IPSecME Working Group Y. Nir
Internet-Draft Check Point
Intended status: Standards Track October 27, 2014
Expires: April 30, 2015

Protecting Internet Key Exchange (IKE) Implementations from Distributed Denial of Service Attacks

draft-ietf-ipsecme-ddos-protection-00

Abstract

This document recommends implementation and configuration best practices for Internet-connected IPsec Responders, to allow them to resist Denial of Service and Distributed Denial of Service attacks. Additionally, the document introduces a new mechanism called "Client Puzzles" that help accomplish this task.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2015.

Copyright Notice

Table of Contents

1. Introduction

The IKE_SA_INIT Exchange described in section 1.2 of [RFC7296] involves the Initiator sending a single message. The Responder replies with a single message and also allocates memory for a structure called a half-open IKE SA (Security Association). This half-open SA is later authenticated in the IKE_AUTH Exchange, but if that IKE_AUTH request never comes, the half-open SA is kept for an unspecified amount of time. Depending on the algorithms used and implementation, such a half-open SA will use from around 100 bytes to several thousands bytes of memory.

This creates an easy attack vector against an Internet Key Exchange (IKE) Responder. Generating the Initial request is cheap, and sending multiple such requests can either cause the Responder to allocate too much resources and fail, or else if resource allocation is somehow throttled, legitimate Initiators would also be prevented

from setting up IKE SAs.

An obvious defense, which is described in Section 5, is limiting the number of half-open SAs opened by a single peer. However, since all that is required is a single packet, an attacker can use multiple spoofed source IP addresses.

Section 2.6 of RFC 7296 offers a mechanism to mitigate this DoS attack: the stateless cookie. When the server is under load, the Responder responds to the Initial request with a calculated "stateless cookie" - a value that can be re-calculated based on

values in the Initial request without storing Responder-side state. The Initiator is expected to repeat the Initial request, this time including the stateless cookie.

Attackers that have multiple source IP addresses with return routability, such as bot-nets can fill up a half-open SA table anyway. The cookie mechanism limits the amount of allocated state to the size of the bot-net, multiplied by the number of half-open SAs allowed for one peer address, multiplied by the amount of state allocated for each half-open SA. With typical values this can easily reach hundreds of megabytes.

The mechanism described in Section 3 adds a proof of work for the Initiator, by calculating a pre-image for a partial hash value. This sets an upper bound, determined by the attacker's CPU to the number of negotiations it can initiate in a unit of time.

1.1. Conventions Used in This Document

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].

2. The Vulnerability

If we break down what a responder has to do during an initial exchange, there are three stages:

1. When the Initial request arrives, the responder:
    * Generates or re-uses a D-H private part.
    * Generates a responder SPI.
    * Stores the private part and peer public part in a half-open SA database.
2. When the Authentication request arrives, the responder:
    * Derives the keys from the half-open SA.
    * Decrypts the request.
3. If the Authentication request decrypts properly:
    * Validates the certificate chain (if present) in the auth request.

Yes, there's a stage 4 where the responder actually creates Child SAs, but when talking about (D)DoS, we never get to this stage.

Stage #1 is pretty light on CPU power, but requires some storage, and

it's very light for the initiator as well. Stage #2 includes

private-key operations, so it's much heavier CPU-wise, but it releases the storage allocated in stage #1. Stage #3 includes a public key operation, and possibly many of them.

To attack such a server, an attacker can attempt to either exhaust memory or to exhaust CPU. Without any protection, the most efficient attack is to send multiple Initial requests and exhaust memory. This should be easy because those Initial requests are cheap.

There are obvious ways for the responder to protect itself even without changes to the protocol. It can reduce the time that an entry remains in the half-open SA database, and it can limit the amount of concurrent half-open SAs from a particular address or prefix. The attacker can overcome this by using spoofed source addresses.

The stateless cookie mechanism from section 2.6 of RFC 7296 prevents an attack with spoofed source addresses. This doesn't solve the issue, but it makes the limiting of half-open SAs by address or prefix work. Puzzles do the same thing only more of it. They make it harder for an attacker to reach the goal of getting a half-open SA. They don't have to be so hard that an attacker can't afford to solve them - it's enough that they increase the cost of a half-open SAs for the attacker.

Reducing the amount of time an abandoned half-open SA is kept attacks the issue from the other side. It reduces the value the attacker gets from managing to create a half-open SA. So if a half-open SA takes 1 KB and it's kept for 1 minute and the capacity is 60,000 half-open SAs, an attacker would need to create 1,000 half-open SAs per second. Reduce the retention time to 3 seconds, and the attacker needs to create 20,000 half-open SAs per second. Make each of those more expensive by introducing a puzzle, and you're likely to thwart an exhaustion attack against responder memory.

