


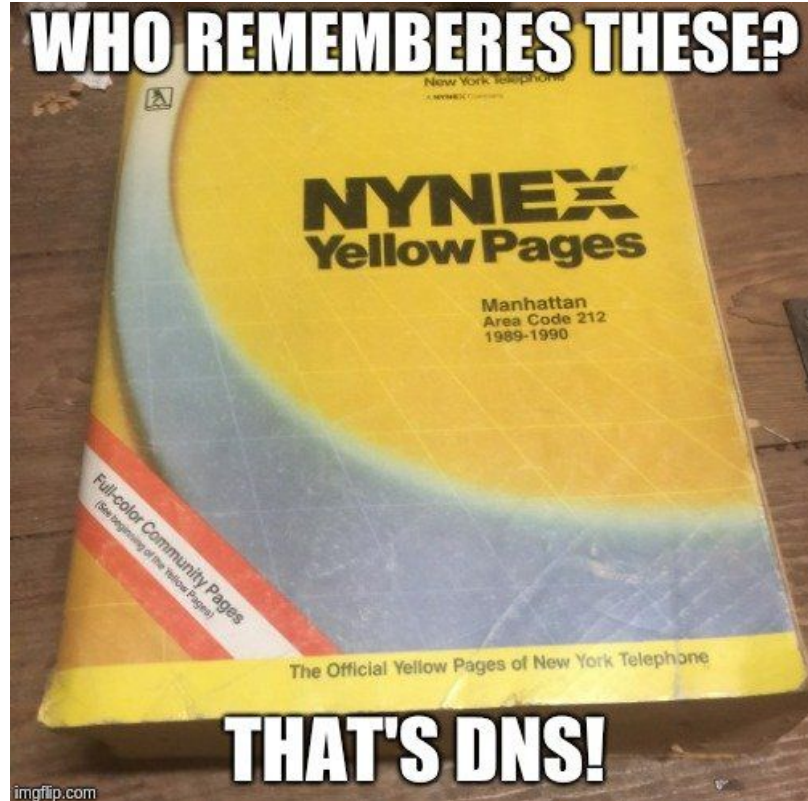
# Security in DNS

What it is, and why it matters ...

# Who am I?

- Michael Casadevall, also known as FOSSFIREfighter, or NCommander
  - Fellow for ICANN63 and ICANN65
    - Member of the New GTLD Subsequent Procedures Policy Development Process
    - Interest in DNSSEC, IDNs, EAI, and other security and accessibility issues
  - Alumni of Canonical for six years developing Ubuntu Linux
  - Experienced with WebPKI, DNSSEC, and attacks on these protocols
  - Independent Freelancer and Researcher
  - Twitter: <https://twitter.com/FOSSfirefighter>
  - GitHub: <http://github.com/NCommander>
  - IRC: NCommander on Freenode
- 

# What is DNS?



# What is DNS (Part 2)

- Short for Domain Name System
- It is core internet protocol, responsible in converting IP addresses to names like google.com, twitter.com or defcon201.org
  - Without it, we'd be stuck remembering that 198.105.254.228 is the DC201 website
- DNS is a very old protocol, and used as part of ARPANET
  - Modern incarnation standardized in 1987, predating the birth of the modern internet.
  - Hierarchical in nature with all entries stemming from a root zone.
  - One of the first mostly of decentralized protocols on the Internet
- Used to publish information about Internet hosts




# DNS Addresses Are Like Phone Numbers

- Let's take a typical phone number and break it down: +1-201-555-1212
  - + (or 011) is the prefix, indicating the country this phone number is from. It's typically omitted unless dialing internationally
  - 1 is the country code, United States in this case
  - 201 is an area code, a geographical region within a country
  - 555 is the exchange area, subdividing area codes into various blocks
  - 1212 is the subscribe number
- DNS Addresses work the same way, let's look.



