Network Security - part II

Srdjan Matic

23rd September 2024

based on the material prepared by:
Emiliano de Cristofaro and Juan Caballero

Outline

Recap - SYN Flooding

How does SYN flooding work[a]?

[a] https://datatracker.ietf.org/doc/html/rfc4987

Recap - SYN Flooding

How does SYN flooding work[a]?

- protocol: TCP or UDP?
- target#1: network bandwidth or the protocol?
- target#2: single host or the single application?
- spoofing#1: optional or necessary?
- spoofing#2: "reachable" or "unreachable" hosts?

[a] https://datatracker.ietf.org/doc/html/rfc4987

Recap - SYN Flooding

How does SYN flooding work[a]?

- protocol: TCP or UDP? $\Rightarrow$ **TCP**
- target#1: network bandwidth or the protocol?
- target#2: single host or the single application?
- spoofing#1: optional or necessary?
- spoofing#2: "reachable" or "unreachable" hosts?

[a] https://datatracker.ietf.org/doc/html/rfc4987

Recap - SYN Flooding

How does SYN flooding work[a]?

- protocol: TCP or UDP? ⇒ **TCP**
- target#1: network bandwidth or the protocol? ⇒ **protocol**
- target#2: single host or the single application?
- spoofing#1: optional or necessary?
- spoofing#2: "reachable" or "unreachable" hosts?

[a] https://datatracker.ietf.org/doc/html/rfc4987

Recap - SYN Flooding

How does SYN flooding work[a]?

- protocol: TCP or UDP? ⇒ **TCP**
- target#1: network bandwidth or the protocol? ⇒ **protocol**
- target#2: single host or the single application? ⇒ **single application**
- spoofing#1: optional or necessary?
- spoofing#2: "reachable" or "unreachable" hosts?

[a] https://datatracker.ietf.org/doc/html/rfc4987

Recap - SYN Flooding

How does SYN flooding work[a]?

- protocol: TCP or UDP? ⇒ **TCP**
- target#1: network bandwidth or the protocol? ⇒ **protocol**
- target#2: single host or the single application? ⇒ **single application**
- spoofing#1: optional or necessary? ⇒ **optional**
- spoofing#2: "reachable" or "unreachable" hosts?

[a] https://datatracker.ietf.org/doc/html/rfc4987

Recap - SYN Flooding

How does SYN flooding work[a]?

- protocol: TCP or UDP? ⇒ **TCP**
- target#1: network bandwidth or the protocol? ⇒ **protocol**
- target#2: single host or the single application? ⇒ **single application**
- spoofing#1: optional or necessary? ⇒ **optional**
- spoofing#2: "reachable" or "unreachable" hosts? ⇒ **"unreachable"**

[a] `https://datatracker.ietf.org/doc/html/rfc4987`

Solution: SYN Cookies

IDEA: store the state info. *after establishing the connection* [Bernstein 1996].

How to achieve it: using a smart selection of server Initial Sequence Number.

- ISN = SYN Cookie
  - **t** = timestamp incremented every 64 seconds
  - **m** = Maximum Segment Size (MSS)
  - **K** randomly selected at boot
  - **s** = *hash*(src. IP, src. port, dst. IP, dst. port, t mod 32, K)

t mod 32	**m**	**s**
5 bits	3 bits	24 bits

- On reception of client's ACK, the server checks:
  - **t** against the current timestamp (to check if is expired)
  - **s** to verify if it is a valid SYN cookie
  - decodes **m** from the 3-bit encoding in the SYN cookie
    $\Longrightarrow$ if successful, create store the state for the connection

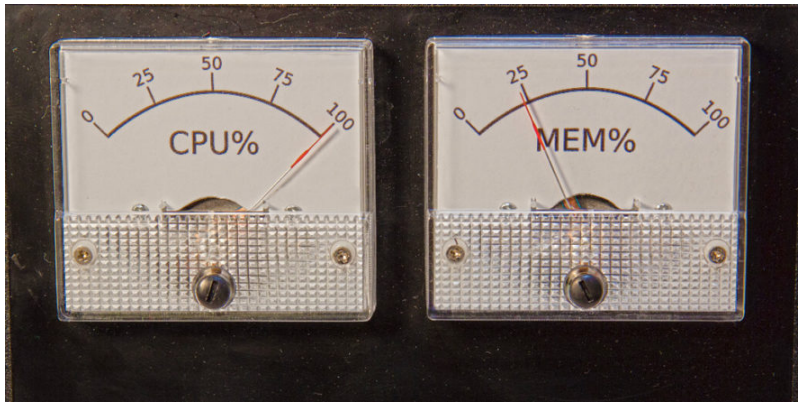
Limitations:

- Only 8 possible encodings for the MSS.
- Not support for TCP options $\Rightarrow$ partially supported since Linux 2.6.26.

Additional info:
`https://cr.yp.to/syncookies.html`
`https://www.geeksforgeeks.org/how-syn-cookies-are-used-to-preventing-syn-flood-attack/`

More Expensive DoS Attacks

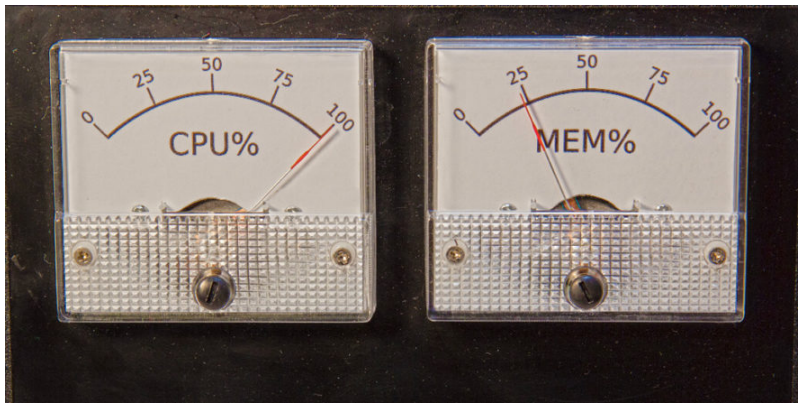
- Force victim server to do work:
  - complete TCP connection;
  - establish SSL connection;
  - application layer attacks (e.g., HTTP).



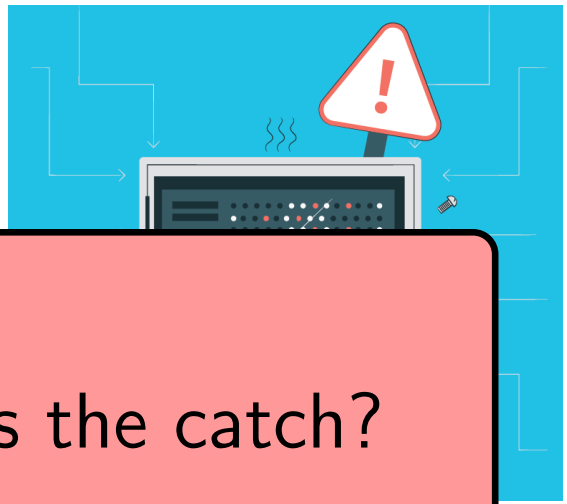
- Server work >> Client work.
- RSA encrypt $\sim$ 10x RSA decrypt.
  $\Rightarrow$ Server workload is *10x* compared to the client.

More Expensive DoS Attacks

- Force victim server to do work:
  - complete TCP connection;
  - establish SSL connection;
  - application layer attacks (e.g., HTTP).



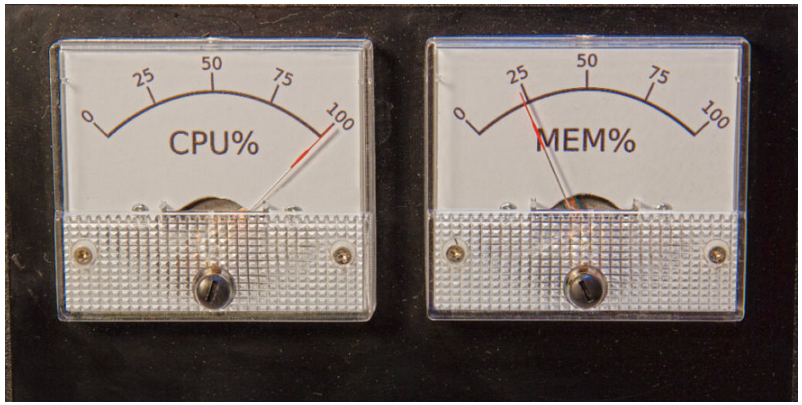
- Server work >> Client work.
- RSA encrypt $\sim$ 10x RSA decrypt.
  $\Rightarrow$ Server workload is *10x* compared to the client.



Looks great, where's the catch?

More Expensive DoS Attacks

- Force victim server to do work:
  - complete TCP connection;
  - establish SSL connection;
  - application layer attacks (e.g., HTTP).



