 4.5.

2.2. Checking Authorization

A CPF-compatible application may perform CPF lookups for protocols protected using certificates, such as SSL-enabled services including HTTPS, SMTPS, LDAPS, etc.

2.3. Interpreting the Result

2.3.1. None

A result of "None" means that no CPF records were published for the target hostname or that the peer address is not eligible for CPF lookups. Ineligible addresses include IP Addresses and unqualified hostnames. The checking software cannot ascertain whether or not the peer is authorized.

2.3.2. Pass

A "Pass" result means that the domain owner has authorized the certificate to represent the services hosted at the given hostname. The client application can trust that the certificate is deemed legitimate from the perspective of the organization operating the service.

2.3.3. Neutral

The domain owner has explicitly acknowledged the existence of a certificate and its association with a specific hostname, but cannot or does not want to assert whether or not the certificate is officially authorized. A "Neutral" result MUST be treated exactly like the "None" result; the distinction exists only for informational purposes to facilitate testing.

2.3.4. SoftFail

A “SoftFail” must be treated as a “Fail”, with the distinction that the client application may decide how to handle this condition. It may be configured to log the failure and continue, warn the end-user and require approval to continue, etc. A client application MUST reflect the reduced level of trust in a connection when the “SoftFail” result is returned.

This option has been created to allow a domain owner to permit a bypass mechanism in the event of a failure. Use of this option is

not recommended, especially for services which are accessed over public or untrusted networks.

2.3.5. Fail

A "Fail" result is an explicit statement that a certificate is not authorized to represent the peer hostname. The client application must abort the connection. This action prevents the software or end-user from bypassing the failure and exchanging data over an untrusted and potentially compromised connection.

If the domain owner does not wish to hard-block users from a hostname on certificate authorization failure, then he may configure the CPF record to result in a “SoftFail” instead.

2.3.6. TempError

A "TempError" result means that the client application encountered a transient error while performing the DNS query. Checking software can choose to accept or temporarily reject the connection.

2.3.7. PermError

A "PermError" result means that the CPF record for the hostname could not be correctly interpreted. This signals an error condition that requires manual intervention to be resolved by the domain owner, as opposed to the TempError result. The client application should treat this like a “Fail” or “SoftFail” from a behavioral standpoint.

3. CPF Records

A CPF record is a DNS resource record (RR) used to publish certificates that are, or are not, allowed to represent services located at a specific hostname.

A CPF record is a single, case-insensitive string of text, processed from left to right. Please consider the following example:

```
v=1 hash_sha1:6a0e9a60583c365eedafad7f4010965515dc014a –
hash_sha1:42ac0d3e30198c893a1f301939ace903019355ec ~all
```

This record has a CPF protocol version of “1” and three directives:

1. Allow certificate with SHA-1 hash of 6a0e9a60583c365eedafad7f4010965515dc014
2. Fail on certificate with a SHA-1 hash of 42ac0d3e30198c893a1f301939ace903019355ec.
3. Soft-fail on any other certificates representing this hostname.

3.1. Publishing

CPF-compliant services must have CPF records published for applicable hostnames by the domain owner.

The example above in Section 3 might be published via these lines in a domain zone file:

```
www.example.com.        IN CPF      “v=1
hash_sha1:6a0e9a60583c365eedafad7f4010965515dc014a –
hash_sha1:42ac0d3e30198c893a1f301939ace903019355ec ~all”

mail.example.com.       IN CPF      “v=1
hash_sha1:938ca8e9a284355ce1a7ff7621c1d2d876ab2543 –all”
```

CPF records are comprised of text-based content, and can be lengthy depending on the hash algorithms used and number of certificate hashes contained in a given record. Large records may cause problems with size limits; this is documented further in section 3.1.5.

3.1.1. DNS Resource Record Types

This document defines a new DNS RR of type CPF, (CODE TBD). The format of this type is identical to the TXT RR [RFC1035]. The character content of the record is encoded as [US-ASCII].

The TXT RR is a common method for distributing text-based information via DNS. However, since the TXT resource record is used generically for multiple purposes, its use may lead to unnecessary bandwidth usage and processing. The CPF RR has been designated in order to avoid this caveat. The TXT RR must not be used to distribute CPF record information.

3.1.2. Scope

A CPF record MUST be applied to a specific DNS hostname, with the exception of wildcard certificates (discussed in 3.1.6). A networked service is traditionally identified by hostname, and as such CPF records should be matched one-to-one with this. Certificates traditionally authenticate a service by hostname as either the “Common Name” or “Subject Alternative Name” attribute.