# Anatomy of a DNS Address

- Let's take an example DNS name: dns-talk.defcon201.org.
    - DNS addresses are processed right to left, starting with the period. Like +, it's typically omitted
    - . represents the Internet Root Zone, the basis of DNS and the effective core of the Internet.
    - org is a Top-Level Domain, one of hundreds that group and organize names within the domain name system
    - defcon201 is a second level domain, representing a single site within org
    - dns-talk is a third level domain, representing a sub-site within defcon201
  - The DNS system is hierarchical, and works through a system of delegation; there is no centralized database of all domain names throughout the world.
  - We'll look at the delegation system later in this presentation.
- 


# Example Zone File

```
$TTL 3600
$ORIGIN com.
example      IN      SOA      ns1.example.com. example.com. (
                2018082601      ; Serial
                5M                ; Refresh
                4H                ; Retry
                4W                ; Expire
                30M               ; default_ttl
            )
NS ns1.example.com.

A      128.66.0.1
AAAA   2001:db8::f03c:91ff:febb:64e9
TXT    "v=spf1 a mx ptr ip4:128.66.0.1 ip6:2001:db8::f03c:91ff:febb:64e9 -all"
MX     0      mail.example.com.
```



# Common Types of DNS Records

- SOA
    - Statement of Authority
    - Information about the authoritative data within zone as a whole
  - A
    - Represents a name to number translation for IPv4
  - AAAA
    - Pronounced quad-A, represents a name to number translation for IPv6
  - MX
    - Mail e(X)change, what systems process incoming mail for this domain name
  - TXT
    - TeXT record, arbitrary data used by some applications such as SPF
- 



# Common Types of DNS Records

- CNAME
  - Canonical NAME - An alias to an A record
- NS
  - Name Server - What nameservers control this zone
- PTR
  - Pointer - Maps a name to an IP address for reverse look up (think \*69)
- CAA
  - Certificate Authority Authorization - What CAs are allowed to sign for this domain




# DNS Lookups In Practice

DNS lookups work through a system of delegation, starting from the root zone, and a network of interconnected NS records. The easiest way to show this in practice is to show a zone lookup from the root zone.

First, we'll show getting the A record for defcon201.org from Cloudflare's 1.1.1.1 resolver directly.

Then we'll show how it derived that answer from Internet Root Zone, request the NS record for org, and follow the chain until we get the A records for defcon201.org



# Querying CloudFlare for the A record

```
$ dig @1.1.1.1 defcon201.org IN A +noall +answer
; <<>> DiG 9.11.3-lubuntu1.2-Ubuntu <<>> @1.1.1.1 defcon201.org IN A +noall +answer
; (1 server found)
;; global options: +cmd
defcon201.org.      3600 IN    A      138.197.87.22
```



# Querying The Root Zone

```
$ dig @a.root-servers.net org NS IN +noall +authority
```

```
; <<>> DiG 9.11.3-1ubuntu1.2-Ubuntu <<>> @a.root-servers.net org NS IN +noall +authority  
; (2 servers found)  
;; global options: +cmd  
org.          172800      IN      NS      d0.org.afilias-nst.org.  
org.          172800      IN      NS      a0.org.afilias-nst.info.  
org.          172800      IN      NS      c0.org.afilias-nst.info.  
org.          172800      IN      NS      a2.org.afilias-nst.info.  
org.          172800      IN      NS      b0.org.afilias-nst.org.  
org.          172800      IN      NS      b2.org.afilias-nst.org.
```



# Querying org.

```
$ dig @d0.org.afilias-nst.org. defcon201.org IN NS +noall +authority
```

```
; <<>> DiG 9.11.3-1ubuntu1.2-Ubuntu <<>> @d0.org.afilias-nst.org. defcon201.org IN NS  
+noall +authority  
; (2 servers found)  
;; global options: +cmd  
defcon201.org.      86400      IN      NS      dns1.registrar-servers.com.  
defcon201.org.      86400      IN      NS      dns2.registrar-servers.com.
```



# Querying defcon201.org's A record

```
$ dig @dns1.registrar-servers.com. defcon201.org IN A +noall +answer
```

```
; <<>> DiG 9.11.3-1ubuntu1.2-Ubuntu <<>> @dns1.registrar-servers.com. defcon201.org IN A  
+noall +answer  
; (2 servers found)  
;; global options: +cmd  
defcon201.org.      3600 IN    A       138.197.87.22
```




# An Insecure Distributed Information Database

As that example shows, DNS is both vast and distributed. It's also lacking one notable feature out of the box: integrity and security.

It's possible for any server to monitor DNS traffic and see what websites you're visiting, even in light of use of secure sockets.

Furthermore, any DNS server can simply lie, and return bogus information. Many ISPs such as Spectrum do this and return "search results" in cases where NXDOMAIN should have been sent. This wrecks havoc with software that expects proper operation of their DNS software.



