DDoS Beasts and How to Fight Them

Artyom Gavrichenkov <ag@qrator.net>

Timeline of ancient history

- First attacks: 1999-2000
- 2005: **STRIDE** model by Microsoft
  - Spoofing Identity
  - Tampering with Data
  - Repudiation
  - Information Disclosure
  - Denial of Service
  - Elevation of Privileges

[D?]DoS

The difference between “a distributed attack” and an,
err, *not* distributed one is vague.

Traditional meaning: a distributed attack comes from *multiple sources*.

- What is a *source*? Is it an *IP address* or a *machine*?
- If it is a **machine**, does a virtual instance count?
  Or a few instances under the same physical hypervisor?
  What if they often migrate between physical machines?
  If I’m a victim, how do I tell a single-sourced from a multiple-sourced?
- If it is an **IP**, then how do we treat spoofed traffic?

[D?]DoS

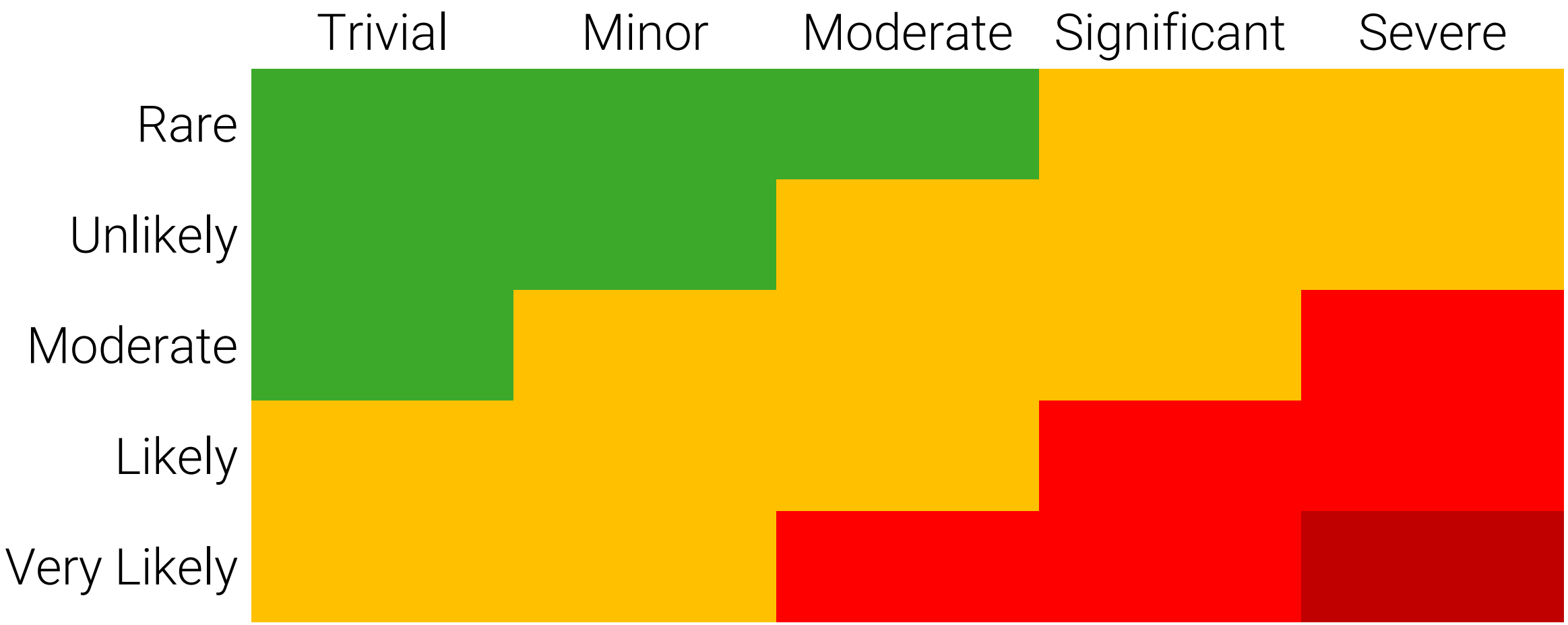
Hence, a different sort of thinking applies:

- DoS (as implied in STRIDE): a **vulnerability** in a software (e.g. **NULL** pointer dereference, like Ping of Death)
- **D**DoS: computational resource **exhaustion**

