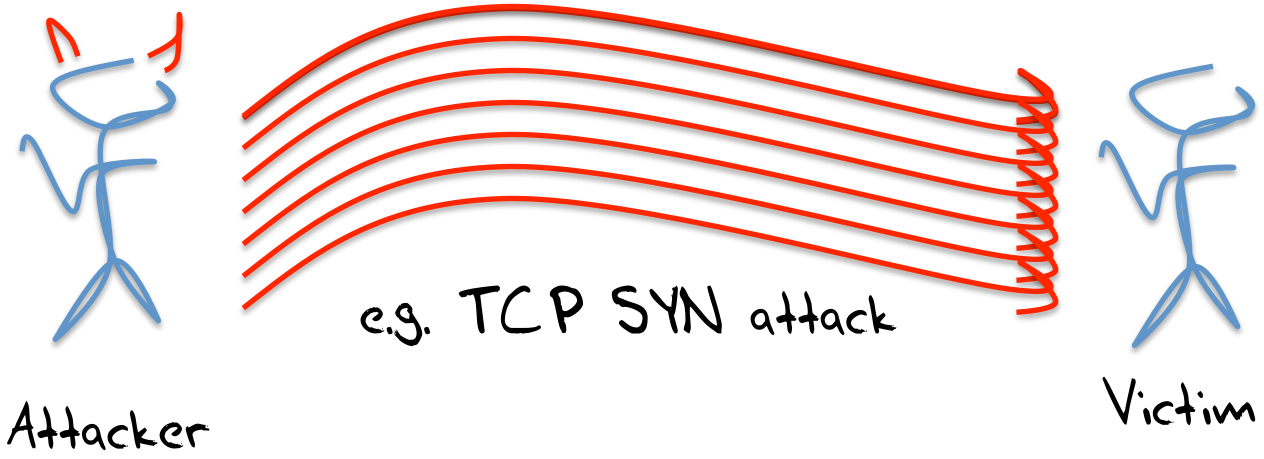
DNS, DNSSEC and DDOS

Geoff Huston
APNIC
February 2014

The Evolution of Evil

- It used to be that they sent evil packets to their chosen victim

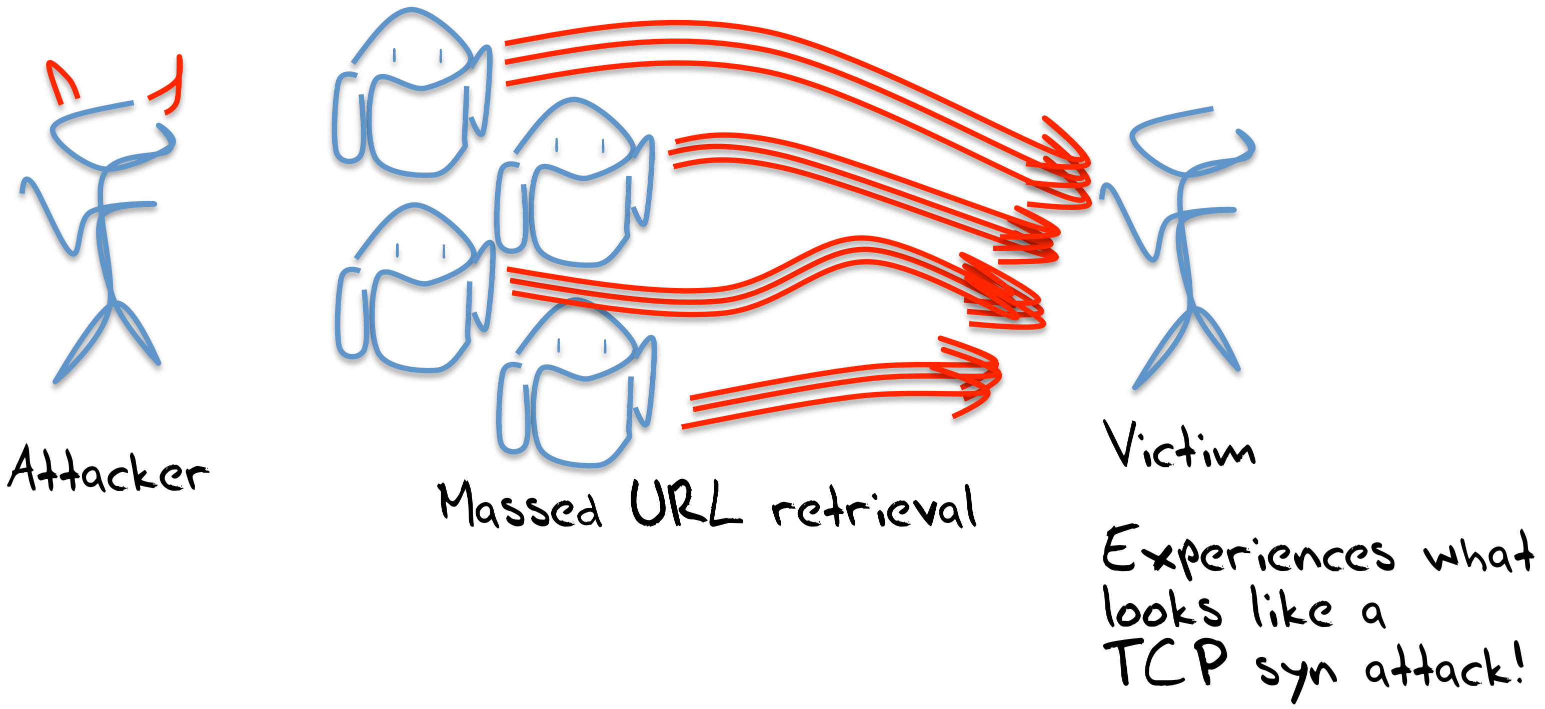
but this exposed the attacker, and limited the damage they could cause