# DNS Hijacking

Because DNS data is neither encrypted nor authenticated by the end-user (DNSSEC does **NOT** fix this), it is trivially easy to subvert DNS queries and return invalid information. Furthermore, as standard DNS operates over UDP, it's also possible to perform packet injection and simply return wrong information.

The two most common ways of subverting DNS traffic is a man-in-the-middle attack, and by DNS cache poisoning. We'll covering both of these.






# Cache Poisoning

To reduce load on the DNS system, records are cached by recursive resolvers for a period known as the Time-To-Live, which is set globally in the SOA record, or on a per record basis.

If an attacker can manipulate responses from upstream DNS servers (via BGP hijack, packet injection, etc), they can cause the recursive resolver to cache invalid data, and redirect clients to another DNS target.

DNSSEC was designed primarily to help combat cache poisoning attacks by making DNS records authenticatable.



# DNSSEC Doesn't Actually Help

DNSSEC unfortunately suffers from a few fatal flaws. In the context of preventing caching attacks, low deployment on the global Internet means most targets are hijackable without any additional effort.

Even when DNSSEC is deployed, it's prone to both stripping and denial-of-service attacks; DNSSEC records are typically too big to include in DNS-over-UDP; by forcing TCP resets, a recursive resolver can be forced to downgrade to UDP and not get signed record information.



# It Gets Worse

DNSSEC itself is also prone to stripping attacks; there is no standard that indicates a DNS record should be or must be signed. Although DNS resolvers can assume the root zone is signed, an attacker can simply delete the DS and RRSIG records in flight if they're capable of doing a TCP/IP man-in-the-middle attack.

Furthermore, DNSSEC doesn't cover NS delegation which can allow for sidechannel attacks even without stripping DNSSEC data. Finally, and most damning, DNSSEC doesn't extend to the last mile; the client never sees signed DNS records directly.

It's entirely dependent on the recursive resolver.



# DNS Man In The Middle

Furthermore, DNS is also trivial to subvert if an attacker can control the recursive resolver or its upstream servers. In the United States, several ISPs such as Verizon and Spectrum hijack NXDOMAIN DNS requests to return a “search results” page.

While this is relatively begin in of itself, ISPs can also use DNS to filter out ‘undesirable’ websites, or manipulate traffic at will with little hope of detection. Since it’s possible to simply capture all traffic on port 53, ISPs can simply hijack all outbound DNS requests even if a client attempts to use a third-party resolver.



# Bringing Security To DNS

The first real efforts to bring security came in the form of DNSSEC, which adds the ability to sign and secure records. DNSSEC is currently the most commonly deployed security system, but it's complex and doesn't solve the problem with "the last mile".

As other DNS security measures are built ontop of DNSSEC, let's take a moment to look at review how it works.



# Terminology

- KSK
  - Key Signing Key - this can be considered the master key to a given zone
- ZSK
  - Zone Signing Key - the key used to sign a zone; it's use is optional but its signed by the KSK to allow a level of PKI control within the DNSSEC ecosystem
- RRSet
  - Resource Record Set - The collective whole of a given type of resource records; i.e. all the A records within a set



# How It Works

DNSSEC works by forming a signed chain of trust from the Internet Root Zone to each end point through a system of designated signers. The root zone was signed in 2010, and most of the TLDs are now signed making real world deployment possible.

DNSSEC was primarily implemented by adding new resource record type to DNS to contained signed data and information.



# DNSSEC RRTypes

- RRSIG
  - Resource Record Signature - A signed record of all the collected RRTypes
- DNSKEY
  - Holds the public key for the zone
- DS
  - Designated Signer - Holds the public key for the next zone in the delegation chain
- NSEC/NSEC3
  - Next Secure Record (v3), affirms that a domain doesn't exist by signed reply






# Chain of Trust

Trust is established from the root zone via a hardcoded KSK (which was recently rolled over). This forms the basis of the DNSSEC system.

The root zone publishes its own DNSKEY record, and DS records of the top level domains that are signed with their own private keys. This creates the first link in the chain.

Second level domains, if signed, have their DS records added to their top level domain, and the process repeats ad-infinitum. Let's walk through a signed DNS zone, [soylentnews.org](https://soylentnews.org).