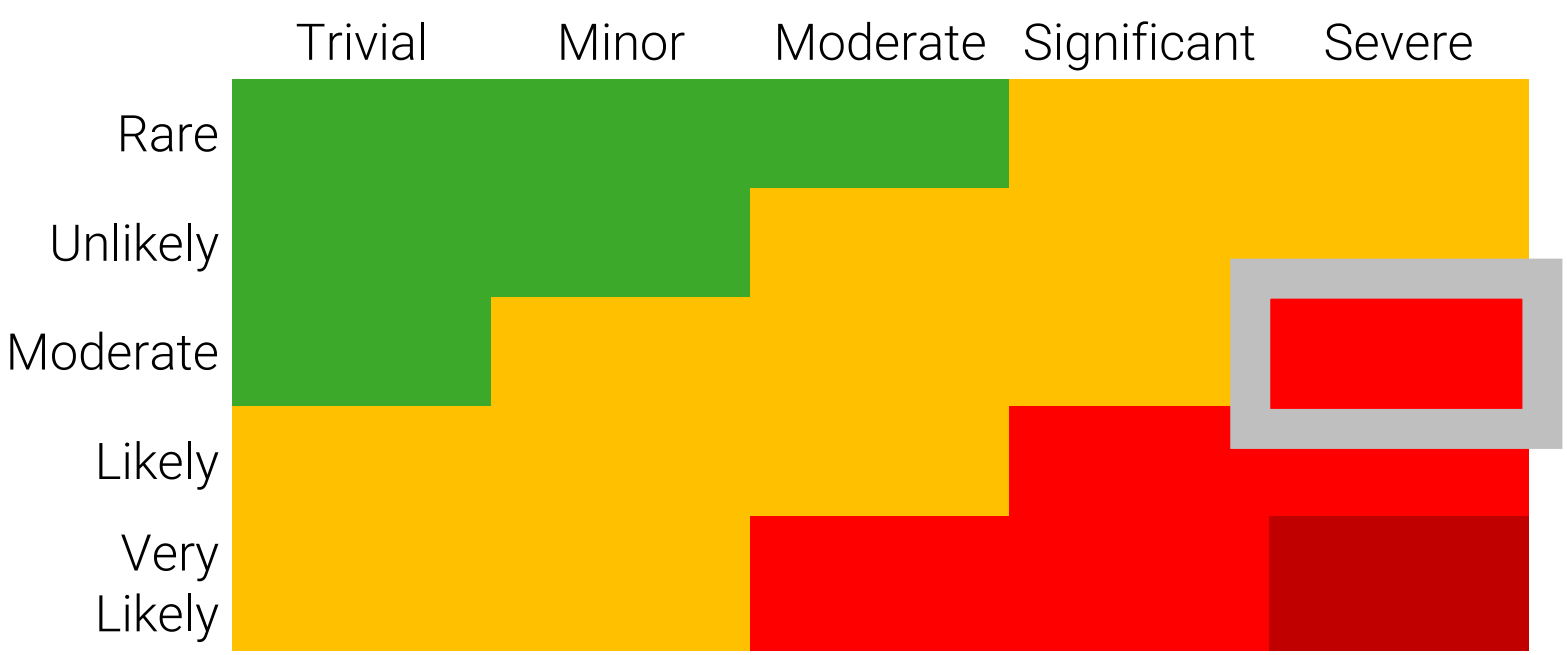
Risk management

The basic idea behind STRIDE and other approaches is **risk assessment, modelling and management**.

Probability/Impact Matrix



Probability/Impact Matrix



DDoS attack, 2018

- Impact: **Severe**
- Probability: ?

Motivation of an attacker

- Fun!
- Blackmail
- Self-promotion
- Political statement
- Revenge
- Market competition
- Diverting attention
  (e.g. in case of theft)
- Preventing access to a
  compromising information

Motivation of an attacker

- Fun!
- Blackmail
- Self-promotion
- Political statement
- Revenge
- Market competition
- Diverting attention (e.g. in case of theft)
- Preventing access to a compromising information

Rather hard to evaluate and control

More or less predictable!

Contents [hide]

Network resource exhaustion

- A computer network, as of today*, consists of *layers*
- A network resource is not available to its users
  when at least one network layer fails to provide service
- Hence, a DDoS attack can be attributed to a network layer
  which it affects

DDoS Classification

According to the ISO/OSI model:

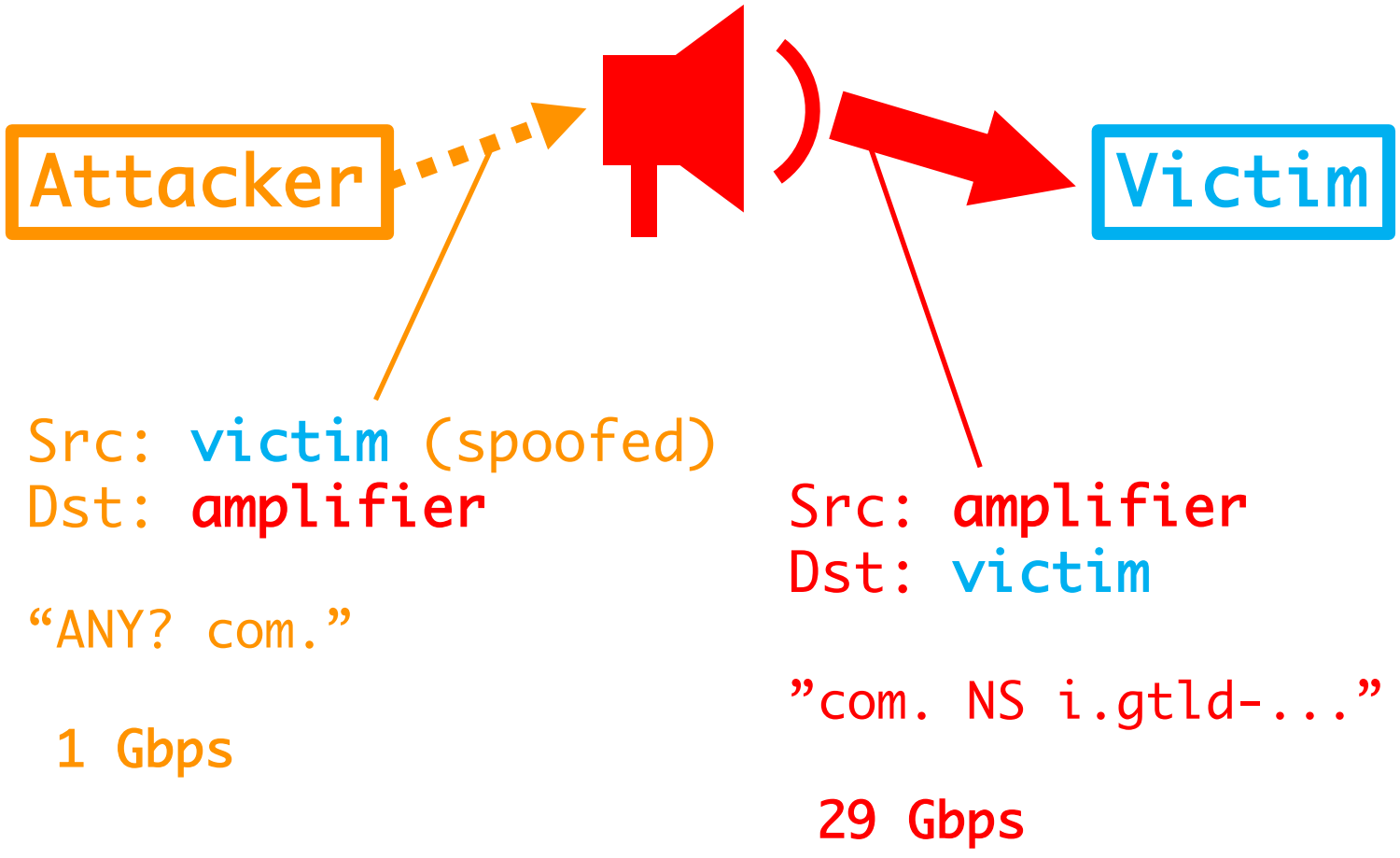
L2-3:	generic **bandwidth** exhaustion
L4-6:	exploitation of **TCP/TLS edge cases**
L7:	application-specific **bottlenecks**

Attack examples

- L2-3
  - Volumetric attacks: UDP flood, SYN flood, amplification…

Typical amplification attack

- Most servers on the Internet send more data to a client than they receive
- UDP-based servers generally do not verify the source IP address
- This allows for amplification DDoS