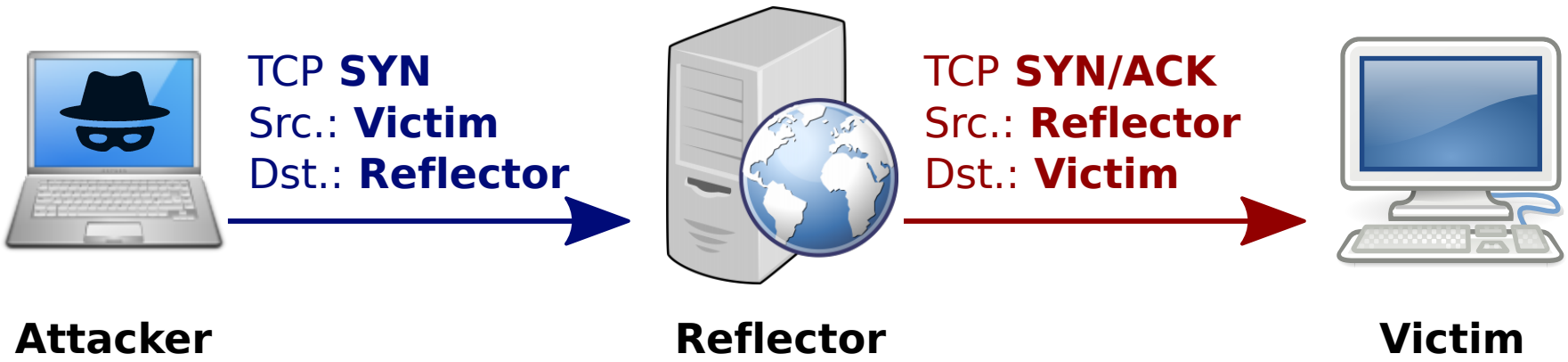
- Server work >> Client work.
- RSA encrypt $\sim$ 10x RSA decrypt.
  $\Rightarrow$ Server workload is *10x* compared to the client.

- Spoofing cannot be used:
  $\Rightarrow$ **It discloses the IP address of the attacker.**

Reflector Attacks

Procedure:

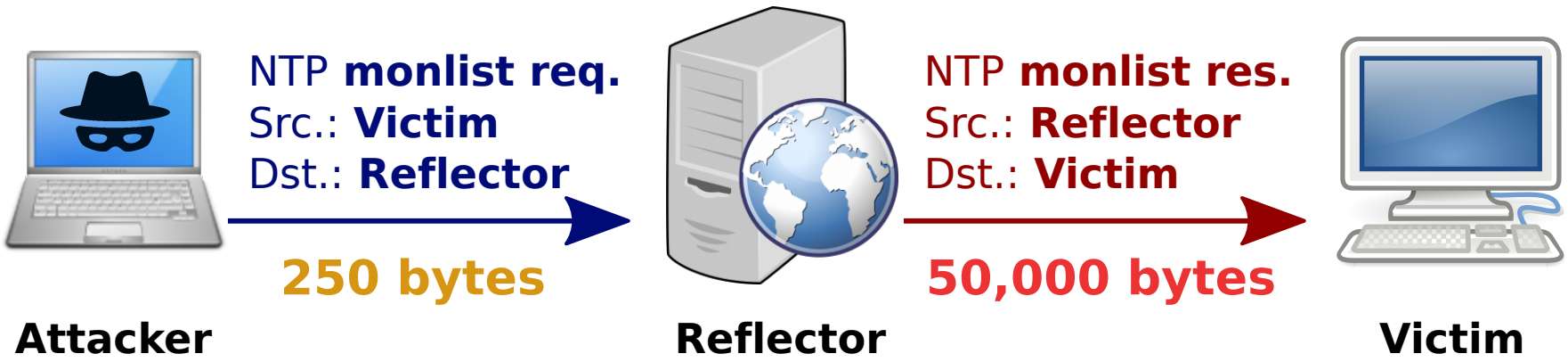
1. Sent traffic to the reflector using the IP of the victim as *source IP address*[1].
2. The response will be sent to the victim.
3. The victim will blame the reflector.



[1] http://ccr.sigcomm.org/archive/2001/jul01/ccr-200107-paxson.pdf

Reflector Amplification Attacks

- Attacker selects requests where:
  reflector response > **Attacker request**
- Bandwidth Amplification Factor
  BAF = size(payload_to_victim) / size(payload_to_reflector)
- Packet Amplification Factor
  PAF = packets_to__victim / packets_to_reflector



BAF = x200

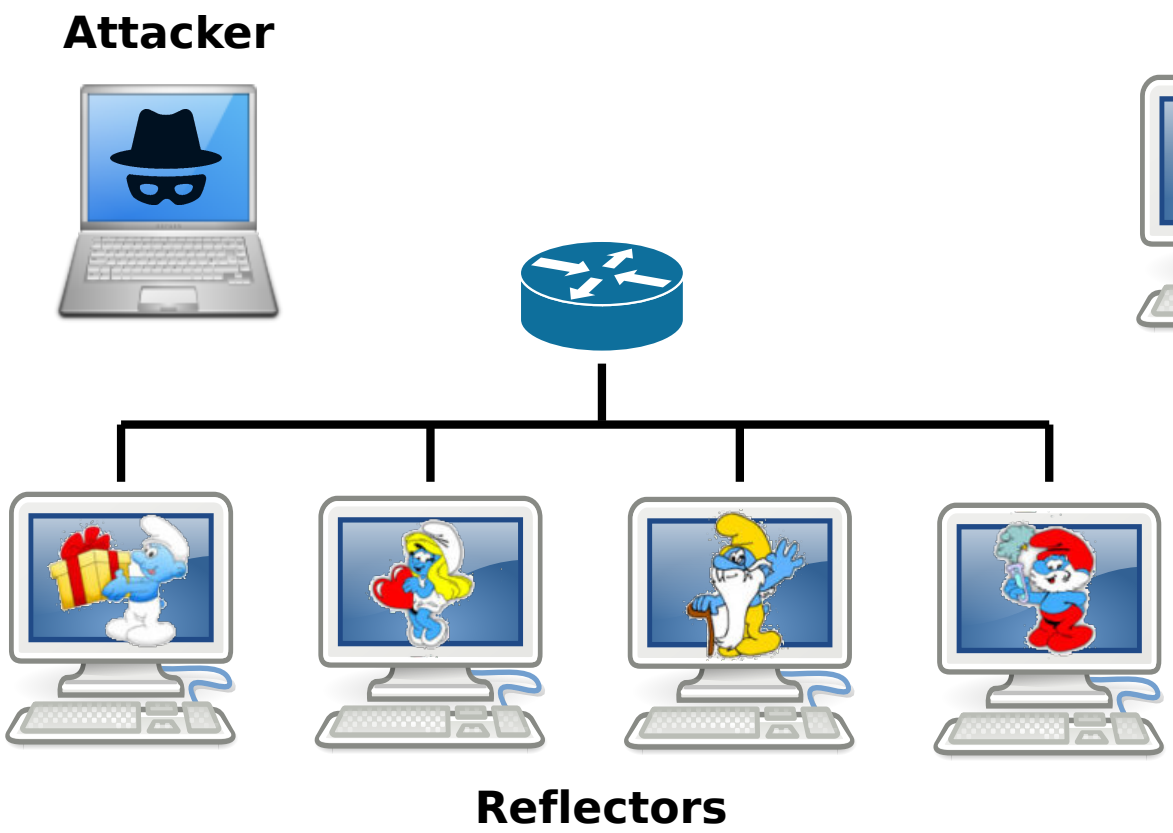
The NTP *monlist* command: `https://nmap.org/nsedoc/scripts/ntp-monlist.html`

Smurf Attack

- Popular in 1990, nowadays almost disappeared.
- ICMP reflector amplification attack:
  1. Attacker sends ping request to a broadcast address;
  2. All hosts in the local network reply to the target.