# Root Zone DNSSEC

```
$ dig @a.root-servers.net org NS +noall +authority +answer +dnssec
org.          172800      IN   NS     a0.org.afiliast.info.
[...]
org.          86400      IN   DS     9795 7 1 364DFAB3DAF254CAB477B5675B10766DDAA24982
org.          86400      IN   DS     9795 7 2
3922B31B6F3A4EA92B19EB7B52120F031FD8E05FF0B03BAFCF9F891B FE7FF8E5
org.          86400      IN   RRSIG  DS 8 1 86400 20181129050000 20181116040000 2134 .
1DLXk7k2GrdgxKJR5bruqm0b0JTRShQzQaDCKs+uI8Kf8W99hinWrf3h
WMx28DlRRD1zCAhMK9+67xjTdjCMw1w+d4FIGpmDtBqDI3u22VAvM/h/
TW1Z6NnPEwrlilgssT2QHDvFir4x/NPSNkgNtIMuy93hKqwdahYlas2N
XdTDAClqpfvHpSulJqEeAR3/uBVMg0wuHlYWQmrFbBKONwLbfPFESpxf
GmWvRTBjoJRXmK5237lhOST+0ifU+VMiy26RJgfpOFkhxs51ZgT9v4z2
CHDtFnFwtUf+GGAHjMK35VM0U2TFTBgs/AOArPOj4V7nDSIN7pXwYjhE QyYJOg==
```



# Notice the difference

When asked for a DNSSEC response, in addition to the NS records, we got DS records, and a RRSIG saying those DS records are signed, and can be verified against the root zone KSK.

An important element is that NS records are not signed by design. The theory is that you can verify the next level by comparing the DS record from the root zone to the DNSKEY of the next zone.

Let's continue walking the chain.



# TLD Signed Response For soylentnews.org

```
$ dig @a0.org.afiliias-nst.info soylentnews.org NS +noall +authority +answer +dnssec
```

```
soylentnews.org.      86400      IN      NS      ns1.linode.com.
[.]
soylentnews.org.      86400      IN      DS      60615 8 2
3BC43E5A590598F993D563A3D66936283350FEB80F3C3693C40FCD8A FF1872AF
soylentnews.org.      86400      IN      DS      64450 8 2
49245CF329C01E645C269582FFE73B4AAC8830AD6F6B3FE12B1616A7 63EB732A
soylentnews.org.      86400      IN      RRSIG   DS 7 2 86400 20181130152943 20181109142943
6368 org. A9vKMuj9poUsIf020kUX0vsdspUbyi7BiP6JHPm2oES+55opsbYFyquo
ZifHZU5t7YtScWlx9W7DYOj8xc6h7fJU9RL7OdvSJJt6L8bRLdQlVqEd
cng4hwXxp+9GPdQYvsf/19VqStmnQS+2PaTdtNb+/cM+hpT5qfjyU/s/ sLw=
```



# soylentnews.org signed A records

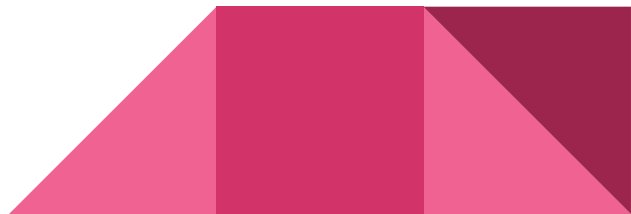
```
$ dig @ns1.linode.com soylentnews.org A +noall +authority +answer +dnssec
```

```
soylentnews.org.      300  IN   A     23.239.29.31
soylentnews.org.      300  IN   RRSIG A 8 2 300 20181204075551 20181104073513 60615
soylentnews.org.  Q/WuvkQR/yTJciSAzbl0FEvWOxQqQZ3468W0VJvFurchHkkOY01Zwb44D
/GZ7MfWcuEGWgq8l8a14QUL3kVWR2fg18/T7xQhX6+4Wnuj1ANQ1AzTD
Fm4Y9g2AU/fmYgPC4P/3Wp+ouR1IQhlVbWNqS0IzIgzYVixidYvJoDPK
J4JqJN57ZZy8zV/G6S2SDQERYwr3xF+8g1ujrz6JSLxCyEOs3C5+jVGB
/+U8N+IbYme0swM9ywnuuU5NR/JnKfSHBzMTYYjyeNFxmJQbC15M6/X6
ohD+as7Erm5ZfY8s54swJ9OW0E5foSX2lIrKZBX/4LdWcbwY2PvOX5VD AgwoXw==
```



# Walking The Chain

Given all the records we're interested in are signed and there's a chain of trust from the root zone to soylentnews.org, we can ask the 'dig' utility to walk the entire chain for us and make sure all keys are verified and OK.