The Evolution of Evil

- Then they enrolled a bot army to send evil

  which kept the attacker hidden and increased the damage leverage



The Evolution of Evil

- But now they co-opt the innocent to the evil cause, and use un-corrupted servers to launch the attack

  which hides the attacker(s) and uses the normal operation of servers to cause damage

UDP is a Fine Protocol

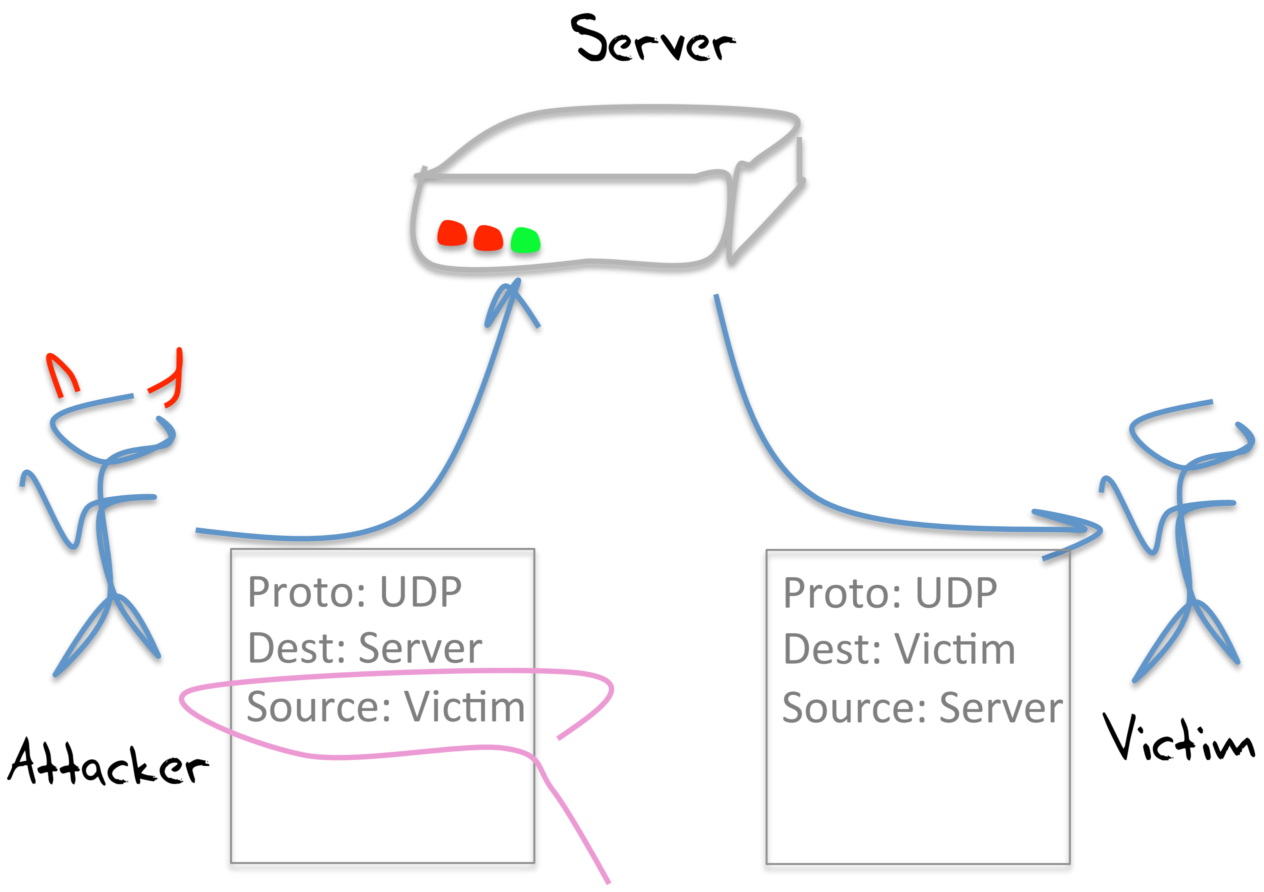
- UDP is used whenever you want a fast and highly efficient short transaction protocol
- Send a query to a server (one packet)
- And the server sends an answer (one packet)
- UDP works best when the question and the answer are small (<512 bytes), but can work on larger transactions *
- Although it's not as reliable as TCP