At this point, filling up the half-open SA database in no longer the most efficient DoS attack. The attacker has two ways to do better:

1. Go back to spoofed addresses and try to overwhelm the CPU that deals with generating cookies, or
2. Take the attack to the next level by also sending an Authentication request.

I don't think the first thing is something we can deal with at the IKE level. It's probably better left to Intrusion Prevention System (IPS) technology.

Sending an Authentication request is surprisingly cheap. It requires a proper IKE header with the correct IKE SPIs, and it requires a single encrypted payload. The content of the payload might as well be junk. The responder has to perform the relatively expensive key derivation, only to find that the Authentication request does not decrypt. Depending on the responder implementation, this can be repeated with the same half-open SA (if the responder does not delete the half-open SA following an unsuccessful decryption - see discussion in Section 4).

Here too, the number of half-open SAs that the attacker can achieve is crucial, because each one of them allows the attacker to waste some CPU time. So making it hard to make many half-open SAs is important.

A strategy against DDoS has to rely on at least 4 components:

1. Hardening the half-open SA database by reducing retention time.
2. Hardening the half-open SA database by rate-limiting single IPs/prefixes.
3. Guidance on what to do when an Authentication request fails to decrypt.
4. Increasing cost of half-open SA up to what is tolerable for legitimate clients.

Puzzles have their place as part of #4.

3. Puzzles

The puzzle introduced here extends the cookie mechanism from RFC 7296. It is loosely based on the proof-of-work technique used in BitCoins ([bitcoins]). Future versions of this document will have the exact bit structure of the notification payloads, but for now, I will only describe the semantics of the content.

A puzzle is sent to the Initiator in two cases:

- The Responder is so overloaded, than no half-open SAs are allowed to be created without the puzzle, or
- The Responder is not too loaded, but the rate-limiting in Section 5 prevents half-open SAs from being created with this particular peer address or prefix without first solving a puzzle.

When the Responder decides to send the challenge notification in response to a IKE_SA_INIT request, the notification includes two fields:

1. Cookie - this is calculated the same as in RFC 7296. As in RFC 7296, the process of generating the cookie is not specified, but this specification does assume that it is fixed-length, meaning that all cookies produced by a particular responder are of the same length.
2. Zero Bit Count. This is a number between 8 and 255 that represents the length of the zero-bit run at the end of the SHA-256 hash of the Cookie payload that the Initiator is to send.

   Since the mechanism is supposed to be stateless for the Responder, the same value is sent to all Initiators who are receiving this challenge. The values 0 and 1-8 are explicitly excluded, because the value zero is meaningless, and the values 1-8 create a puzzle that is too easy to solve to make any difference in mitigating DDoS attacks.

Upon receiving this challenge payload, the Initiator attempts to append different strings to the Cookie field from the challenge, and calculates the SHA-256 hash of the result. When a string is found such that the resulting hash has a sufficient number of trailing zero bits, that result is sent to the Responder in a Cookie notification, similar to what is described in RFC 7296. The difference is that the string in this Cookie notification is longer than the one transmitted.

When receiving a request with an extended Cookie, the Responder verifies two things:

- That the first bits of the transmitted cookie are indeed valid.
- That the hash of the transmitted cookie has a sufficient number of trailing zero bits.

Example 1: Suppose the calculated cookie is fdbcfa5a430d7201282358a2a034de0013cfe2ae (20 octets) and the required number of zero bits is 16. After successively trying a bunch of strings, the Initiator finds out that appending three octets: 022b3d yields a 23-octet string whose SHA-256 hash is 3b4bdf201105e059e09f65219021738b8f6a148896b2e1be2fdc726aeb6e0000. That has 17 trailing zero bits, so it is an acceptable cookie.

Example 2: Same cookie, but this time the required number of zero bits is 22. The first string to satisfy that requirement is 5c2880, which yields a hash with 23 trailing zero bits. Finding this requires 6,105,472 hashes.

Appended String	Last 24 Hex Hash Digits	# 0-bits	Time To Calculate
04	2817ae10f20f4e0b0739f5cc	2	0.000
06	e540cf315fff88c1c5f362a8	3	0.000
0d	8c459376268f747d7ed40da0	5	0.000
1c	398c49be1babe50576cdae40	6	0.000
00f0	3f523ad7c0e00252c51ad980	7	0.000
0182	e284296e2ffffa256bdfa800	11	0.000
235c	7dc74302dc8bd695821ab000	12	0.006
7186	a4411c3df3661eff1d574000	14	0.019
d836	498bcd04ab1ae0c2c3a08000	15	0.036
022b3d	96b2e1be2fdc726aeb6e0000	17	0.136
0aa679	620f48af85428996c1f00000	20	0.512
4ffbad	f9ba0ece854cd0fa88e00000	21	3.602
5c2880	d44e6467d8fc37723d800000	23	4.143
cdafe1	0d4058660c3e67be62000000	25	9.245
022bffc8	5f2d874764a71e2948000000	27	36.169
181ac92a	c3b5449fa1019b0580000000	31	255.076
a987978d	95a5673968a9b37a00000000	33	1309.519