# Chain Walking

```
mcasadevall@dawntreader:~$ dig @1.1.1.1 +sigchase +trusted-key=./root.keys soylentnews.org.  
A +noall  
;; RRset to chase:  
soylentnews.org.      286  IN    A      23.239.29.31  
  
[...]  
;; VERIFYING DS RRset for org. with DNSKEY:2134: success  
;; OK We found DNSKEY (or more) to validate the RRset  
;; Ok, find a Trusted Key in the DNSKEY RRset: 19036  
;; Ok, find a Trusted Key in the DNSKEY RRset: 20326  
;; VERIFYING DNSKEY RRset for . with DNSKEY:20326: success  
  
;; Ok this DNSKEY is a Trusted Key, DNSSEC validation is ok: SUCCESS
```



# What DNSSEC Provides

In short, DNSSEC provides a way to get authoritative signed information for DNS records, knowing that they're both valid and have not been tampered with.

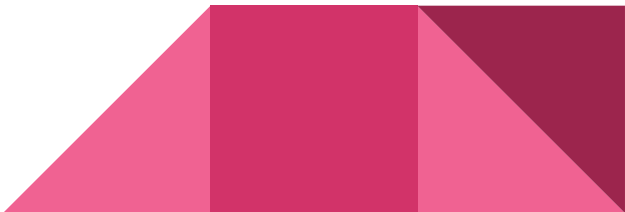
It also provides a mechanism of knowing who controls what and potentially detecting any record tampering; it's impossible to send invalid RRSIG records unless an attack has the private keys.

However, it is a system with flaws.






# Flaws in DNSSEC

- DNSSEC first requires that all domains manually sign their zones, and maintain up-to-date KSKs with their registrars. This is non-trivial and has greatly slowed deployment of DNSSEC on the public internet.
  - DNSSEC records are too big to send in a single UDP packet; TCP connections must be used to download the records which have a high setup/teardown cost compared to regular DNS
  - Information is still sent in the clear, an attacker can see what is being queried even if they can't tamper it, a form of information leakage.
  - It gets worse.
- 

# The Last Mile

- Doing signature validation requires walking the chain and contacting multiple servers in series to find if DNSSEC record data is good or bad. This is not a fast process, and can fail if a given nameserver is down or overloaded.
  - DNSSEC is prone to stripping attacks; there's nothing mandating that a zone is supposed to be signed.
  - Because of the amount of traffic, client resolvers built into Windows and Linux **do not** download DNSSEC data directly. They're dependent on the recursive resolver on their router or their ISP to validate DNSSEC data on their behalf. This is sent to the client as a single bit flag.
- 

# In Short

DNSSEC provides authentication for zones that have been signed, but clients do not (and may not be able to) check the zone data if DNS traffic is being tampered with.

It also provides no mechanism to prevent stripping attacks, or anything to protect the last mile from malicious data.

In short, while it is a sufficient base to prove data is correct, it's not good enough.


Fortunately, there has been some work to rectify part of this.



# DNS-over-TLS (DoT)

One of two new technologies is wrapping the existing DNS protocol around TLS. TLS is normally used to encrypt websites and is used by every website that shows a green “Secure” lock in major web browsers.

DNS-over-TLS provides the ability to encrypt the last mile of DNS to the recursive DNS resolver. This however presents two problems

- Many corporate firewalls block non HTTP/HTTPS traffic
  - DoT still relies on the standard DNS behavior of trusting the upstream server to do DNSSEC validation.
- 

# DNS-over-HTTPS (DoH)

In an effort to at least allow (more) secure DNS to work in virtually any environment, efforts have been made to allow DNS to work as a subprotocol with HTTPS; this would allow secure DNS in any situation where only basic web access is available.

Both DoT and DoH are available today from several public providers such as Cloudflare, and available in major DNS software. However, there are still major problems to be solved.