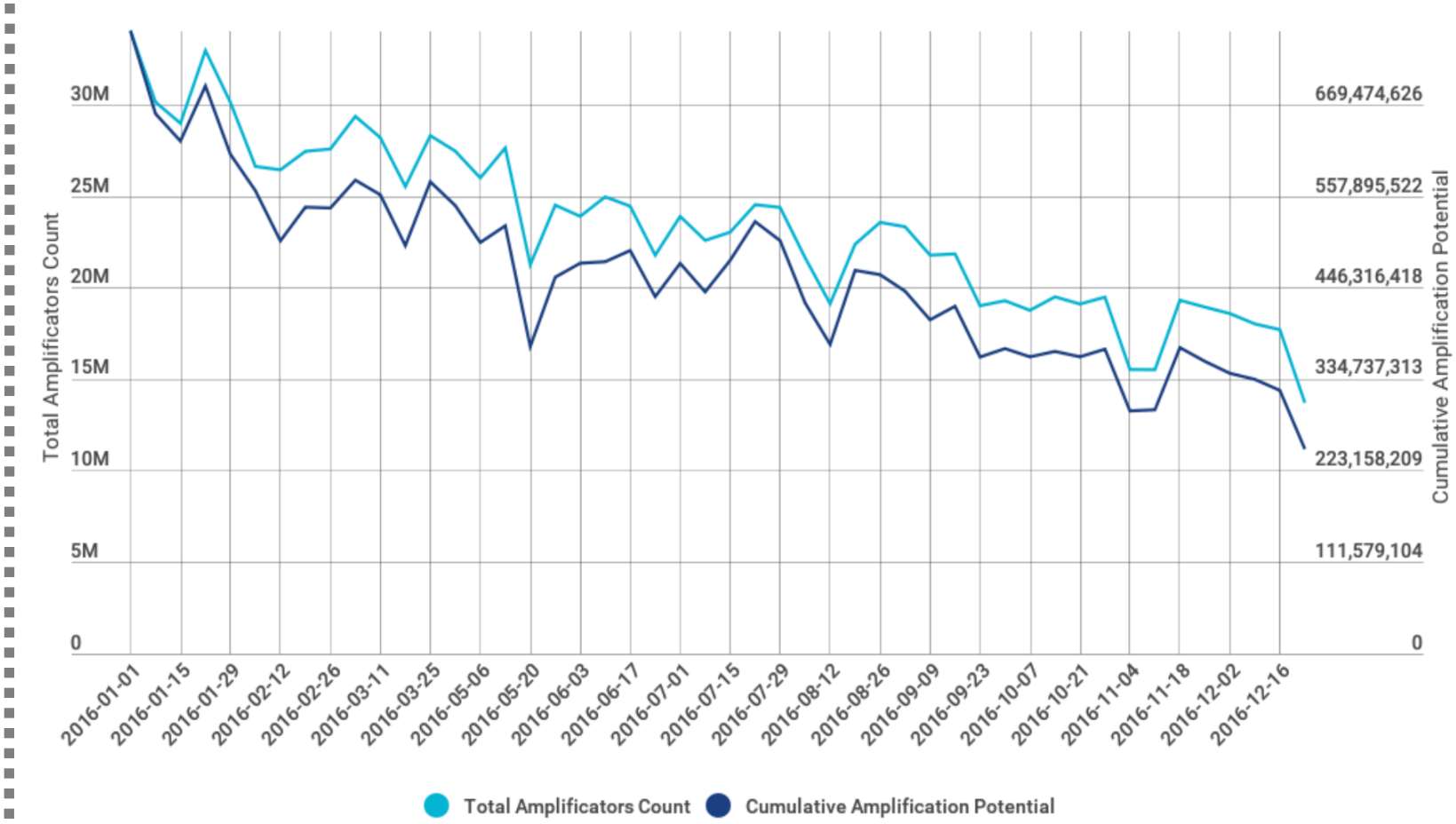
Vulnerable protocols

- A long list actually
- Mostly obsolete protocols (RIPv1 anyone?)
- Modern protocols as well: gaming

- NTP
- DNS
- SNMP
- SSDP
- ICMP
- NetBIOS
- RIPv1
- PORTMAP
- CHARGEN
- QOTD
- **Quake**
- …

Vulnerable servers

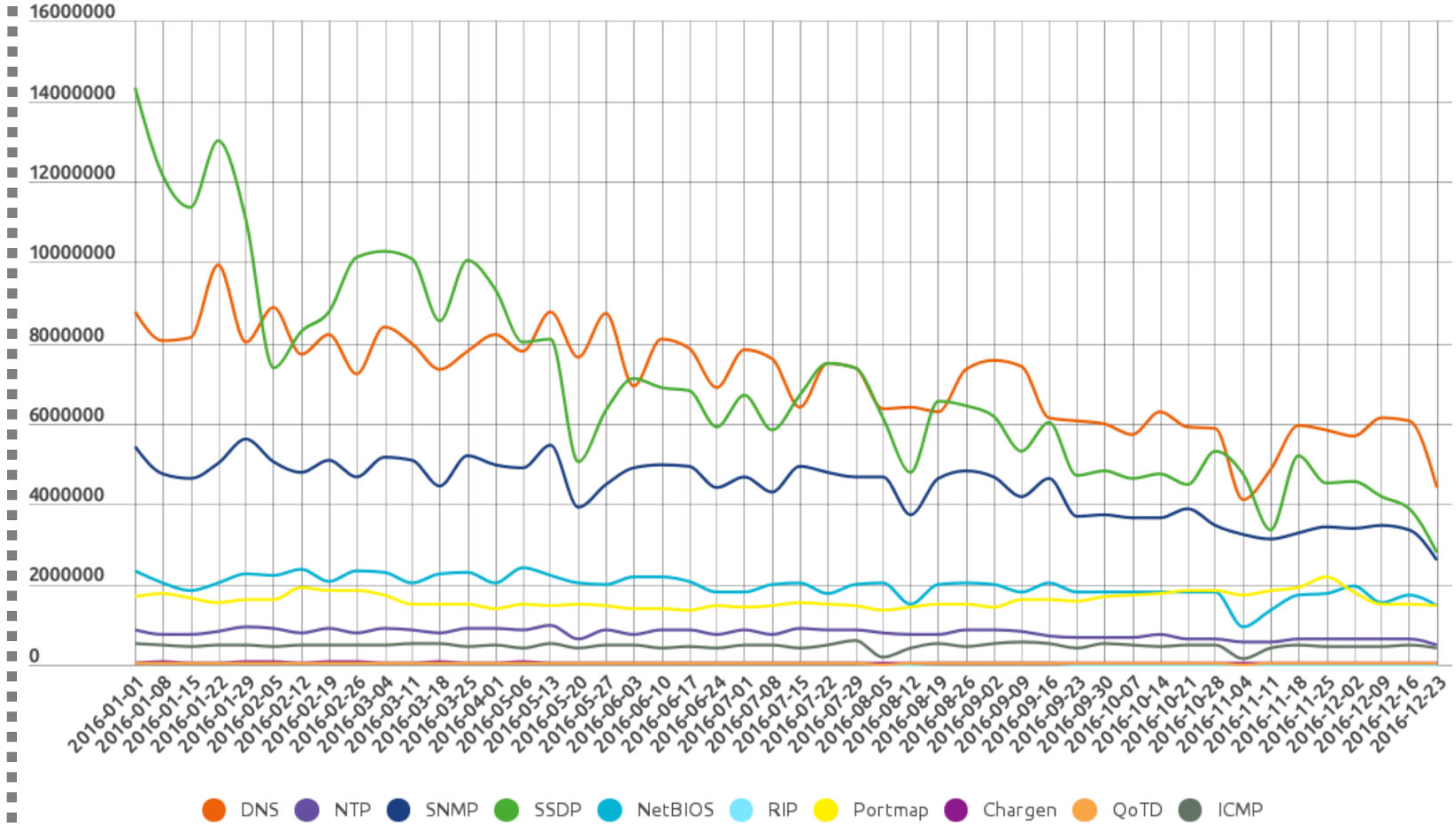
- As it's mostly obsolete servers, they eventually get updated
  - or replaced
  - or just trashed
- Thus,
  the amount of amplifiers shows steady downtrend



