INTERNET DRAFT – DNS SSQ (Secure Source Quench)

ietf-dnsop-ssq-00 P. Vixie
October 4, 2013 R. Karp

ABSTRACT

At the time of this writing, the most common method of Internet distributed denial of service attack (DDoS) is to forge the IP source address of DNS queries sent to high capacity domain name servers, who then reflect an amplified flow of large responses to the victim whose IP source address was forged. This attack is currently non-remediable because the attack traffic was not solicited by the victim and DNS has no provision for source quench controls. This document proposes a secure source quench control to be implemented by cooperating domain name servers and victim endpoints. The method described here is a defense against off-path attacks only.

INTRODUCTION

While admission control is theoretically practicable at the Internet edge [SAC004] [BCP38], such controls are rare enough that reflected amplified attacks in which a victim's IP source address is forged by an attacker are now distressingly common. From the point of view of the victim, the attack is an uncomfortably high volume of large unsolicited response datagrams which often causes service-affecting link, gateway, and server congestion. From the point of view of the reflecting amplifying DNS servers, the forged-source request flow appears merely to be a particularly busy requestor. Techniques exist for a responding DNS server to limit its rate of responses to a network, but these techniques would be inadequate in the face of a determined attacker who can specifically design smart forged-source request flows to overcome the rate limiting filters [RRL].

In this document we propose a method of DNS source quench whereby a victim endpoint can participate in a secure source quench handshake with a cooperating name server, to stop an attack flow instantly. Care is taken to prevent state overflow attacks by off-path attackers who can forge the IP source address of a victim or of a name server. This method can be deployed opportunistically and will do no harm when used asymmetrically between cooperating and non-cooperating endpoints. Any name server which implements this method is expected to become instantly less attractive as a reflecting amplifying DDoS amplifier. No cryptography is required, beyond a simple weak and fast one-way hash such as MD5. HMAC is not required. Pre-shared keys are not required.

The method described here is effective against off-path attacks utilizing spoofed IP-Source addresses. No attempt is made to prevent on-path attacks where packets between an Initiator and a Responder can be seen, modified, or inserted by an attacker.

METHOD

We propose three new operations to be added to the DNS protocol:

- DSQ (DNS Suspension Request), whereby an endpoint requests that a name server install a source quench rule for an endpoint's IP-Source address.

- DSRQ (DNS Suspension Request Acknowledgement), whereby a name server affirmatively answers a victim's DSQ request for a source quench rule.
- DSAC (DNS Suspension Final Acknowledgement), whereby an endpoint confirms a DSRQ request for a source quench rule for an endpoint's IP-Source address.
- DSNA (DNS Suspension Negative Acknowledgement), whereby a Responder informs an Initiator that it should restart the SSQ handshake process

We also propose a new Extended DNS (EDNS) option:

- DSAA (DNS is Suspended but please Answer Anyhow)

The data flow of a complete DNS source quench transaction is as follows:

1. Initiator -> Responder, DSQ<nonce>
2. Responder -> Initiator, DSRQ<nonce, proof, salt-generation>
3. Initiator -> Responder, DSAC<proof, salt-generation, time-limit>

Thereafter, any normal DNS operation such as QUERY, NOTIFY, or UPDATE performed by this Initiator against this Responder must use Extended DNS (EDNS) and must include the following option:

4. Initiator -> Responder, EDNS, DSAA<proof, salt-generation>

Thereafter, any normal DNS operation such as QUERY, NOTIFY, or UPDATE which has been certified with a DSAC option as described above can solicit a DSNA message indicating that the key is unknown or the proof is bad.

5. Responder -> Initiator, EDNS, RCODE=BADKEY, DSAA<proof, salt-generation>

The signals listed above are defined as follows:

DSQ<nonce> (and DSRQ<nonce> which copies it) is a pseudo-random 128-bit number chosen by the Initiator.

DSRQ<proof> (and DSAC<proof> which copies it) is computed by the Responder using a fast, low quality, non-crypto-authentic hash such as MD5, whose inputs are Salt-Generation, Salt-Value, and Initiator-IP-Source.

DSRQ<salt-generation> (and DSAC<salt-generation> which copies it) is a copy of the Salt-Generation value used to generate the associated DSRQ<proof> (or DSAC<proof>). This parameter is a 64 bit unsigned integer.

DSAC<time-limit> is a relative time interval, measured in minutes, giving the lifetime of the source quench rule being requested. This is expressed as an integer from 0 to 255, denoting time intervals from 1 to 256 minutes.

INTERNET DRAFT – DNS SSQ (Secure Source Quench)

ietf-dnsop-ssq-00 P. Vixie
October 4, 2013 R. Karp

INITIATOR BEHAVIOUR

An Initiator is an endpoint who is receiving unsolicited response flows. When an attack flow is detected, the Initiator transmits DSQ<nonce> to the purported source address of the attack flow. An Initiator keeps a bounded list of up to a few thousand outstanding DSQ transactions, expiring elements from this list after a few seconds if they do not complete successfully.

An initiator hearing a DSRQ<nonce, proof> message will check to see whether the Responder IP source address and the <nonce> match an outstanding DSQ transaction. If not, the DSRQ is silently discarded. Otherwise, the Initiator will answer with DSAC<proof, salt-generation, time-limit> and remove this DSQ transaction from its set of outstanding similar transactions. The initiator will record this <proof, salt-generation> tuple for subsequent transactions involving this Responder.