Table 1: COOKIE=fdbcfa5a430d7201282358a2a034de0013cfe2ae

The figures above were obtained on a 2.4 GHz single core i5. Run times can be halved or quartered with multi-core code, but would be longer on mobile phone processors, even if those are multi-core as well. With these figures I believe that 20 bits is a reasonable choice for puzzle level difficulty for all Initiators, with 24 bits

acceptable for specific hosts/prefixes.

4. Retention Periods for Half-Open SAs

As a UDP-based protocol, IKEv2 has to deal with packet loss through retransmissions. Section 2.4 of RFC 7296 recommends "that messages be retransmitted at least a dozen times over a period of at least several minutes before giving up". Retransmission policies in practice wait at least one or two seconds before retransmitting for the first time.

Because of this, setting the timeout on a half-open SA too low will cause it to expire whenever even one IKE_AUTH request packet is lost.

When not under attack, the half-open SA timeout SHOULD be set high enough that the Initiator will have enough time to send multiple retransmissions, minimizing the chance of transient network congestion causing IKE failure.

When the system is under attack, as measured by the amount of half-open SAs, it makes sense to reduce this lifetime. The Responder should still allow enough time for the round-trip, enough time for the Initiator to derive the Diffie-Hellman shared value, and enough time to derive the IKE SA keys and the create the IKE_AUTH request. Two seconds is probably as low a value as can realistically be used.

It could make sense to assign a shorter value to half-open SAs originating from IP addresses or prefixes from which are considered suspect because of multiple concurrent half-open SAs.

5. Rate Limiting

Even with DDoS, the attacker has only a limited amount of nodes participating in the attack. By limiting the amount of half-open SAs that are allowed to exist concurrently with each such node, the total amount of half-open SAs is capped, as is the total amount of key derivations that the Responder is forced to complete.

In IPv4 it makes sense to limit the number of half-open SAs based on IP address. Most IPv4 nodes are either directly attached to the Internet using a routable address or are hidden behind a NAT device with a single IPv4 external address. IPv6 networks are currently a rarity, so we can only speculate on what their wide deployment will be like, but the current thinking is that ISP customers will be assigned whole subnets, so we don't expect the kind of NAT deployment that is common in IPv4. For this reason it makes sense to use a 64-bit prefix as the basis for rate limiting in IPv6.

The number of half-open SAs is easy to measure, but it is also worthwhile to measure the number of failed IKE_AUTH exchanges. If possible, both factors should be taken into account when deciding which IP address or prefix is considered suspicious.

There are two ways to rate-limit a peer address or prefix:

1. Hard Limit - where the number of half-open SAs is capped, and any further IKE_SA_INIT requests are rejected.
2. Soft Limit - where if a set number of half-open SAs exist for a particular address or prefix, any IKE_SA_INIT request will require solving a puzzle.

The advantage of the hard limit method is that it provides a hard cap on the amount of half-open SAs that the attacker is able to create. The downside is that it allows the attacker to block IKE initiation from small parts of the Internet. For example, if a certain purveyor

of beverages resembling coffee provides Internet connectivity to its customers through an IPv4 NAT device, a single malicious customer can

create enough half-open SAs to fill the quota for the NAT device external IP address. Legitimate Initiators on the same network will not be able to initiate IKE.

The advantage of a soft limit is that legitimate clients can always connect. The disadvantage is that a sufficiently resourceful (in the sense that they have a lot of resources) adversary can still effectively DoS the Responder.

Regardless of the type of rate-limiting used, there is a huge advantage in blocking the DoS attack using rate-limiting in that legitimate clients who are away from the attacking nodes should not be adversely affected by either the attack or by the measures used to counteract it.

6. Plan for Defending a Responder

This section outlines a plan for defending a Responder from a DDoS attack based on the techniques described earlier. The numbers given here are not normative, and their purpose is to illustrate the configurable parameters needed for defeating the DDoS attack.

Implementations may be deployed in different environments, so it is RECOMMENDED that the parameters be settable. As an example, most commercial products are required to undergo benchmarking where the IKE SA establishment rate is measured. Benchmarking is indistinguishable from a DoS attack and the defenses described in this document may defeat the benchmark by causing exchanges to fail or take a long time to complete. Parameters should be tunable to allow for benchmarking (if only by turning DDoS protection off).