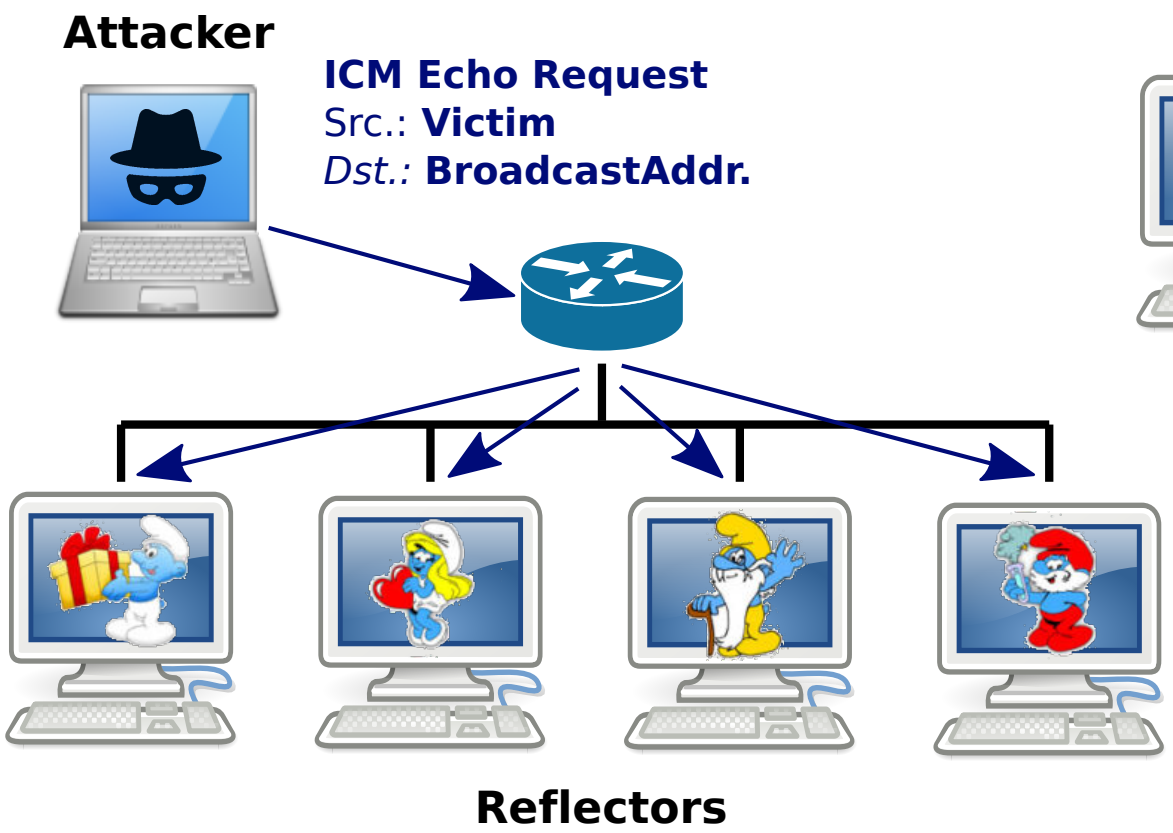
Smurf Attack

- Popular in 1990, nowadays almost disappeared.
- ICMP reflector amplification attack:
  1. Attacker sends ping request to a broadcast address;
  2. All hosts in the local network reply to the target.



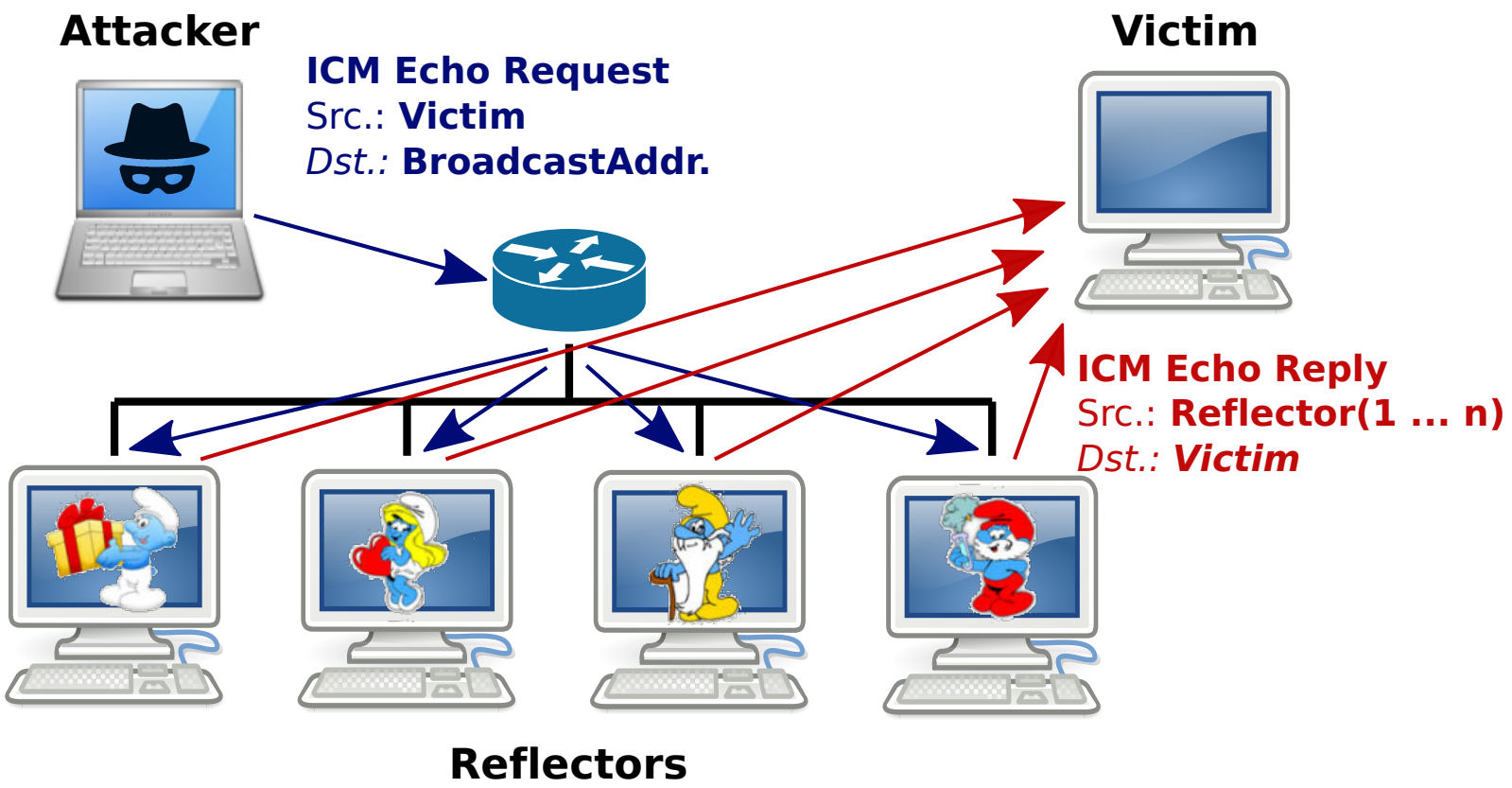
Smurf Attack

- Popular in 1990, nowadays almost disappeared.
- ICMP reflector amplification attack:
  1. Attacker sends ping request to a broadcast address;
  2. All hosts in the local network reply to the target.



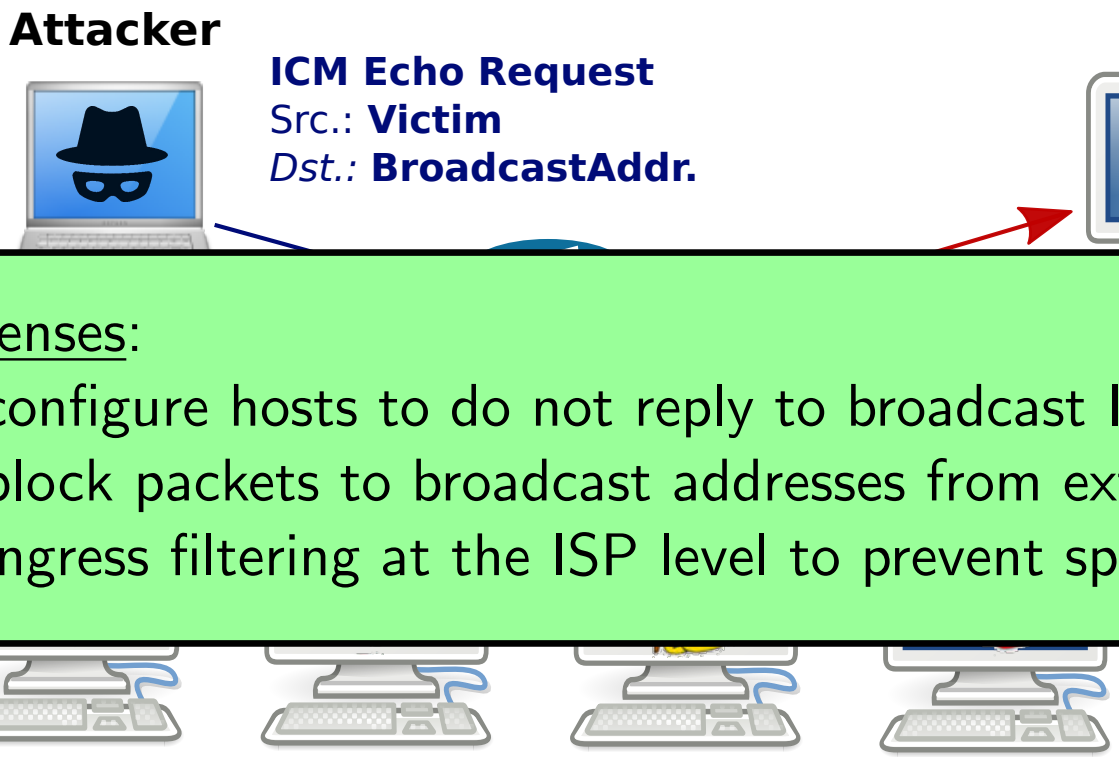
Smurf Attack

- Popular in 1990, nowadays almost disappeared.
- ICMP reflector amplification attack:
  1. Attacker sends ping request to a broadcast address;
  2. All hosts in the local network reply to the target.