Source: Qrator.Radar network scanner

Amp power

- Downtrend in terms of the amount – and a downtrend in terms of available power
- However, once in a while, a new vulnerable protocol is discovered



Source: Qrator.Radar network scanner

Mitigation

- Most amplification attacks are easy to track, as the source UDP port is fixed

- NTP
- DNS
- SNMP
- SSDP
- ICMP
- NetBIOS
- RIPv1
- PORTMAP
- CHARGEN
- QOTD
- **Quake**
- …

BGP Flow Spec solves problems?

Mitigation

- Most amplification attacks are easy to track, as the source UDP port is fixed
- Two major issues:
  - ICMP
  - Amplification without a fixed port (Bittorrent?)

- NTP
- DNS
- SNMP
- SSDP
- ICMP
- NetBIOS
- RIPv1
- PORTMAP
- CHARGEN
- QOTD
- **Quake**
- …

memcached

- A **fast** in-memory cache
- Heavily used in Web development

memcached

- A **fast** in-memory cache
- Heavily used in Web development

- Listens on all interfaces, port 11211, by default

memcached

- Basic ASCII protocol doesn't do authentication
- 2014, Blackhat USA:
  "An attacker can inject arbitrary data into memory"

memcached

- Basic ASCII protocol doesn't do authentication
- 2014, Blackhat USA:
  "An attacker can inject arbitrary data into memory"

- **2017, Power of Community:**

  "An attacker can send data from memory to a third party via spoofing victim's IP address"

```
import memcache
m = memcache.Client([
    ‘reflector.example.com:11211’
])
m.set(’a’, value)
```

– to inject a value of an arbitrary size under key “a”

```
print ’\0\x01\0\0\0\x01\0\0gets a\r\n’
```

– to retrieve a value

```
print ’\0\x01\0\0\0\x01\0\0gets a a a a a\r\n’
```

– to retrieve a value **5 times**

```
print ’\0\x01\0\0\0\x01\0\0gets a a a a a\r\n’
```

– to retrieve a value **5 times.**

Or 10 times.
Or a hundred.

Default memcached conf. in Red Hat

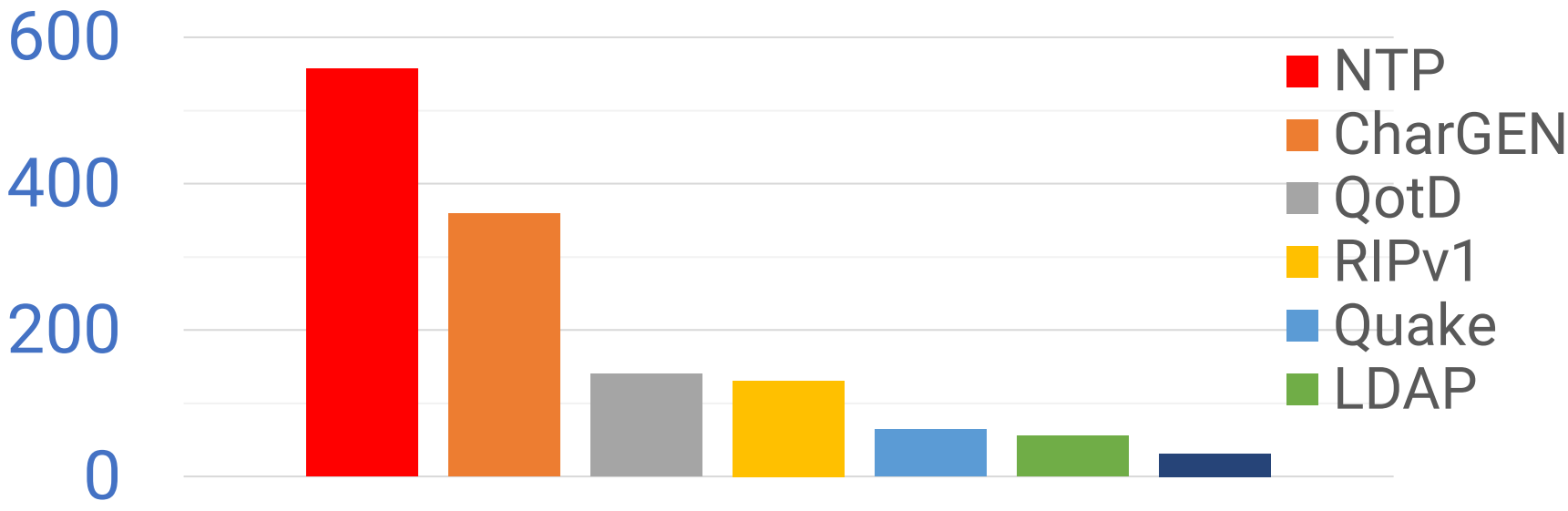
- memcached listens on all network interfaces
- both TCP and UDP transports are enabled
- no authentication is required to access Memcached

- the service has to be manually enabled or started
- the default firewall configuration
  does not allow remote access to Memcached

- Also Zimbra, etc.