3.1.3. Multiple DNS Records

A hostname MUST NOT have multiple CPF records published. In the event that more than one record is returned, the result “PermError” should be issued.

3.1.4. Multiple Strings in a single DNS Record

As defined in [RFC1035] sections 3.3.14 and 3.3, a single text DNS record can be composed of more than one string. If a published record contains multiple strings, then the record MUST be treated as if those strings are concatenated together without adding spaces. For example:

```
IN CPF "v=1 .... first" "second string..."
```

MUST be treated as equivalent to

```
IN CPF "v=1 .... firstsecond string..."
```

CPF records containing multiple strings are useful in constructing records that would exceed the 255-byte maximum length of a string within a single CPF RR record.

3.1.5. Record Size

The published CPF record for a given hostname SHOULD remain small enough that the results of a query for it will fit within 512 octets. This will keep even older DNS implementations from falling over to TCP. Since the answer size is dependent on many things outside the scope of this document, it is only possible to give this guideline: If the combined length of the hostname and the text of a CPF record is under 450 characters, then DNS answers should fit in UDP packets. Records that are too long to fit in a single UDP packet should fail to TCP. In the event that the entire record cannot be retrieved, the PermError result should be returned.

3.1.6. Wildcard Certificates

Wildcard certificates are commonly used to represent any hostname under a given domain, where the common name attribute resembles “*.example.com”. From a security perspective it’s not ideal to use wildcard certificates, but there are practical use cases for them.

A CPF resource record for wildcard certificates MUST be applied using a special, reserved hostname “_wcc_cpf” (for “wildcard certificate CPF). In order to enable CPF for a service using a wildcard certificate, the service’s hostname should advertise a CPF record similar to “include:_wcc_cpf.example.com”.

3.1.7. Subject Alternative Names

Subject Alternative Names are used for services which are accessed via multiple hostnames. This attribute allows a single certificate to identify both the common name and alias addresses.

The CPF lookup SHOULD be initiated for the hostname requested by the application in order to support multitasking lookups. The CPF record for the Subject Alternative Name will likely be the same as the CPF record for the common name. The domain administrator may use the “include” mechanism to link one to the other, allowing for simpler management.

3.1.8. CPF Record Caching and TTL

CPF records are subject to caching, just like any other DNS resource record. Since CPF information reflects certificate authorization and access controls, it’s ideal to refresh this information on a more regular basis. It is recommended to set the time-to-live (TTL) for these records between 20 minutes and 4 hours.

4. Client Lookups

4.1. Arguments

In order to query the CPF record via DNS, the client application SHOULD be provided with a fully-qualified hostname, such as mail.example.com. IP addresses are not DNS-resolvable and thus are not supported by CPF. It is NOT RECOMMENDED to conduct CPF lookups on unqualified names where a domain name is not provided. Certificate name attributes should always be fully-qualified for security reasons, and CPF embraces this practice.

4.2. Initial Processing

If the hostname is malformed per [RFC1034] name space specifications or is not fully qualified, or if the DNS lookup returns "domain does not exist" (RCODE 3), the client application must reflect the result as “none”.

Mechanisms like “include” will require DNS lookups to fetch additional CPF information. A maximum of ten (10) DNS queries may be executed as part of a CPF lookup, including the initial CPF query. If this limit is exceeded, then the application should abort the lookup and return a “PermFail” result.

4.3. Record Validation

After one CPF record has been returned, the client application must validate the syntax, and if successful will process the directives. If there is a syntax error in the CPF record, then the application will treat the result as “PermError” and discontinue processing.

CPF entries are space-delimited, and read from left to right. The first entry MUST be v=#, where “#” is the CPF protocol version. Each following entry is a directive comprised of a mechanism, and optionally a qualifier preceding it. The last entry should consist of the “all” mechanism, with the desired qualifier. Please see the example below.

```
v=1 hash_sha1:938ca8e9a284355ce1a7ff7621c1d2d876ab2543 ~all
\_/ \_____________________________________________/ \___/
 |                          |                         |
Version      Directive 1 – hash mechanism     Directive 2 – “all”
             with default qualifier of “+”    mechanism with “~”
                                              qualifier
```

Mechanisms and qualifiers are documented in greater detail in section 4.4

4.4. Record Processing

The CPF record is interpreted one entry at a time, on a “first hit” basis much like an access control list. If the entire record is processed without any explicit matches, then the default result will be applied as documented in section 4.5.