Smurf Attack

- Popular in 1990, nowadays almost disappeared.
- ICMP reflector amplification attack:
  1. Attacker sends ping request to a broadcast address;
  2. All hosts in the local network reply to the target.



Defenses:

1. configure hosts to do not reply to broadcast ICMP requests;
2. block packets to broadcast addresses from external interfaces;
3. ingress filtering at the ISP level to prevent spoofing.

Botnets - I

- **Bot**:
- **Botnet**:
- **Botmaster**:

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**:

- **Botmaster**:

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**:

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

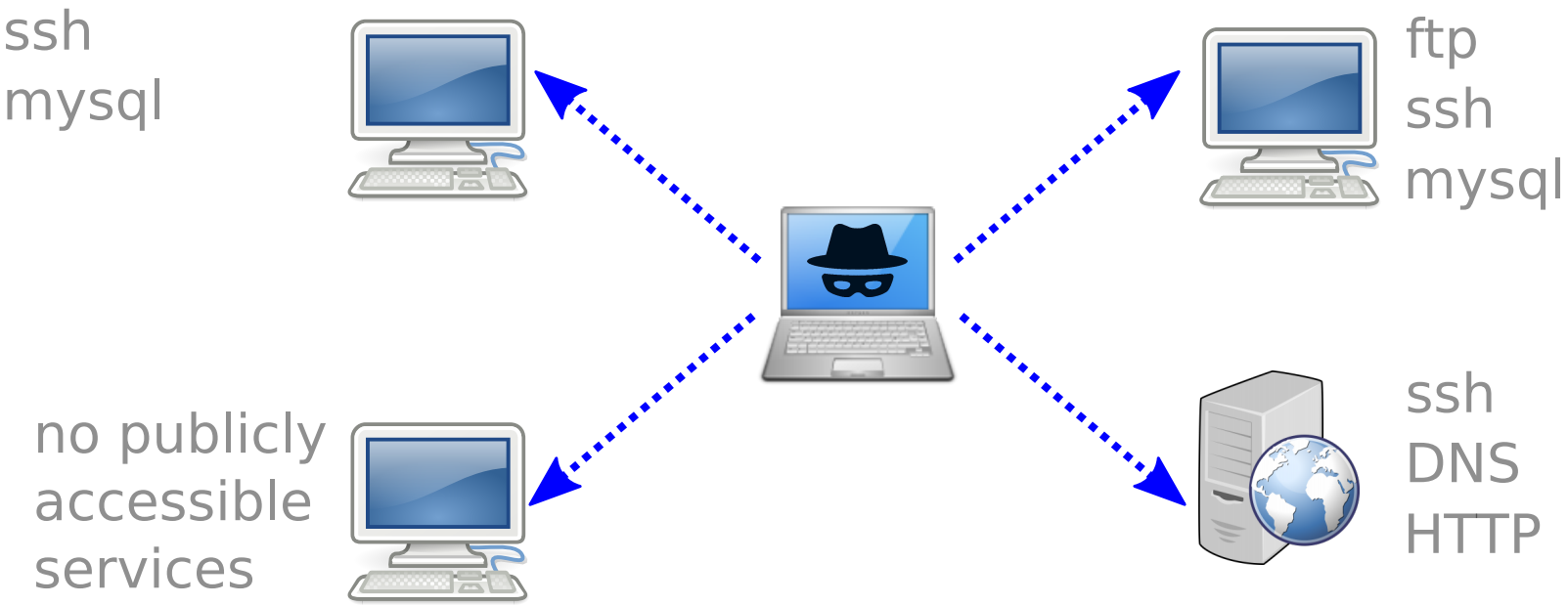
Overview of the botnet creation process:

Attacker's GOAL: take control of other machines.

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

Overview of the botnet creation process:

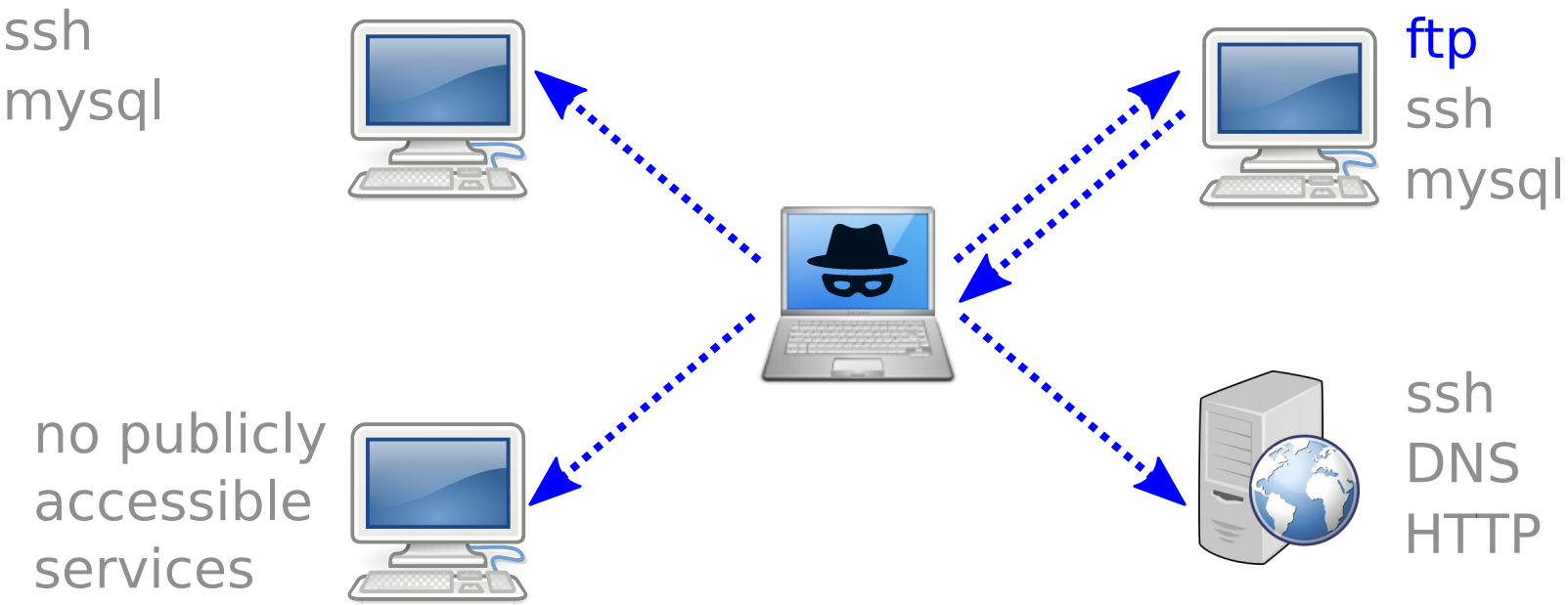


Scan for open port 21 (*ftp*)

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

Overview of the botnet creation process:

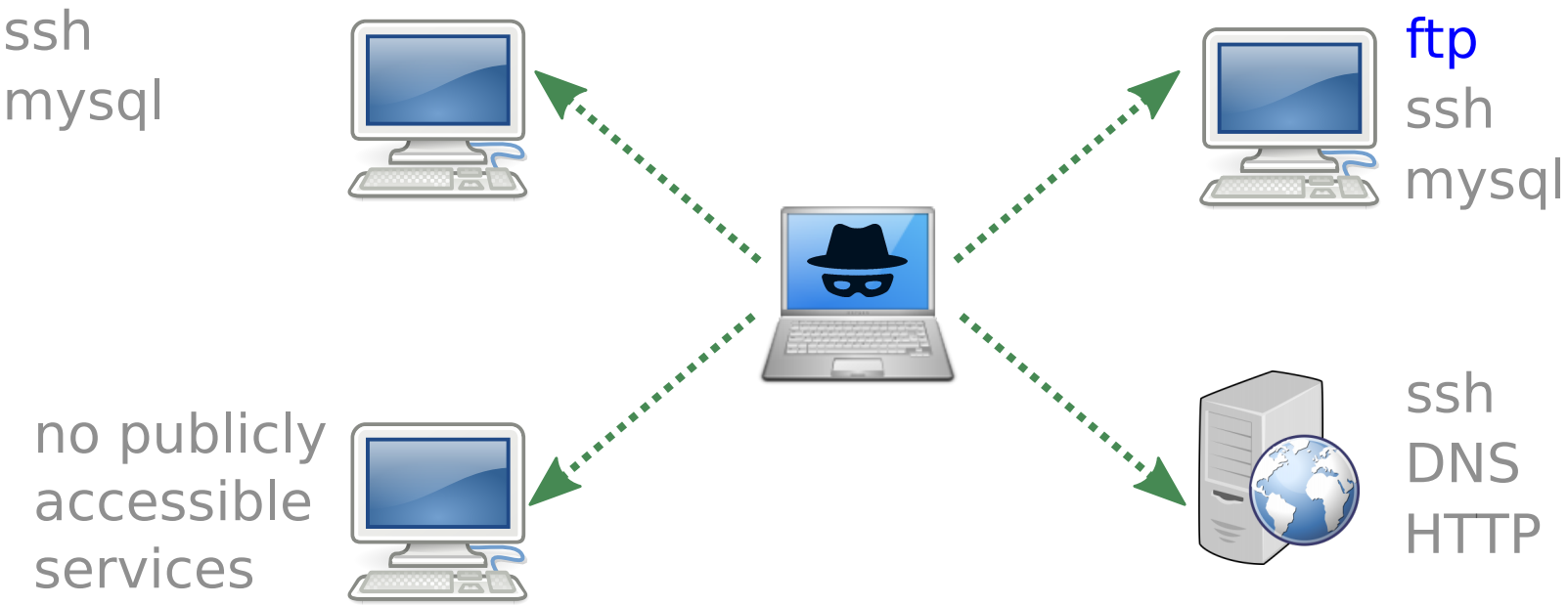


Scan for open port 21 (*ftp*)