Amplification factor

- Typical amplification factor used to be hundreds
- For **memcached**, it's millions, and no fixed source port
- Amplification isn't something to underestimate



Source: https://www.us-cert.gov/ncas/alerts/TA14-017A

```
ipv4 access-list exploitable-ports
 permit udp any eq 11211 any
!
ipv6 access-list exploitable-ports-v6
 permit udp any eq 11211 any
!
class-map match-any exploitable-ports
 match access-group ipv4 exploitable-ports
 end-class-map
!
policy-map ntt-external-in
 class exploitable-ports
  police rate percent 1
   conform-action transmit
   exceed-action drop
  !
  set precedence 0
  set mpls experimental topmost 0
 !
```

Source: http://mailman.nlnog.net/pipermail/nlnog/2018-March/002697.html

```
...
    class class-default
     set mpls experimental imposition 0
     set precedence 0
    !
    end-policy-map
  !
  interface Bundle-Ether19
   description Customer: the best customer
   service-policy input ntt-external-in
   ipv4 address xxx/x
   ipv6 address yyy/y
   ...
  !
  interface Bundle-Ether20
   service-policy input ntt-external-in
   ...
  ... etc ...
```

Source: http://mailman.nlnog.net/pipermail/nlnog/2018-March/002697.html

Proof of Source Address Ownership

E.g., QUIC:

- Initial handshake packet padded to 1280 bytes
- Source address validation

Attack examples

- L2-3
  - Volumetric attacks: UDP flood, SYN flood, amplification…

IoT attacks!

- 2014: LizardStresser
- 2015: SOHO routers become a persistent target for malware
- 2016: Mirai
- 2017: Persirai, Hajime, …

Attack examples

- L2-3
  - Volumetric attacks: UDP flood,
    SYN flood, amplification,
    and so on (we don’t *need* to care exactly)
  - Infrastructure attacks

L2-3 mitigation

From a victim’s perspective:

- Anycast network with enough inspection power
- Inventory management to drop unsolicited traffic vectors (e.g. UDP towards an HTTP server)
- Rate-limiting less important traffic
- Challenges and handshakes (more on that later)

L2-3 mitigation

From a victim’s perspective:

- Anycast network with enough inspection power
- Inventory management to drop unsolicited traffic vectors (e.g. UDP towards an HTTP server)
- Rate-limiting less important traffic
- Challenges and handshakes (more on that later)

From an ISP’s view:

- Simple heuristics against typical attacks
- RTBH (and let the customer take care of it themselves)

Attack examples

- L2-3
  - Volumetric attacks: UDP flood,
    SYN flood, amplification,
    and so on (we don’t *need* to care exactly)
  - Infrastructure attacks

Attack examples

- L2-3
  - Volumetric attacks: UDP flood,
    SYN flood, amplification,
    and so on (we don't *need* to care exactly)
  - Infrastructure attacks
- L4-6
  - SYN flood, TCP connection flood,
    Sockstress, **and so on**
  - TLS attacks

Attack examples

- L2-3
  - Volumetric attacks: UDP flood, **SYN flood**, amplification, and so on (we don't need to care exactly)
  - Infrastructure attacks
- L4-6
  - **SYN flood**, TCP connection flood, Sockstress, and so on
  - TLS attacks

An attack can affect multiple layers at once

```
988                 //util_strcpy(buf + util_strlen(buf), "POST /cdn-cgi/l/chk_captcha
989                 util_strcpy(buf + util_strlen(buf), "POST /cdn-cgi/");
990                 rand_alphastr(buf + util_strlen(buf), 16);
991                 util_strcpy(buf + util_strlen(buf), " HTTP/1.1\r\nUser-Agent: ");
992                 util_strcpy(buf + util_strlen(buf), conn->user_agent);
993                 util_strcpy(buf + util_strlen(buf), "\r\nHost: ");
994                 util_strcpy(buf + util_strlen(buf), conn->domain);
995                 util_strcpy(buf + util_strlen(buf), "\r\n");
```

```
21:30:01.226868 IP 94.251.116.51 > 178.248.233.141:
  GREv0, length 544:
  IP 184.224.242.144.65323 > 167.42.221.164.80:
  UDP, length 512

21:30:01.226873 IP 46.227.212.111 > 178.248.233.141:
  GREv0, length 544:
  IP 90.185.119.106.50021 > 179.57.238.88.80:
  UDP, length 512

21:30:01.226881 IP 46.39.29.150 > 178.248.233.141:
  GREv0, length 544:
  IP 31.173.79.118.42580 > 115.108.7.79.80:
  UDP, length 512
```

L4+ mitigation

- SYN flood: 3-way handshake-based SYN cookies & SYN proxy, allowing a victim to verify the source IP address
- Other packet-based flood: other handshakes and challenges to do the same
- The rest: session analysis, heuristics and blacklists

- **It is dangerous to use blacklists or whitelists without source IP address verification!**

IPv6 issues

- 128-bit IP addresses
- **Possible**: to address each atom on the Earth surface
- **Impossible**: to store a large number of entries in memory