Each mechanism is an identifier for a certificate, or another DNS resource record identifying a certificate. Qualifiers specify the action to take on a specific mechanism. The combination of the

mechanism and qualifier is a directive, which CPF-compatible clients are able to enforce.

The possible qualifiers, and the results they return are as follows:

"+" Pass
"-" Fail
"~" SoftFail
"?" Neutral

The qualifier is optional and defaults to "+" if unspecified.

The general syntax is:

[+|-|~|?]<mechanism>:<data>

For example:

cpf:example.com
~hash_sha1:6a0e9a60583c365eedafad7f4010965515dc014a

4.5. Default Result

If none of the directives match and there is no “all” directive, then the application SHOULD implicitly enforce this as “~all”.

4.6. Lookup Methodology

This section offers guidelines for an application to conduct lookups, to ensure the most practical and secure implementation.

4.6.1. Order-of-Operations

An application should conduct CPF lookups using a hostname provided from one of two sources:

- Destination string (sub-component of URL)
- Certificate “common name” attribute

The following order of operations is RECOMMENDED:

```
//Part 1 – lookup by destination string (preferred)
IF the destination string is a hostname
   THEN initiate CPF lookup on destination string.
END IF

//Part 2 – lookup by certificate common-name (fallback approach)
IF the destination string is an ip-address OR the certificate name
attributes do not match the destination string (hostname mismatch)
   THEN initiate CPF lookup on certificate common name
END IF
```

This approach is optimal because the destination string should always match a certificate name attribute for a legitimate service. This also allows multi-threaded applications to conduct the CPF lookup and download the remote service’s certificate simultaneously, minimizing delays.

In the case that a hostname mismatch occurs, the certificate SHOULD be looked up independently. Some applications, such as web browsers, display a warning when a name mismatch occurs. However, by conducting this additional CPF lookup, we may learn that the domain owner for the certificate common name has blocked the signature for the provided certificate. This may happen if the public/private key pair has been compromised. In the event that the CPF lookup on the “common name” attribute returns a soft-fail or fail result, this SHOULD be enforced by the application. Other results (pass, neutral, etc.) may be ignored and the default application settings for certificate mismatches should be enforced.

4.6.2. Pre-fetching Data

A remote service that is CPF-compatible MUST be verified prior to the exchange of application data in order to minimize the risk of data exposure.

Some applications may wish to download content before the identity of the remote service has been verified via CPF, and then process the cached data if approved. This behavior is often implemented in order to accelerate content delivery and improve the user’s experience. However, certain information may be exposed to an untrusted party unintentionally. For example, an HTTP GET request is used to download content, but may include a query string with sensitive variables such as a session ID.

5. Mechanism Definitions

5.1. “all”

The “all” mechanism is a default that will match anything, and is the last mechanism evaluated in the record. This acts as a default directive, which will take effect if no previous criteria are matched.

For example:

```
v=1 hash_sha1:6a0e9a60583c365eedafad7f4010965515dc014a -all
```

This entry will allow a certificate matching the sha-1 signature specified, and deny all other certificates.

5.2. “include”

The "include" mechanism allows a CPF record to link to the CPF record of a different hostname. This is useful when services accessed at a specific hostname are being represented by wildcard certificates, or if the hostname is a subject alternative name for the service.

All directives of an included CPF record are processed except the “all” mechanism (if present). Matches must be honored, regardless of the qualifier applied to the directive.

For example, www.example.com is identified by a wildcard certificate:

```
www   IN CPF      “v=1 include:example.com –all“
      IN CPF      “v=1 hash_sha1:
      6a0e9a60583c365eedafad7f4010965515dc014a –all
```

This will direct the client to query the CPF record for example.com as well, and apply it to the hostname www.

5.3. “hash” (hash_*<algorithm>)*

The “hash” mechanism provides the hash value of a public key certificate. The mechanism suffix specifies the hash value selected by the domain owner. A fixed number of hash algorithms will be supported by CPF, to ensure greater compatibility and security. The following hash mechanisms are currently supported:

```
hash_sha1:     SHA1 hash of certificate
hash_sha256:   SHA256 hash of certificate (RECOMMENDED)
hash_sha512:   SHA512 hash of certificate
```

The hash value should be generated from the PEM format of a certificate, with all new-line characters removed (carriage return, line feed, etc). This will ensure that the hash value is generated properly regardless of operating system or application.

5.4. Future expansion

There are several DNS resource record types emerging for identifying certificates; one example is the CERT record defined by RFC draft-hallambaker-certhash-00, or the CAA record defined by RFC draft-hallambaker-donotissue-03. If adopted by the global IT community, these could be used as mechanisms in the CPF protocol.