An Initiator who has completed a DNS SSQ handshake with a given Responder will include the resulting <proof> in an Extended DNS (EDNS) option DSAA on all subsequent normal DNS transactions such as QUERY, NOTIFY, or UPDATE sent to this Responder.

RESPONDER BEHAVIOUR

A Responder is any cooperating name server. The Responder will periodically calculate a Salt-Generation and a Salt-Value, where the periodicity of recalculation is a matter of a few minutes and need not be kept to a precise schedule. Each <Salt-Generation, Salt-Value> tuple must be saved for at least 256 minutes and should be persistent across Responder restarts. Salt-Generation is expected to be an absolute time stamp but may be any unique value including a simple increment. Salt-Value is expected to be a 128-bit pseudo-random number. These values are not exposed outside the Responder and need not be standardized.

Upon hearing a DSQ<nonce> message, the Responder will generate a <proof> by calculating the hash value of inputs Salt-Generation, Salt-Value, and Initiator-IP-Source, and transmit DSRQ<nonce, proof, salt-generation> where the <salt-generation> parameter is a copy of the current Salt-Generation value. No per-Initiator state will be saved in the Responder. The hash function used by an SSQ Responder is expected to be weak and fast, such as MD5, and need not be standardized.

Upon hearing a DSAC<proof, salt, time-limit> message, the Responder will compare the <proof> to the computed hash value of <salt, Salt-Value, and Initiator-IP-Source>. If no match is found, the DSAC message is silently discarded. Otherwise a source quench rule is installed for this Initiator (denoted by its IP source address) with the time limit given by DSAC<time-limit>.

Upon hearing a normal DNS request such as QUERY, NOTIFY, or UPDATE, the Initiator-IP-Source is compared against the Responder's list of source quench rules. If an unexpired source quench rule matches the Initiator-IP-Source, then the Responder will check for the presence of an Extended DNS (EDNS) DSAA<proof> option. If this option is not present, then the DNS request is silently discarded. If

the <salt-generation> value is not found in the set of <Salt-Generation, Salt-Value> tuples generated in the last 256 minutes of operation, then a rate limited response of DSNA<proof, salt> is sent to the Initiator and the request is discarded. Otherwise the DSAA<proof> value is compared to the hash values computed from the given Salt-Generation, the stored Salt-Value, and Initiator-IP-Source. If DSAA<proof> does not match this computed proof, then discarded rate limited DSNA<proof, salt> message is sent to the Initiator and the request is discarded.

IMPLEMENTATION NOTES

An Initiator may have new memory requirements equal to several thousand times the size of an outstanding DSQ transaction.

A Responder may have new persistent storage requirements equal to 256 minutes' worth of <Salt-Generation, Salt-Value> tuples.

ENCODING

DSQ can be OPCODE=QUENCH, RCODE=NOERROR, QR=0, QDCOUNT=1, QNAME=[Hex(nonce)], QTYPE=ANY, QCLASS=ANY, and all other sections empty.

DSRQ can be OPCODE=QUENCH, RCODE=NOERROR, QR=1, RA=0, QDCOUNT=1, QNAME=[Hex(proof) . Base64(nonce)], QTYPE=ANY, QCLASS=ANY, and all other sections empty.

DSAC can be OPCODE=QUENCH, RCODE=NOERROR, QR=1, RA=1, QDCOUNT=1, QNAME=[Hex(proof)], QTYPE=ANY, QCLASS=ANY, and all other sections empty.

DSNA can be RCODE=BADKEY, QR=1, EDNS, DSAA<proof, salt-generation>.

DSAA can be an Extended DNS (EDNS) option having two LV tuples as payload, <proof, salt-generation>

Hex(n) is a printable ASCII string made up of characters 0..9 and A..F, where quadets of the unsigned input <n> are converted starting with the first high order non-zero quadet, are converted to ASCII characters moving left to right.

FUTURE WORK

The method described here may also be applicable to other reflecting amplifying attack vectors such as SNMP, or to the UDP protocol itself using extensions to the ICMP protocols.

The method described here may also be deployable by an on-path defensive proxy who can detect incoming attacks, forge the victim's IP-Source, and intercept datagrams bound for the victim's IP-Destination.

SECURITY CONSIDERATIONS

INTERNET DRAFT – DNS SSQ (Secure Source Quench)

ietf-dnsop-ssq-00 P. Vixie
October 4, 2013 R. Karp

An on-path attacker who can see, modify, or insert packets between an Initiator and a Responder, can use the method described here to as an extremely low-cost denial of service vector. Initiator and Responder operators for whom this risk is palpable should utilize DNS Transaction Signatures [RFC2845] for hop-by-hop on-path defense, and DNS Security [DNSSEC] for end-to-end on-path defense.

REFERENCES

[BCP38] “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing”, also RFC 2827, P. Ferguson, D. Senie, May 2000.

[SAC004] “Securing the Edge”, ICANN SSAC, P. Vixie, October 2002.

[RRL] “DNS Response Rate Limiting”, P. Vixie, V. Schryver, April 2012.

[RFC2845] XXX

[DNSSEC] XXX

ACKNOWLEDGEMENTS

Paul Mockapetris and Vint Cerf helped review this document.

NOTES

PVM wonders what happens when each endpoint reboots.

Steve Crocker asked whether the victim should retransmit its SSQ periodically. Perhaps keeping state on real transactions so as to answer every unsolicited response would work.