- About 10 years ago, blacklisting whole IPv4 networks was already considered a bad practice
- With IPv6, this method has no other way than to return

Attack examples

- L2-3
  - Volumetric attacks: UDP flood,
    SYN flood, amplification,
    and so on (we don't *need* to care exactly)
  - Infrastructure attacks
- L4-6
  - SYN flood, TCP connection flood,
    Sockstress, **and so on**
  - TLS attacks

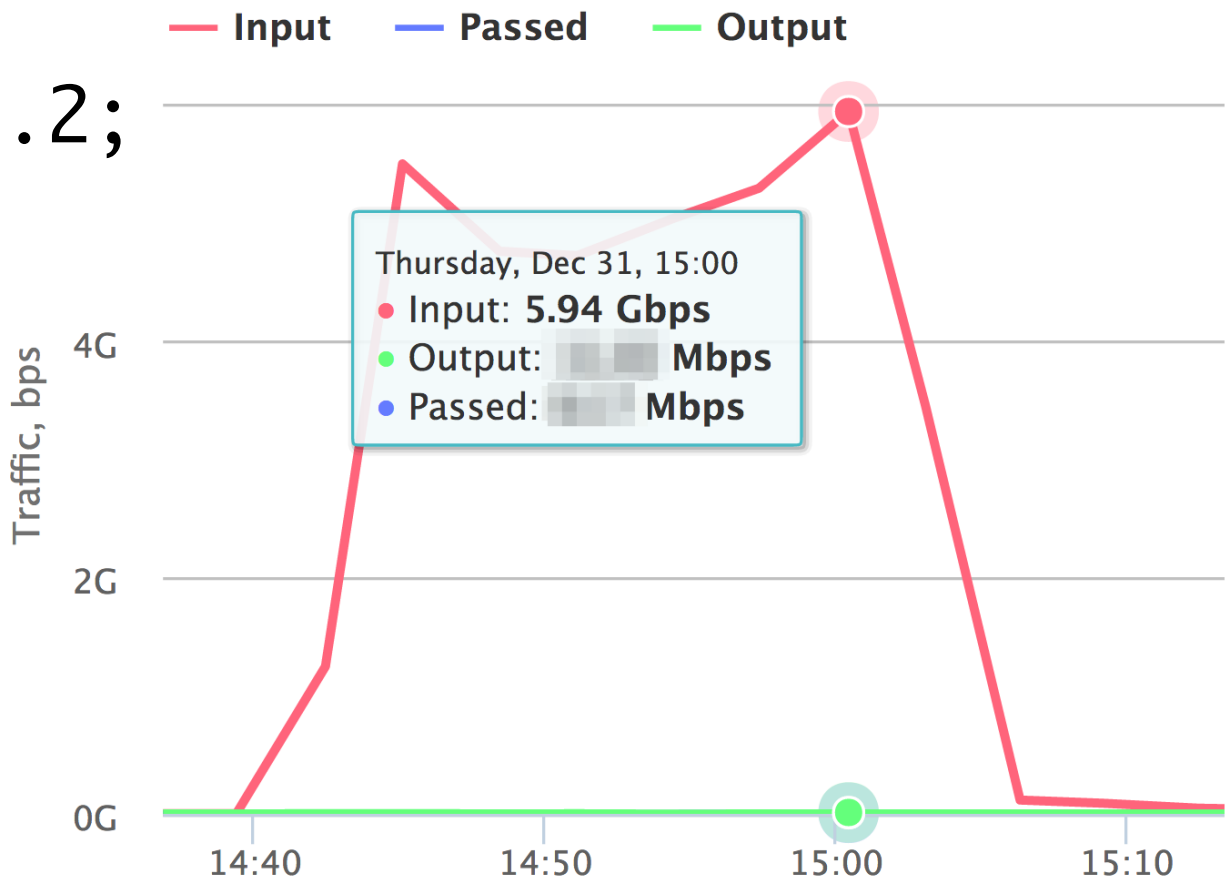
Attack examples

- L2-3
  - Volumetric attacks: UDP flood, SYN flood, amplification, **and so on** (we don't *need* to care exactly)
  - Infrastructure attacks
- L4-6
  - SYN flood, TCP connection flood, Sockstress, **and so on**
  - TLS attacks
- L7
  - Application-specific flood

Wordpress Pingback

```
GET /whatever
User-Agent: WordPress/3.9.2;
 http://example.com/;
 verifying pingback
 from 192.0.2.150
```

- 150 000 – 170 000 vulnerable servers at once
- SSL/TLS-enabled



Data from Qrator monitoring engine

Another example of a L7 attack: FBS

- A bot can actually be more clever than a Wordpress machine
- Advanced botnets are capable of using a headless browser (IE/Edge or Chrome)
  => *“full browser stack”* (FBS) botnets
- A FBS-enabled bot is able to go through even complex challenges, like Javascript code execution

Another example of a L7 attack: FBS

CAPTCHA is a weapon of last resort against FBS.

Pros:

- Easy to implement
- Generally, might work

Cons (1/2):

- Sometimes harder for humans than for robots
- Not all bots are malicious, and not all humans are innocent
- CAPTCHA proxies and farms, like http://antigate.com/

Another example of a L7 attack: FBS

CAPTCHA is a weapon of last resort against FBS.