# Both These Protocols Are Flawed

DNS-over-TLS ties in the complex world of WebPKI and certificates to DNS. While it does prevent passive whistleblowing, realities of WebPKI prevent DoT from being affective as it can be


First off, it's impossible to get certificates for RFC1918 space, meaning any DoT-enabled resolver must have a public IP address to get a cert or use a local CA. In general, this means many users will likely only be able to use public resolvers over the Internet.



## DoT Flaws (Continued)

Furthermore, DoT has a bootstrapping issue; it's (generally) impossible to check the revocation status of a certificate without a DNS lookup. This also inherits all the flaws and problems of TLS revocation. In theory, OCSP stapling could be used, but it's unclear if any DoT client/server implements support for this today.

Secondly, DoT doesn't solve issues relating to cache poisoning or the recursive resolver flat out lying to clients. As it is simply DNS over TLS, DNSSEC data is not transmitted to the client for validation. As such it provides no additional authentication or security than standard DNS.



# DoH Adds More Issues

In addition to the issues with DNS-over-TLS, DNS-over-HTTPS also brings its own set of issues. It suffers from all the flaws previously described, and introduces a new one; specifically, it allows JavaScript code in the web browser to execute arbitrary DNS lookups.

This can be used as a type of hard-to-detect tracking token, or could even be used to probe internal networks should said networks be running a DoH service.

In short, DoH/DoT don't actually help add any realistic security to DNS data.





# Other Open Issues

As previously mentioned, the fact is there is no standard method of specifying if DNSSEC information should be available for a given domain. The creation of an equivalent to HSTS for DNS would help solve this issue but shares the same risks of trust on first use.

Secondly, no operating system today gathers and collects DNSSEC data directly over any protocol. There have been efforts to create “stapled DNSSEC” to TLS, but this work has currently stalled in drafting.



# Open Issues

Finally, both DoT and DoH still do server side validation of DNS records. This means that your DNS provider can silently change these records if they so choose.


In other words, while DNSSEC provides authentication, and DoT/DoH provide security, you don't have both at the same time, but it is possible to work around that.



# Client-Side Validation of DNSSEC

It is possible today to run a local DNS resolver that will handle retrieving RRSIG records. One such package is dnssec-trigger (<https://www.nlnetlabs.nl/projects/dnssec-trigger/about/>), available for Windows, Mac, and Linux

dnssec-trigger will download RRSIG records, run the validation and ensure known good results. Although it is still vulnerable to DNSSEC stripping attacks, it allows a high degree of confidence that DNS data being retrieved is known to be good and valid.



# Conclusions

Despite being a core Internet protocol, DNS can not be trusted to provide accurate information due to man in the middle attacks done by ISPs, as well as other forms of tampering such as DNS cache poisoning.

DoH and DoT provide encryption and prevent eavsdropping but do not prevent MitN attacks by DNS providers should they choose to provide them.

If you need secure DNS, your only option is to hope the servers you connect to provide DNSSEC information, your resolvers do not strip said information and use dnssec-trigger to validate it locally.



# Studying DNS Hijacking - DNSCatcher

In an effort to understand the full scope of DNS hijacking on the public Internet, I have started a project known as DNSCatcher (<https://github.com/NCommander/dnscatcher>).

DNSCatcher is written in Ada and designed to run as a DNS server, and then do A/B testing of data provided to it; for example, when given a request for apple.com, it will check both the local resolver and a lookup from the root zone.



# DNSSCatcher Theory

The intended result of DNSSCatcher is to note discrepancies between local DNS data and what is published in the root zone. Currently, Catcher is in early alpha and only exists as proof of concept. I'm actively seeking funding to develop and flesh it out further.

It is the intent of DNSSCatcher to provide a standard API for cross-checking DNS records from a client (or specialized tool like OONI Probe) to see if DNS data is accurate and provide a detailed cross-section of DNS hijacking across the Internet.




# Currently Implemented Functionality

Currently DNSCatcher supports the full DNS protocol over UDP; with TCP support being easy to add at this point; it cross-checked against a “known good” DNS server and checks for discrepancies which are logged.

It is intended to add support for storing in a database and caching DNS records properly as per TTLs to build comprehensive datasets of hijacking and where hijacked records point to.

Catcher is MIT-licensed and is free and open source software. Contributions welcome.



# Questions





# Contact Information

Email: [michael@casadevall.pro](mailto:michael@casadevall.pro)

Twitter: @fossfirefighter

IRC: NCommander on Freenode

I'm available for consulting and contracting work on issues relating to security, software development and more.