Since all countermeasures may cause delays and work on the initiators, they SHOULD NOT be deployed unless an attack is likely to be in progress. To minimize the burden imposed on Initiators, the Responder should monitor incoming IKE requests, searching for two things:

1. A general DDoS attack. Such an attack is indicated by a high number of concurrent half-open SAs, a high rate of failed IKE_AUTH exchanges, or a combination of both. For example, consider a Responder that has 10,000 distinct peers of which at peak 7,500 concurrently have VPN tunnels. At the start of peak time, 600 peers might establish tunnels at any given minute, and tunnel establishment (both IKE_SA_INIT and IKE_AUTH) takes anywhere from 0.5 to 2 seconds. For this Responder, we expect there to be less than 20 concurrent half-open SAs, so having 100 concurrent half-open SAs can be interpreted as an indication of an attack. Similarly, IKE_AUTH request decryption failures

should never happen. Supposing the the tunnels are established using EAP (see section 2.16 or RFC 7296), users enter the wrong password about 20% of the time. So we'd expect 125 wrong password failures a minute. If we get IKE_AUTH decryption failures from multiple sources more than once per second, or EAP failure more than 300 times per minute, that can also be an indication of a DDoS attack.

2. An attack from a particular IP address or prefix. Such an attack is indicated by an inordinate amount of half-open SAs from that IP address or prefix, or an inordinate amount of IKE_AUTH failures. A DDoS attack may be viewed as multiple such attacks. If they are mitigated well enough, there will not be a need enact countermeasures on all Initiators. Typical figures might be 5 concurrent half-open SAs, 1 decrypt failure, or 10 EAP failures within a minute.

Note that using counter-measures against an attack from a particular IP address may be enough to avoid the load on the half-open SA database and the amount of failed IKE_AUTH exchanges to never exceed the threshold of attack detection. This is a good thing as it prevent Initiators that are not close to the attackers from being affected.

When there is no general DDoS attack, it is suggested that no Cookie or puzzles be used. At this point the only defensive measure is the monitoring, and setting a soft limit per peer IP or prefix. The soft limit can be set to 3-5, and the puzzle difficulty should be set to such a level (number of zero-bits) that all legitimate clients can handle it without degraded user experience.

As soon as any kind of attack is detected, either a lot of initiations from multiple sources or a lot of initiations from a few sources, it is best to begin by requiring stateless cookies from all Initiators. This will force the attacker to use real source addresses, and help avoid the need to impose a greater burden in the form of cookies on the general population of initiators. This makes the per-node or per-prefix soft limit more effective.

When Cookies are activated for all requests and the attacker is still managing to consume too many resources, the Responder MAY increase the difficulty of puzzles imposed on IKE_SA_INIT requests coming from suspicious nodes/prefixes. It should still be doable by all legitimate peers, but it can degrade experience, for example by taking up to 10 seconds to calculate the cookie extension.

If the load on the Responder is still too great, and there are many nodes causing multiple half-open SAs or IKE_AUTH failures, the

Responder MAY impose hard limits on those nodes.

If it turns out that the attack is very widespread and the hard caps are not solving the issue, a puzzle MAY be imposed on all Initiators.
Note that this is the last step, and the Responder should avoid this if possible.

7. Operational Considerations

[This section needs a lot of expanding]

Not all Initiators support the puzzles, but all initiators are supposed to support stateless cookies. If this notification is sent to a non-supporting but legitimate initiator, the exchange will fail.
Responders are advised to first try to mitigate the DoS using stateless cookies, even imposing them generally before resorting to using puzzles.

The difficulty level should be set by balancing the requirement to minimize the latency for legitimate initiators and making things difficult for attackers. A good rule of thumb is for taking about 1 second to solve the puzzle. A typical initiator or bot-net member in 2014 can perform slightly less than a million hashes per second per core, so setting the difficulty level to n=20 is a good compromise. It should be noted that mobile initiators, especially phones are considerably weaker than that. Implementations should allow administrators to set the difficulty level, and/or be able to set the difficulty level dynamically in response to load.

Initiators should set a maximum difficulty level beyond which they won't try to solve the puzzle and log or display a failure message to the administrator or user.

8. Security Considerations

To be added.

9. IANA Considerations

IANA is requested to assign a notify message type from the status types range (16430-40959) of the "IKEv2 Notify Message Types - Status Types" registry with name "PUZZLE".

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7296] Kivinen, T., Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7296, October 2014.

10.2. Informative References

[bitcoins] Nakamoto, S., "Bitcoin: A Peer-to-Peer Electronic Cash System", October 2008.

Author's Address

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 6789735
Israel

Email: ynir.ietf@gmail.com