DNSCHK: Free Malicious Download Detection with Highly-Available Distributed Systems

Anonymous Author(s)

Abstract

Downloading resources from the internet comes with considerable risk due to the general inability of the end user to verify the integrity of the resource they received. An adversary could tamper with the resource en route to its destination or even compromise the server that hosts the resource, as was the case with the 2016 breach of the Avast distribution servers that provided downloads of the popular software suite CCleaner. The de facto standard method for addressing this risk is with the use of *checksums*signatures generated by cryptographic hashing functions to verify a resources integritycoupled with some secure transport medium like TLS/HTTPS. This method is problematic for a whole host of reasons, the foremost being the fact that the clear majority of end users will not be burdened with manually calculating a resource's checksum. Even if they do, said user must search for the corresponding ”correct” checksum to verify their calculation. If that verification fails, the user is then expected to ”do the right thing” in context.

With this research, we explore a novel method of verifying the integrity of resources downloaded over the internet with two key concerns: a) it is automatic, *i.e.,* no additional interaction is required from the end user during standard usage and b) configuring the validation method is simple for service administrators and system operators to integrate and deploy. Hence, we propose DNSCHK, a novel automated resource validation method that is transparent to end users and simple for administrators to deploy. We implement DNSCHK as a proof-of-concept Google Chrome extension as well as a patch to the FileZilla FTP client. We evaluate the security, scalability, and performance of the DNSCHK approach and provide a publicly accessible demonstration of its utility via a patched HotCRP instance.

1 Introduction

Downloading resources over the internet is a remarkably simple and painless process for application developers and end users alike. The user (via their browser) requests a server resource at some URL. The server responds with the resource. The browser completes downloading the resource. Unfortunately, downloading content over the internet can be risky.

[TODO: (the rest; an expository summarization of research)]

Discuss ”free”: no interface changes, no addition to resource download time, no additional burden on the end user (qualified statement).

DNSSEC: Eastlake, D., ”Domain Name System Security Extensions”, RFC 2535, March 1999.

In summary, our primary contributions are:

- We propose to **[TODO: the DNSCHK approach using DNS/DHT]** ...
- We present our prototype DNSCHK implementations for Google Chrome and FileZilla. To the best of our knowledge, this is the *first* system having such capabilities. Further, we release the DNSCHK solution to the community as open source software (footnote with link).
- We carefully and extensively evaluate the security, scalability, and performance of our automated defense against resource corruption to demonstrate the effectiveness and high practicality of the DNSCHK approach. We further provide a publicly accessible demonstration of DNSCHK's utility via a patched HotCRP instance (footnote with link).

2 Background

In this section we ...

[TODO: Use the language of IETF RFC3552 to describe active attack]

2.1 Current Detection and Prevention Methods

There are several. Blah blah.

Anti-Malware Software. (what it is, why it fails; also talk about manual scanning of files for viruses)

HTTPS / Encrypted Channel. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, ”HyperText Transfer Protocol”, RFC 2616, June 1999.

Dierks, T. and C. Allen, ”The TLS Protocol Version 1.0”, RFC 2246, January 1999.

Rescorla, E., ”HTTP over TLS”, RFC 2818, May 2000.

Blake-Wilson, S., Nystrom, M., Hopwood, D. and J. Mikkelsen, ”Transport Layer Security (TLS) Extensions”, RFC 3546, May 2003.

Browser-based Heuristics and Blacklists. (what it is, why it fails)

Checksums. (what it is, why it fails)

Public Key Infrastructure. Including DANE and similar technologies.

2.2 Motivation: Case Studies.

Blurb about case studies.

Case 1: x. Explain

Case 2: x. Explain

Case 3: x. Explain

Case 4: x. Explain

2.3 "Free" Highly-Available Distributed Systems.

There might have been DNS size concerns, but EDNS/EDNS0 assuages those fears.

Others are considering this as well, such as securitytxt draft (cite). A widely deployed example is DKIM. (cite).

tools.ietf.org/html/rfc6891

3 The DNSCHK Approach

In this section we ...

[TODO: (describe generalized solution system using any sort of distributed highly-available key-value store that exists or could exist (like dns))]

Go over algorithms!

4 Implementations

In this section we ...

4.1 DNSCHK: Google Chrome Extension

[TODO: (describe implementation details; works with DNS or DHT and is published to Chrome store; no interface changes!–i.e. downloads work exactly the same with or without the extension; users still have to confirm/deny suspicious judgements, but they're rare occurrences)]

Reference hotcrp demo but leave the description for the evaluation.

uggest chrome download API improvement ability to mark downloads dangerous.

4.2 DNSCHK: FileZilla (FTP) Patch

[TODO: (describe implementation details; minor interface change if the download is judged unsafe or suspicious, requires user to confirm/deny download)]

5 Evaluation

The primary goal of any DNSCHK implementation is to alert end-users when the resource they've downloaded is something other than what they were expecting. We tested the effectiveness of our approach using the DNSCHK extension for Google Chrome, a real-world deployment of HotCRP, and a random sampling of papers published in previous Usenix proceedings.