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

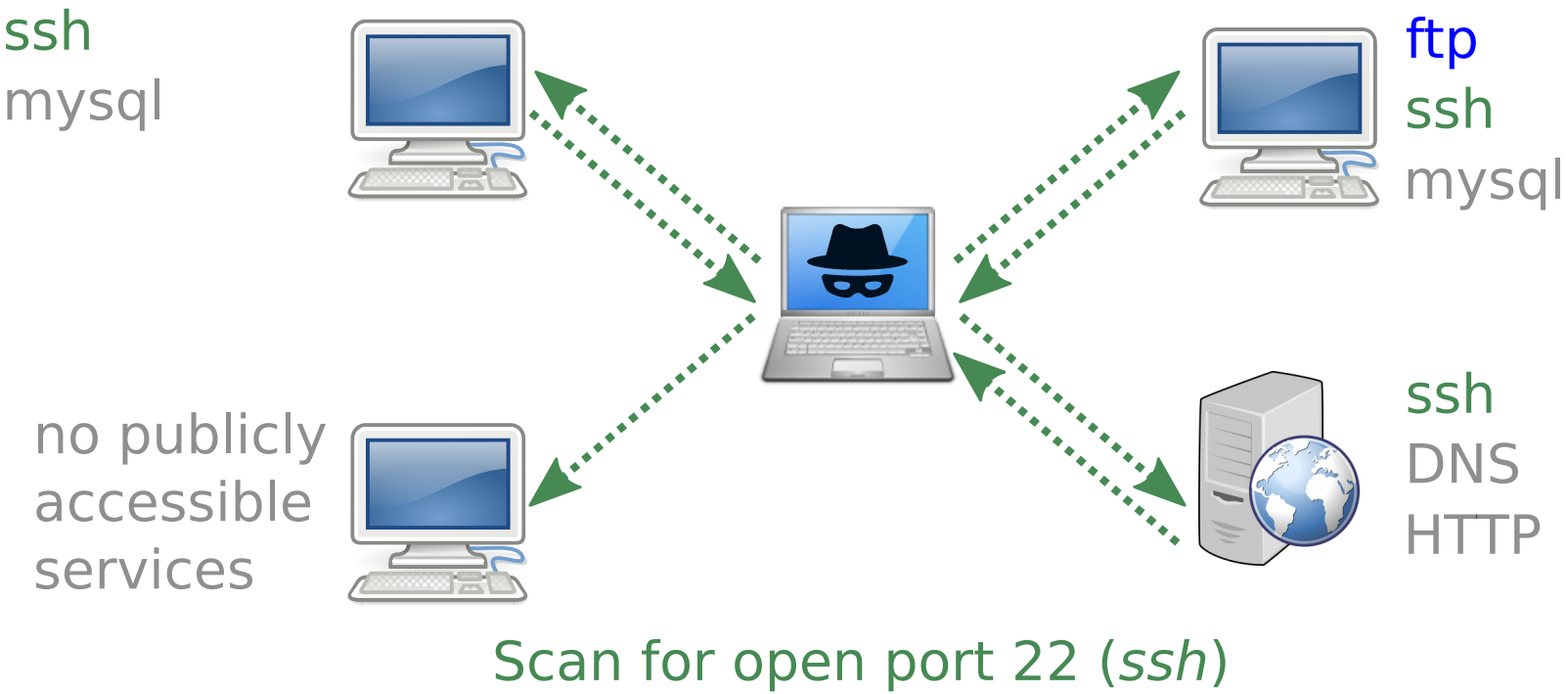
Overview of the botnet creation process:



Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

Overview of the botnet creation process:



Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

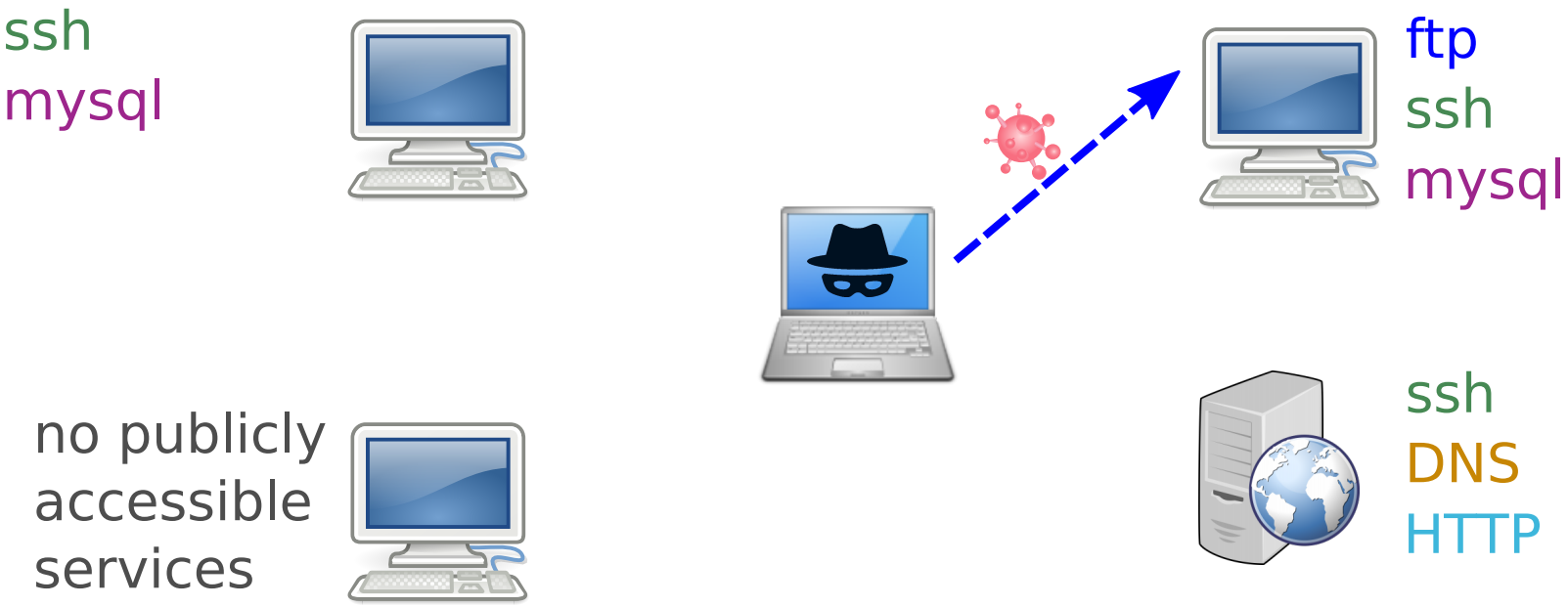
Overview of the botnet creation process:

Step#1: scan machines && identify services.