Pros:

- Easy to implement
- Generally, might work

Cons (2/2):

- OCR tools evolve fast
- Voice recognition evolves even faster
- “Security by obscurity”: an open-sourced CAPTCHA is relatively easy to break using open source machine learning tools. Example: https://medium.com/@ageitgey/how-to-break-a-captcha-system-in-15-minutes-with-machine-learning-dbebb035a710

Another example of a L7 attack: FBS

Under most conditions though, unlike Wordpress pingback,
such attacks won’t cause a link degradation,
hence generally out of scope of a network operator’s responsibility

Another example of a L7 attack: DNS

- DNS is built on top of UDP*,
  and a DNS request fits in a packet
- The structure of a DNS query is simple

DNS lookup

```
10:00:34.510826 IP
   (proto UDP (17), length 56)
   192.168.1.5.63097 > 8.8.8.8.53:
      9508+
      A? facebook.com.
      (30)
10:00:34.588632 IP
   (proto UDP (17), length 72)
   8.8.8.8.53 > 192.168.1.5.63097:
      9508 1/0/0
      facebook.com. A 31.13.72.36
      (45)
```

DNS lookup

- DNS is built on top of UDP*, and a DNS request fits in a packet
- The structure of a DNS query is simple
- An attacker capable of generating spoofed queries
  will make a userspace DNS application process
  all those fake requests,
  rendering a DNS server unavailable L7-wise.

DNS lookup

- An attacker capable of generating spoofed queries will make an userspace DNS application process all those fake requests, rendering a DNS server unavailable, this time L7-wise.
- *“Water torture”*
- This is what happened in October 2016 with Dyn.

Home / Internet

Major DDoS attack on Dyn DNS knocks Spotify, Twitter, Github, PayPal, and more offline

The sound of silence.

DNS lookup

- An attacker capable of generating spoofed queries
  will make an userspace DNS application process
  all those fake requests,
  rendering a DNS server unavailable, this time L7-wise.

- Luckily, DNS protocol allows switching to TCP,
  and in TCP, we have a handshake to verify the source IP address,
  hence, blacklists apply.
- Once again, though, enough bandwidth and inspection power
  is required

DNS lookup

- Luckily, DNS protocol allows switching to TCP,
  and in TCP, we have a handshake to verify the source IP address,
  hence, blacklists apply.
- Unfortunately, other UDP-based protocols (e.g. gaming)
  are mostly built without DDoS mitigation in mind

Attack examples

- L2-3
  - Volumetric attacks: UDP flood, SYN flood, amplification, **and so on** (we don't *need* to care exactly)
  - Infrastructure attacks
- L4-6
  - SYN flood, TCP connection flood, Sockstress, **and so on**
  - TLS attacks
- L7
  - Application-based flood

A classification which is:

- Mutually **exclusive** *
- Collectively **exhaustive**

However

The Internet is a complex thing.

A decades old job interview quiz

- *“What happens when you type www.google.com in your browser?”*

- https://github.com/alex/what-happens-when:

Table of Contents

- The "g" key is pressed
- The "enter" key bottoms out
- Interrupt fires [NOT for USB keyboards]
- (On Windows) A `WM_KEYDOWN` message is sent to the app
- (On OS X) A `KeyDown` NSEvent is sent to the app
- (On GNU/Linux) the Xorg server listens for keycodes

“What happens when…”?

- DNS lookup
- Opening of a socket
- TLS handshake
- HTTP protocol
- HTTP Server Request Handle

“What happens when…”?

- DNS lookup
- IPv4/IPv6 selection
- Opening of a socket
- Deep packet inspection
- TLS handshake
- CRL/OCSP
- HTTP protocol
- Load balancer
- HTTP Server Request Handle
- CDN

“What happens when…”?

- DNS lookup
- IPv4/IPv6 selection
- Opening of a socket
- Deep packet inspection
- TLS handshake
- CRL/OCSP
- HTTP protocol
- Load balancer
- HTTP Server Request Handle
- CDN

- As the Dyn incident shows: an application server could not only be a *direct* target of a DDoS attack
- Each step could suffer from an attack, L2-L7-wise

- **Inventory management**
- **Infrastructure monitoring**

Architectural view

- Security is not a product, not an appliance, it’s a process
- Ability of a DDoS mitigation must be built into the design of any protocol
- A concerned company must follow policies:
  - Updates
  - Risk management
  - Incident handling

Risk management for a network operator

- A network operator will basically suffer only from bandwidth-consuming attacks
- However, an attacker will most likely use just the tool they have at their disposal: amplifier or a botnet, doesn’t matter

- Thus, the probability of an attack towards the network is **the aggregate probability** of an attack for each customer in the network

What’s next?

- memcached:
  - Disclosure in November 2017
  - In the wild: February 2018

- Three months are an overly short interval
- Next time, it might be even shorter
- Meltdown/Spectre show: the “embargo” approach doesn’t work well for a community large enough

What’s next?

- Collaboration
- Proper and timely reaction
- RFC 2350: ~~CERT/CSIRT~~ for network operators?
  - No matter the name

Q&A

mailto: Artyom Gavrichenkov <ag@qrator.net>