* The fine print – you'll need to magnify this to read it!
Some UDP applications use multiple UDP packets for large answers (e.g. NTP). Some rely of IP level fragmentation (e.g. DNS with EDNS0)
The problem with relying on fragmentation is firewall filtering and NATs (the trailing frags have no transport level header to assist in locating the NAT binding , as fragme
And the problem with multiple UDP packets is reliably reassembly is pushed into the application, which may not necessarily do this well!

UDP Mutation

- Unlike TCP there is no handshake between the two parties who are communicating
  - Send the server a UDP packet
  - The server flips the source and destination IP addresses and responds with a UDP packet
  - The server never checks the authenticity of the source address
- This allows a simply reflection attack…

UDP Reflection Attack



Note the fake source!

UDP and DDOS Reflection Attacks

This works "best" for a UDP-based service when

- The service is widely used
- Servers are commonplace
- Servers are poorly maintained (or unmaintained)
- Clients are not "qualified" by the server (i.e. anyone can pose a query to a server)
- The answer is far bigger than the question

Hmmmmm

What could that be?

DNS as an attack vector

- UDP-based query response service

  UDP is now almost ubiquitous for the DNS – EDNS0 wiped out the last vestiges of TCP fallback

- The service is widely used

  Everybody is a client of the DNS

- Servers are commonplace

  Resolvers are scattered all over the Internet

- Servers are poorly maintained (or unmaintained)

  There are some 30 million open resolvers

- Clients are not "qualified" by the server (i.e. anyone can pose a query to a server)

  DNS servers are by design promiscuous

  Many DNS resolvers are unintentionally promiscuous

- The answer is far bigger than the question

  Just ask the right DNS question!

DNS and DDOS

- DNS DDOS attacks are now very commonplace on today's Internet
- They can (and do) operate at sustained gigabit speeds
- They can use corrupted intermediaries to broaden the attack surface and further increase the query intensity
- And efforts to mitigate at the server tend to degrade the quality of the DNS service, as well as affecting the victim

DNS Queries and Responses

dig **A** isc.org - query size = 36 bytes

  149.20.64.69 – response size = 52 bytes

Conventional DNS queries and answers tend to be relatively poor attack amplifiers – in general the answer is not all that much larger than the question

But there are particular questions that generate more impressive answers…

The DNS ANY query

dig **ANY** isc.org – query size = 36 bytes

response size = 3,587 bytes

That’s more like it! In this case the response is 100x larger than the query

Blocking the ANY attack

- Modify the resolver not to respond to ANY queries in a meaningful way

```
$ dig ANY isc.org @8.8.8.8

; <<>> DiG 9.8.3-P1 <<>> ANY isc.org @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 6696
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;isc.org.                IN   ANY

;; Query time: 632 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun Feb 16 09:42:48 2014
;; MSG SIZE  rcvd: 25
```

The DNSSEC query

With DNSSEC, if the client requests DNSSEC information, then the additional records in the response contain crypto values

These crypto records can be quite large…

dig **+dnssec A** isc.org – query = 36 bytes

    149.20.64.69 – response = 1,619 bytes

That's an additional 1,567 bytes of crypto payload that has been provided by DNSSEC

Blocking DNSSEC DNS attacks

- Stop serving DNSSEC-signed zones
- And/or configure resolvers to turn off the DNSSEC-OK EDNS0 flag

But resolver-level query blocking defeats the entire purpose of DNSSEC!