6. Considerations

6.1. DNS Security

CPF records are exchanged via DNS, which is an unencrypted protocol by default. This makes it susceptible to a network-based man-in-the middle (MITM) attack, where the DNS packet can be altered to neutralize CPF. The use of DNSSEC is recommended to assure the integrity and authenticity of DNS traffic, including CPF traffic.

6.2. Local Attacks

CPF can mitigate network-based (MITM) attacks in most circumstances. However, its benefits are limited against local attacks. If malware infects a machine, it can potentially manipulate an application or operating system. Potential attacks include, but are not limited to, the following:

- Compromise an application and alter its behavior to see every CPF lookup as a non-failing result like “none” or “pass”. A man-in-the-browser (MITB) attack would be one example of this.
- Compromise an operating system and act as a shim to intercept API calls for DNS queries, and always return a non-failing result.

6.3. Compromised DNS Services

If a DNS server were compromised, the attacker may gain the ability to alter DNS records including CPF. By itself, this type of incident does not significantly impact the potential to misrepresent a service since a certificate still needs to be issued by a valid CA. However, it could be used to launch a denial-of-service attack by changing the CPF record to “-all”.

6.4. Compromised Certificate Management Services

A number of certificate authorities provide web-based portals for certificate management, protected by username/password authentication. These portals typically allow users to create, revoke, or re-issue certificates.

If a certificate management account were compromised, the attacker could gain the ability to order new certificates or revoke/re-issue an existing certificate. Depending on the validation checks conducted by the CA, a valid certificate may be issued to the correct account, but under the control of an illegitimate individual. This would enable the attacker to deploy services and identify themselves with legitimacy, endangering any client system initiating the connection.

CPF-compliant hostnames will have less exposure than hostnames lacking a CPF record. So long as the client application supports CPF lookups, it will compare the re-issued certificate to the CPF record and detect that it is not authorized by the domain owner.

6.5. Transparent SSL Interception

A growing number of organizations are implementing network security solutions including data-loss prevention, intrusion detection/prevention, web filtering, antivirus scanning, etc. In some cases these countermeasures are configured to transparently intercept encrypted traffic in order to scan for legitimate purposes. This is accomplished via man-in-the-middle interception using an internal PKI with a custom certificate authority that each client device is configured to trust. When using CPF, client applications will see all certificates signed by a proxy appliance, which will not be in the CPF record for the associated domain. While it’s generally good for CPF to act against unauthorized certificates, this situation presents challenges.

When client applications implement CPF, it may be beneficial to include a local setting which allows specific CAs to override CPF. Also, the software/appliances transparently proxying connections should support the CPF protocol and have logic to protect client applications.

Alternatively, organizations may be able to reconfigure the transparent device to act as an explicit proxy. This resolves the issue by forcing the client application to establish connections directly with the proxy device.

6.6. Self-Signed Certificate Authorization

Certificate authorities are an essential component of PKI, and provide the valuable function of validating a certificate requestor's legitimacy. CPF is intended to provide an additional layer of protection, and must not be viewed as a replacement for CAs. If a certificate is not signed by a trusted issuer, then CPF must not be evaluated. Rather, the application should enforce a specific policy for these conditions on a global and/or case-by-case basis.

If CPF were allowed to override CA trust, it would introduce a dangerous condition where a malicious individual could manipulate both network connections and trusts via DNS (either by compromise or MITM). Consider the situation where an organization's DNS server is compromised – the attacker could redirect network traffic to a controlled device with a self-signed certificate. In addition, the attacker could add a CPF record for the self-signed cert, allowing it to appear as trusted by the client application. By allowing this functionality, it would provide an easy mechanism for attackers to fake PKI security.

6.7. Complimenting CRL/OCSP

Certificate Revocation Lists (CRLs) are used to publish certificates which have been revoked by the certificate authority and are no longer considered trusted. Online Certificate Status Protocol (OCSP) is a newer method for delivering similar information on a certificate-by-certificate basis, offering better performance and faster distribution than CRLs.

CPF offers the ability to block certificates matching specific signatures using the "fail" qualifier. This functionality has been included as a secondary method for domain owners to block illegitimate or compromised certificates. This mechanism is intended to compliment CRLs/OCSP, not to replace them. Certificates MUST still be revoked through a certificate authority, and applications MUST continue to fetch CRLs and OCSP information.