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

Overview of the botnet creation process:

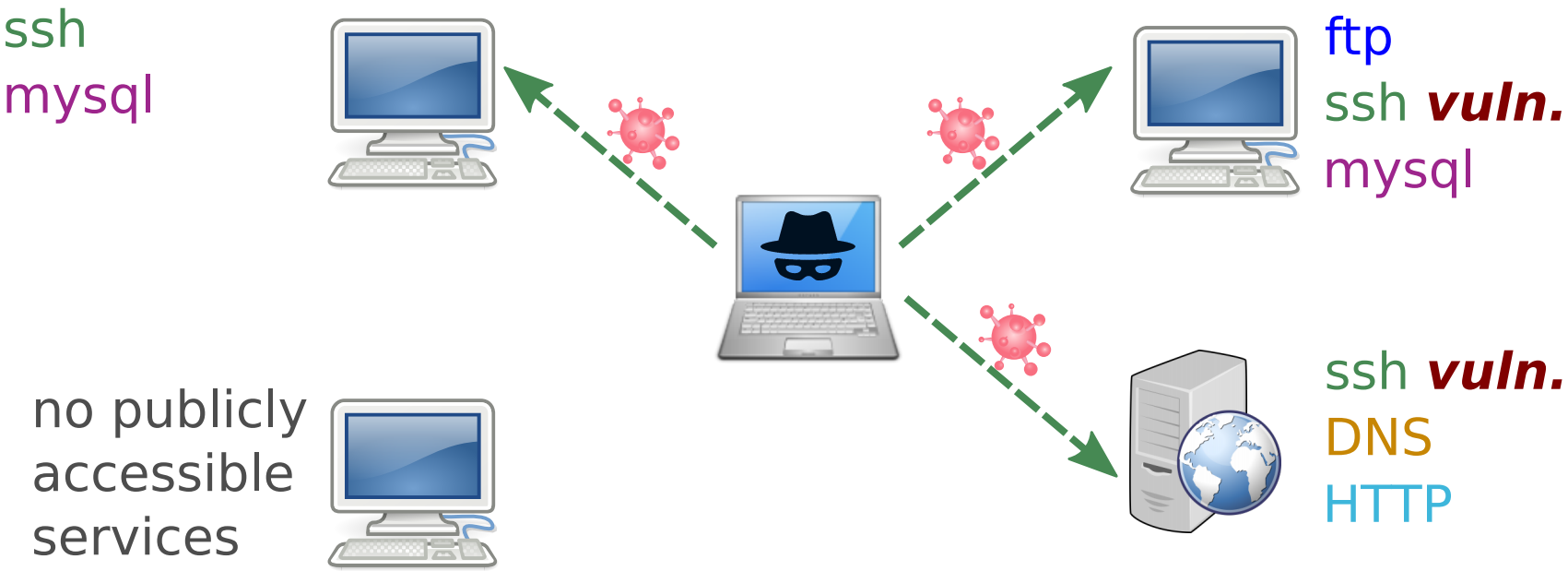


Find vulnerable versions && **exploit** service (ftp)

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

Overview of the botnet creation process:

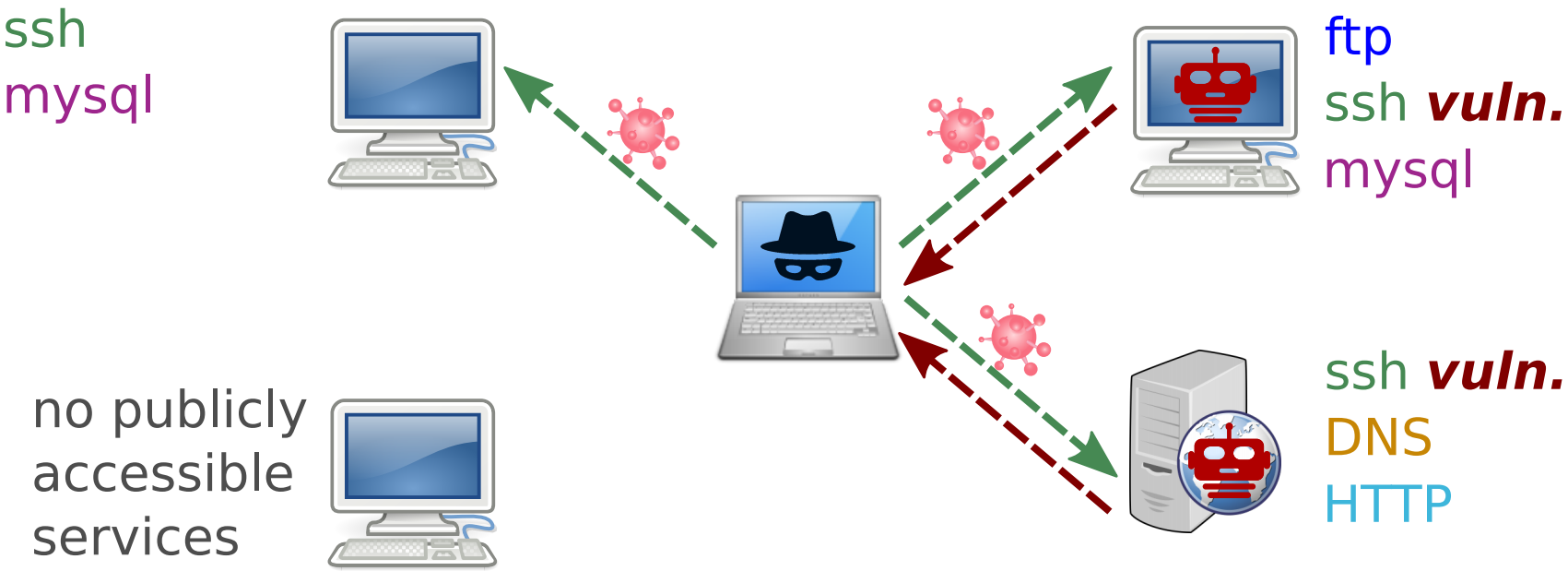


Find vulnerable versions && **exploit** service (ssh)

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

Overview of the botnet creation process:



Find vulnerable versions && **exploit** service (ssh)

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

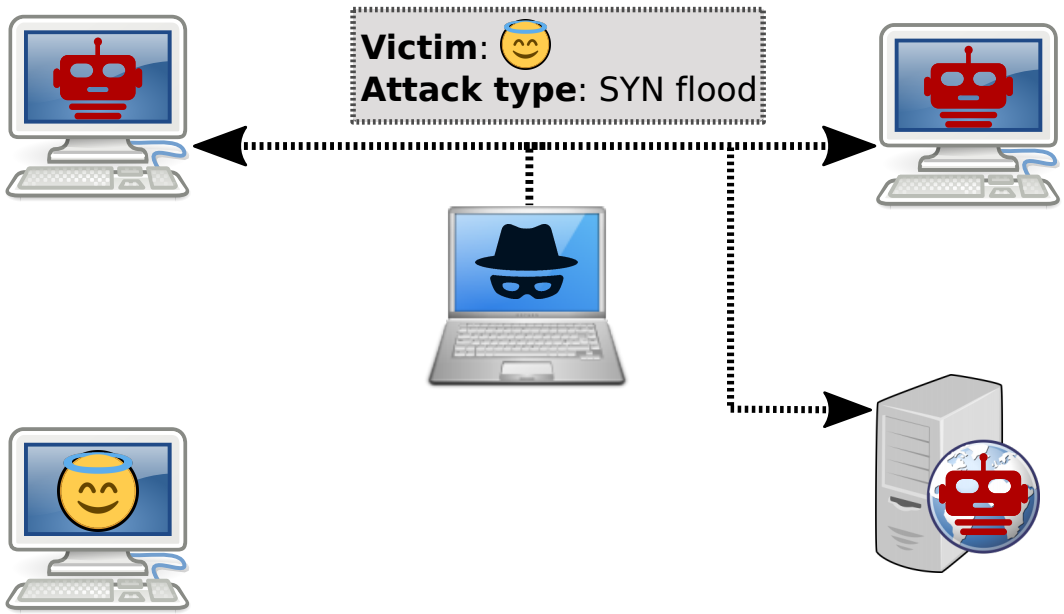
Overview of the botnet creation process:

Step#2: identify vuln. services and compromise machines.

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

Overview of the botnet creation process:

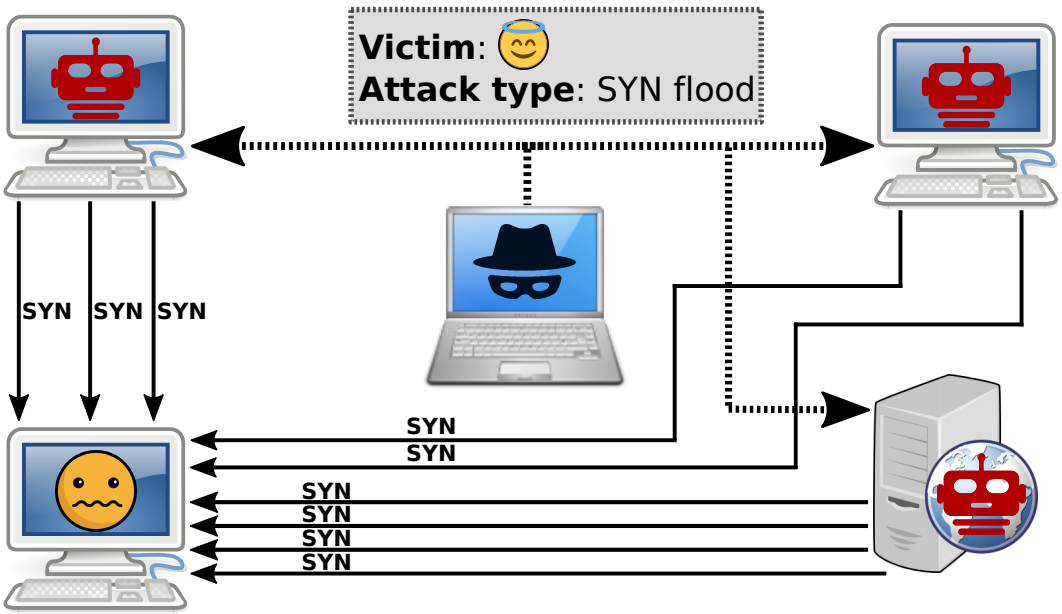


Step#3: attack the victim(s).