5.0.1 Threat Model and Assumptions

tools.ietf.org/html/rfc3552 section-5

Include threat model for Chrome extension and FTP patch.

5.1 Real-World Resource Corruption Detection with Google Chrome and HotCRP

It also seems from ACME that HTTP challenges are good enough of a proof to issue TLS certificates, so why not good enough for checksums? Threat model of ACME thoroughly goes through this, cite bit.ly/2PYS30a

5.2 Overhead

Additional Download Latency, Additional Network Load, Runtime overhead, etc. All nixed.

6 Discussion

In this section we ...

6.1 Related Work

Short blurb about how other research exists.

DNS-Based Authentication of Named Entities (DANE). The DNS-Based Authentication of Named Entities (DANE) specification [RFC6698] introduces the DNS "TLSA" resource record (RR) type ("TLSA" is not an acronym). TLSA records associate a certificate or a public key of an end-entity or a trusted issuing authority with the corresponding Transport Layer Security (TLS) [RFC5246] or Datagram Transport Layer Security (DTLS) [RFC6347] transport endpoint. DANE relies on the DNS Security Extensions (DNSSEC) [RFC4033]. DANE TLSA records validated by DNSSEC can be used to augment or replace the use of trusted public Certification Authorities (CAs).

DANE + OpenPGP: tools.ietf.org/html/rfc7929; Different concern. tools.ietf.org/html/rfc7671

PGP. Notoriously hard to use. Similar problems as checksums and DNSSEC in that its hard to use correctly. Solves a fundamentally different problem in a different domain (email).

6.2 Deployment and Scalability

Discuss envisioned deployment strategies for resource providers.

Can this be scaled? Yes it can. What are the practical limits? EDNS0 means it ain't DNS size, though packet fragmentation is still a concern. How about max record length? Maximum number of records? A service could have thousands or millions of files it serves! Can DNS handle that? DHT failover is still a solution anyway.

6.3 Limitations

Short blurb on limitations.

DNSSEC Adoption is Slow. DNSSEC is hard to use and harder to use correctly. It does not make the DNS network, i.e. properly configured DNS servers protected with DNSSEC, any more vulnerable to reflection (cite Neustar) and amplification (cite ieeexplore.ieee.org/document/4159821) attacks than it already is as a UDP-based content service (cite ss.vix.su/ vixie/isc-tn-2012-1.txt, us-cert.gov/ncas/alerts/TA14-017A) but does arguably make services significantly more fragile because DNSSEC is hard to get right.

The metrics of signed DNSSEC zones are not easy to come by for the entire Internet (apnic). DNSSEC adoption across is small and slow. Worldwide, less than 14 percent of DNS requests have DNSSEC validated by the resolver (cite apnic) but thanks to community initiatives is on the rise (cite blog.cloudflare.com/automatically-provision-and-maintain-dnssec) (use graph as figure bit.ly/2zSR7A6).

From SO: DNSSEC does have risks! It's hard to use, and harder to use correctly. Often it requires a new work flow for zone data changes, registrar management, installation of new server instances. All of that has to be tested and documented, and whenever something breaks that's related to DNS, the DNSSEC technology must be investigated as a possible cause. And the end result if you do everything right will be that, as a zone signer, your own online content and systems will be more fragile to your customers. As a far-end server operator, the result will be, that everyone else's content and systems will be more fragile to you. These risks are often seen to outweigh the benefits, since the only benefit is to protect DNS data from in-flight modification or substitution. That attack is so rare as to not be worth all this effort. We all hope DNSSEC becomes ubiquitous some day, because of the new applications it will enable. But the truth is that today, DNSSEC is all cost, no benefit, and with high risks.

The overwhelming majority of domain name zone administrators appear to be just not aware of DNSSEC, or, even if they want to sign their zone, they cannot publish a signed zone because of limitations in the service provided by the registrar, or if they are aware and could sign their zone, then they dont appear to judge that the perceived benefit of DNSSEC-signing their zone adequately offsets the cost of maintaining the signed zone.

DNS-Specific Protocol Limitations. Whether we created our own DNS resource record type or settled for a well-known TXT record, there are other DNS records that could be far larger. Resource records like CERT, IPSECKEY, OPENPGPKEY, and TLSA (with a full certificate) may hold several kilobytes of data, far more than any compliant security.txt file ever would.

I do hope that all the major DNS server operators have implemented DNS-over-TCP or some sort of rate limiting by now, but I feel that's out of the scope of this project. All we can do is urge any company running a vulnerable DNS server to quickly create a security.txt file.

DNS data can handle a lot more then the amount of data needed here. And amplification protection can be done simply by using TCP or insisting on DNS-COOKIES for these answers. This is a generic DNS problem is solved by DNS people and there is no need for additional requirements here.

6.4 Future Work

(adapt wiki entry)

7 Conclusion

[TODO: (summarize intro, contributions, evaluation, and discussion tidbits)]