So we need to look to other measures to mitigate this vulnerability

Possible responses

Drop “excessive” queries at the resolver

- collateral damage to the server and the served names
- can lead to cache poisoning attacks

Drop EDNS0 and revert to original DNS behaviour

No DNS UDP responses over 512 bytes

Requestor directed to use TCP instead

- Poor DNS resolution performance for all clients
- Can lead to server overload though increased TCP load

Maybe we can combine the two approaches

DNS Response Rate Limiting (RRL)

- Set a maximum rate that any requestor will be told the same answer

  note: this is not about the query – its about the response!

- Above this threshold either drop the query, or respond with the query and the truncated bit (TC) set ON
  - The ratio of candidate queries to TC responses is termed the "SLIP ratio"
    - Some folk say SLIP=2 is enough
    - Others seem to prefer SLIP=1

This will not eliminate the problem

As the attacker can broaden the attack plane across a large set of open recursive resolvers and not overload any single resolver to trigger its local RRL response

- And there are some ~30M such open resolvers
  http://openresolverproject.org

But it does increase the effort required to mount an attack based on DNS reflection, due to the added need to distribute the attack profile over a large set of resolvers

Attackers tend to exert no more effort than is strictly necessary to achieve the desired outcome, so increasing the effort needed to use the DNS to mount a reflection attack may well shift attention to other vulnerabilities, such as NTP

What you need to be naughty

To Do List

- ❑ Generate a list of open resolvers (zmap, for example is a good starting point)
- ❑ Write a simple script that sends a simple DNS query to an open resolver, with UDP source address spoofing
- ❑ Enlist a collection of coercible hosts to generate some 250,000 DNS queries per second across your list of open resolvers
- ❑ And the servers will respond with a 300Mbps DDOS stream!
- ❑ Rinse, repeat and multiply

What you need to be nice

- ❑ Add RRL to your DNS resolvers
- ❑ Clean up open DNS resolvers in your networks

  Limit queries on recursive resolvers to be sourced from your client cone, if you can

But…

- Being nice is not always possible
  - There is a significant volume of embedded DNS functionality in appliances and NAT-based consumerware
  - And enough of includes open DNS resolver functionality to be a problem that is not going to be “fixed” anytime soon

It’s not just the DNS

- NTP uses a UDP-based command and control channel over the same port as the time exchange (UDP port 123)
- And NTP servers are often installed with an open config
- The NTP monlist command is 220 bytes to send, and the response is a set of packets with a total volume of 46,800 bytes

What you need to be nice

- ❑Add RRL to your DNS resolvers
- ❑Clean up open DNS resolvers in your networks
  - Limit queries on recursive resolvers to come from your client cone, if you can
- ❑While you are at it, do the same filtering for NTP, and the `monlist` command in particular

In the longer term…

- Commonly used protocols that can generate large UDP responses are a long term problem
  - And DNSSEC will not cram into a 512 byte payload in the DNS
- So maybe it's the ability to pass through IP packets through the network with a false IP source address that is the basic problem, and just UDP exposes this problem to application level behaviour

What you need to be nice

- ❑Add RRL to your DNS resolvers
- ❑Clean up open DNS resolvers in your networks

  Limit queries on recursive resolvers to come from your client cone, if you can
- ❑While you are at it do the same filtering for NTP, and the `monlist` command in particular
- ❑Perform outbound traffic filtering to support source address validation: BCP38

Some Useful Resources

Open DNS resolvers:

http://openresolverproject.org

DNS RRL description

http://www.redbarn.org/dns/ratelimits

Sealing up NTP – a template for ntp.conf

http://www.team-cymru.org/ReadingRoom/Templates/secure-ntp-template.html

Open NTP servers

http://openntpproject.org

BCP 38

http://bcp38.info

BCP 38 tracking

http://spoofer.cmand.org//

Thanks!