Botnets - I

- **Bot**: compromised machine controlled by a third-party entity.
- **Botnet**: network of (thousands) of compromised machines under the control of a single third-party entity.
- **Botmaster**: The entity/group that controls the compromised machines that are part of the botnet.

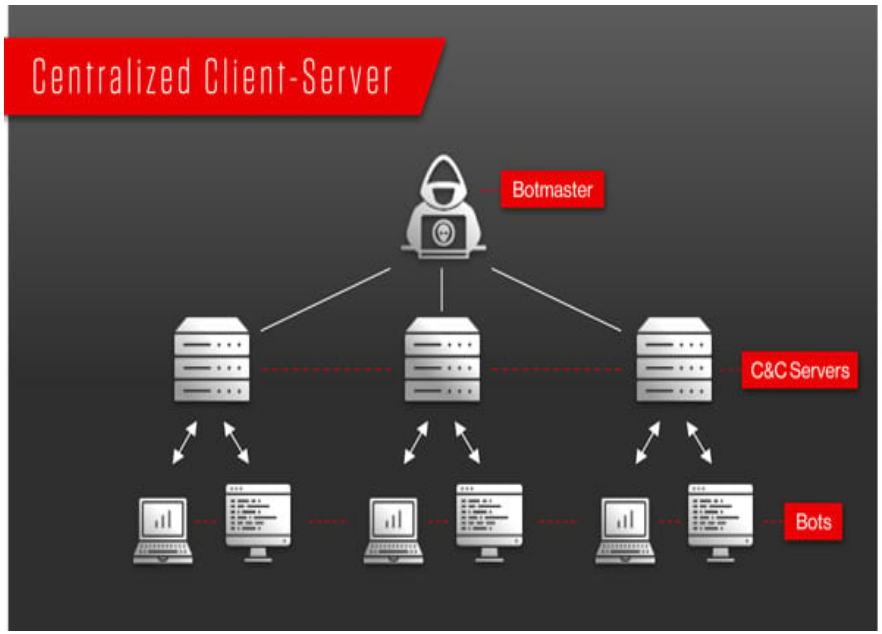
Overview of the botnet creation process:

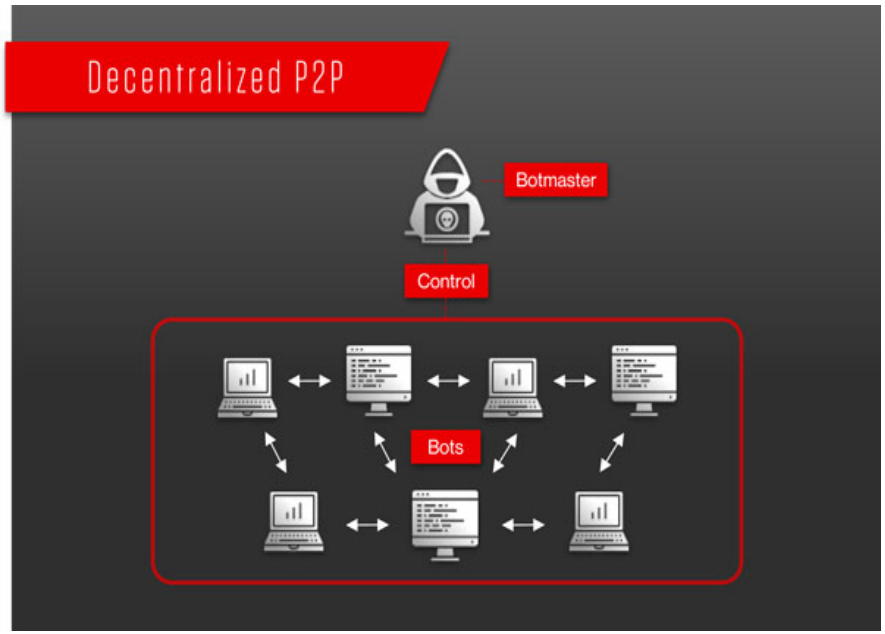


Step#3: attack the victim(s).

Botnets - II

- **Command and Control (C&C)**: the protocol that the botmaster leverages for controlling the infected machines.
  - Two communication models: *centralized* or *decentralized*.
- The bot software allows for multiple types of DoS attacks.





Source `https://www.crowdstrike.com/cybersecurity-101/botnets/`

DDoS Botnets - Lizard Stresser

- Written by the *Lizard Squad* group
  - Brute-forcing of Telnet using a list of credentials;
  - *bots*: home/university/company routers (pre-2015 malware).
    - **source code leaked** $\Longrightarrow$ *bots*: webcams and Digital Video Recording systems (post-2016 malware).
- Support for multiple DDoS attacks[2]:
  - HOLD: holds open TCP connections;
  - JUNK: random strings to TCP port(s);
  - UDP: random string on UDP port(s);
  - TCP: repeatedly send TCP packets with specified flags.
- 25th December 2014: the botnet knocked off the Sony PlayStation and Microsoft Xbox networks.

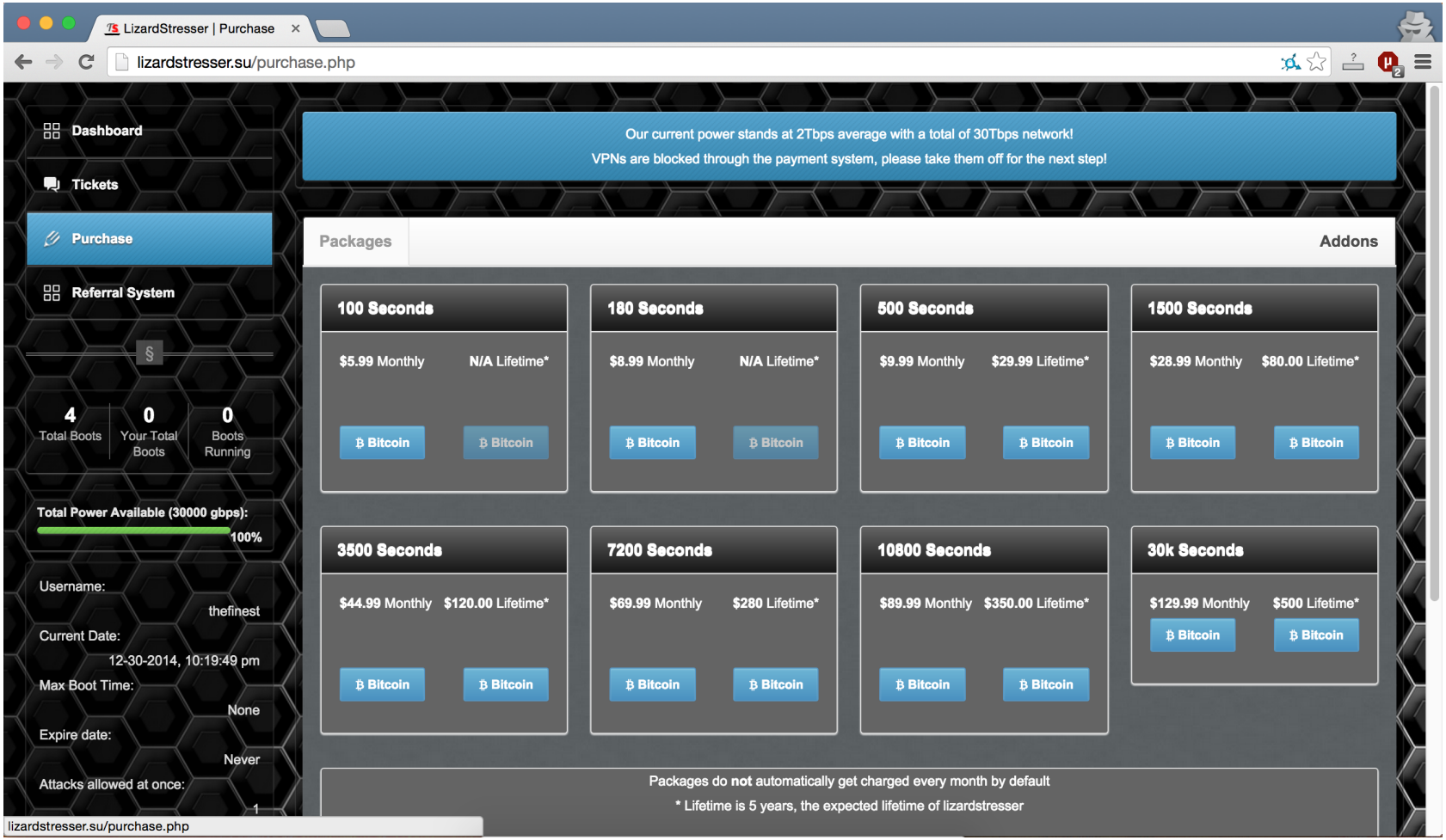
[2] `https://www.securityweek.com/botnet-uses-iot-devices-power-massive-ddos-attacks`