7. Acknowledgements

The CPF protocol was modeled to be very similar to Sender Policy Framework (SPF) [RFC4408]. SPF has been widely accepted as a method of authorizing mail servers, and has led to notable improvements in email security. Similarly, the intention of CPF is to accomplish the same goal for certificates and PKI security.

8. References

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1034, November 1987.

[RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.

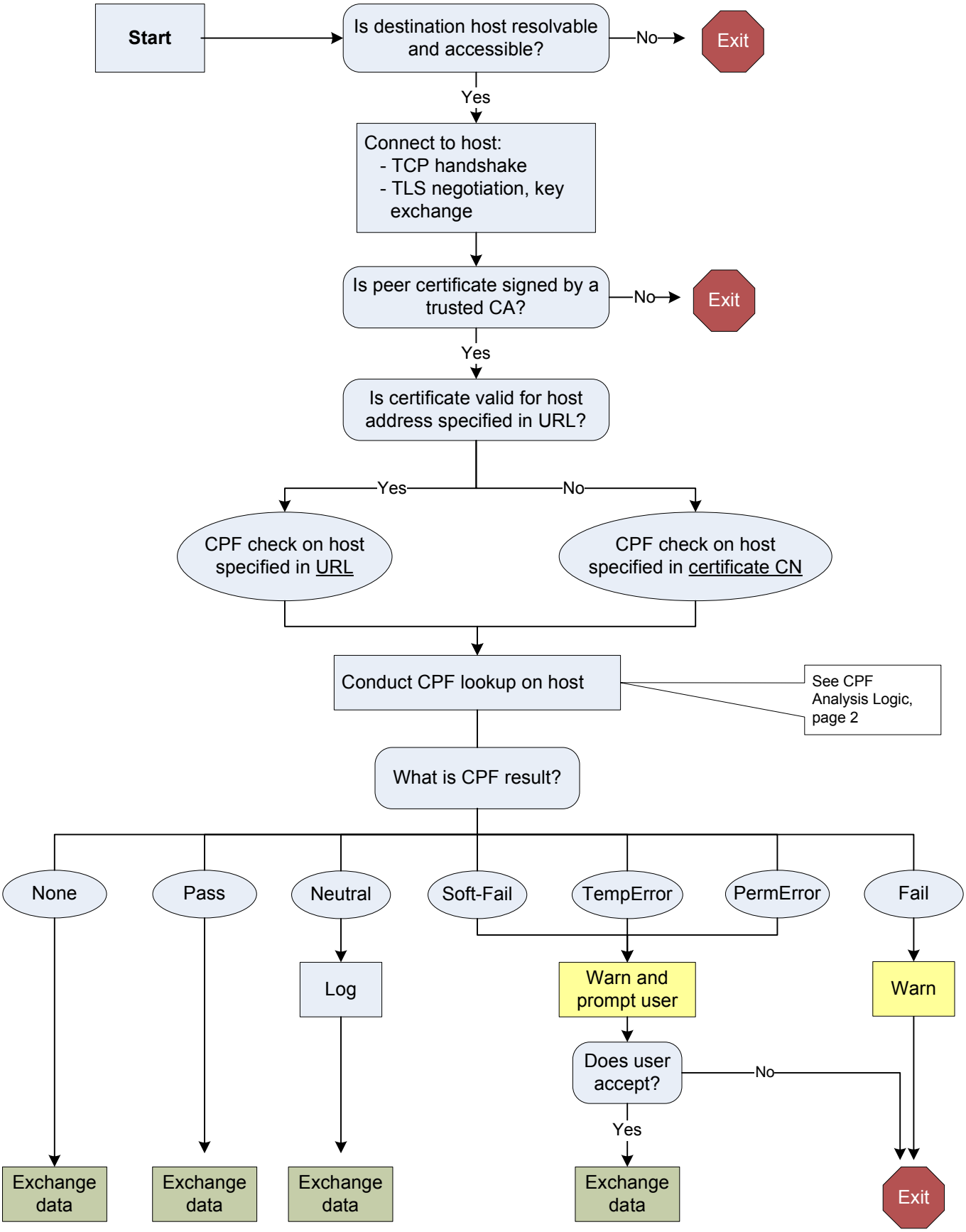
[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4408] Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. M. Wong, W. Schlitt. April 2006.

Logic Diagram for Proof-of-Concept HTTPS Browser

CPF-Compatible Browser HTTPS Logic

Author: Matthew Lidestri
Version: 1.1
Date: 09/13/2011



CPF Analysis Logic

Author: Matthew Lidestri
Version: 1.1
Date: 09/13/2011