DDoS Botnets - Lizard Stresser

- Written by the *Lizard Squad* group
  - Brute-forcing of Telnet using a list of credentials;
  - *bots*: home/university/company routers (pre-2015 malware).
    - **source code leaked** $\Longrightarrow$ *bots*: webcams and Digital Video Recording systems (post-2016 malware).
- Support for multiple DDoS attacks[2]:
  - HOLD: holds open TCP connections;
  - JUNK: random strings to TCP port(s);
  - UDP: random string on UDP port(s);
  - TCP: repeatedly send TCP packets with specified flags.
- 25th December 2014: the botnet knocked off the Sony PlayStation and Microsoft Xbox networks. $\Rightarrow$ promotion of their new service

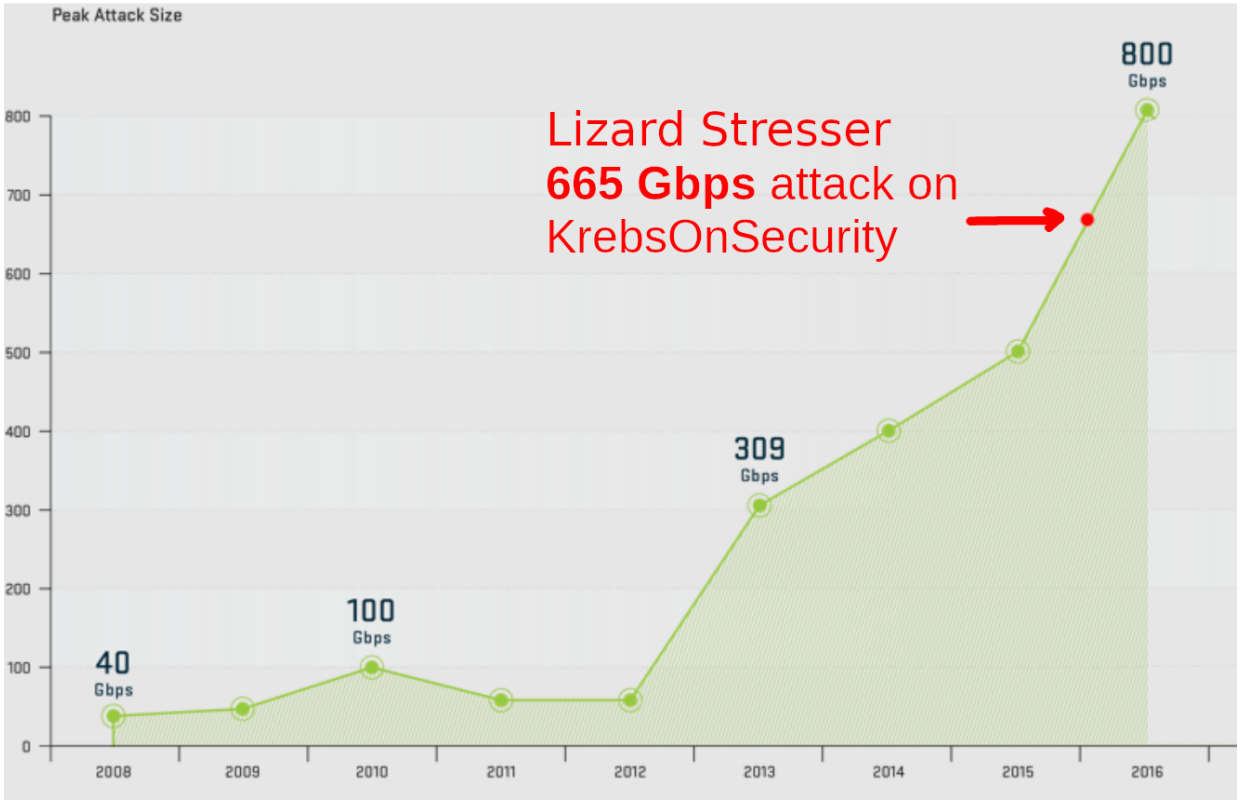
[2] https://www.securityweek.com/botnet-uses-iot-devices-power-massive-ddos-attacks

DDoS as a Service - Lizard Stresser



- Marketplace hosted on a *bulletproof provider*.
  - Multiple payment options (e.g., PayPal, card, Bitcoin)
  - Price proportional to the duration of the DDoS attack.

Largest DDoS Attack in 2016



- Brian Krebs investigated the Lizard Squad.
  - Lizard Stresser was used launch a **665 Gbps** attack on the blog KrebsOnSecurity!

Sources:
`https://blog.apnic.net/2018/04/03/the-ddos-threat-landscape-in-2017/` and
`https://krebsonsecurity.com/2014/12/lizard-kids-a-long-trail-of-fail/` and
`https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/`

DDoS with Amplification

Largest DDoS attacks:

- GitHub (2018, **1.3 Tbps**)[3]

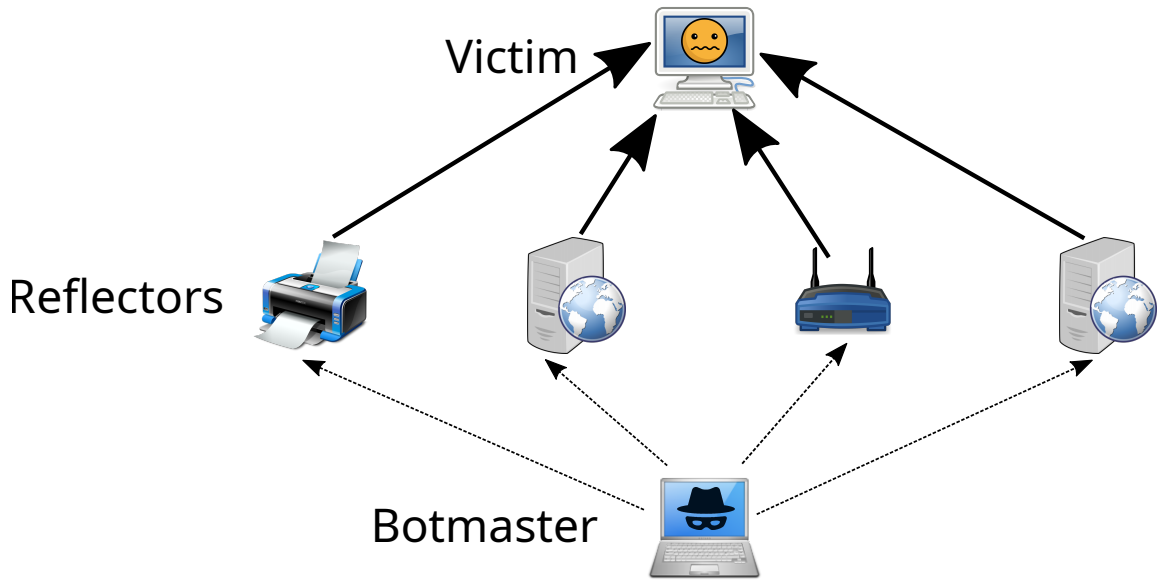
- Amazon AWS (2020, **2.3 Tbps**)[4]
- Microsoft (2021, **3.25 Tbps**)[5]

[3] https://www.wired.com/story/github-ddos-memcached/
[4] https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf
[5] https://www.thesslstore.com/blog/largest-ddos-attack-in-history/

DDoS with Amplification

Largest DDoS attacks: $\Rightarrow$ **amplification attacks**

- GitHub (2018, **1.3 Tbps**)[3]
  - no botnets involved $\rightarrow$ leveraged *memcache*;
  - 100,000 exploitable servers $\rightarrow$ attack increase by 50 times.
- Amazon AWS (2020, **2.3 Tbps**)[4]
- Microsoft (2021, **3.25 Tbps**)[5]



[3] https://www.wired.com/story/github-ddos-memcached/
[4] https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf
[5] https://www.thesslstore.com/blog/largest-ddos-attack-in-history/

Amplifier Example: Open DNS Resolvers

- **Open resolver**: DNS server that accepts remote requests.

[6] https://christian-rossow.de/publications/amplification-ndss2014.pdf.

Amplifier Example: Open DNS Resolvers

- **Open resolver**: DNS server that accepts remote requests.
  1. Reachable from *outside* the local network they serve.

[6] https://christian-rossow.de/publications/amplification-ndss2014.pdf.

Amplifier Example: Open DNS Resolvers

- **Open resolver**: DNS server that accepts remote requests.
  1. Reachable from *outside* the local network they serve.
  2. Attackers (ab)use them as *reflectors*.

[6] https://christian-rossow.de/publications/amplification-ndss2014.pdf.

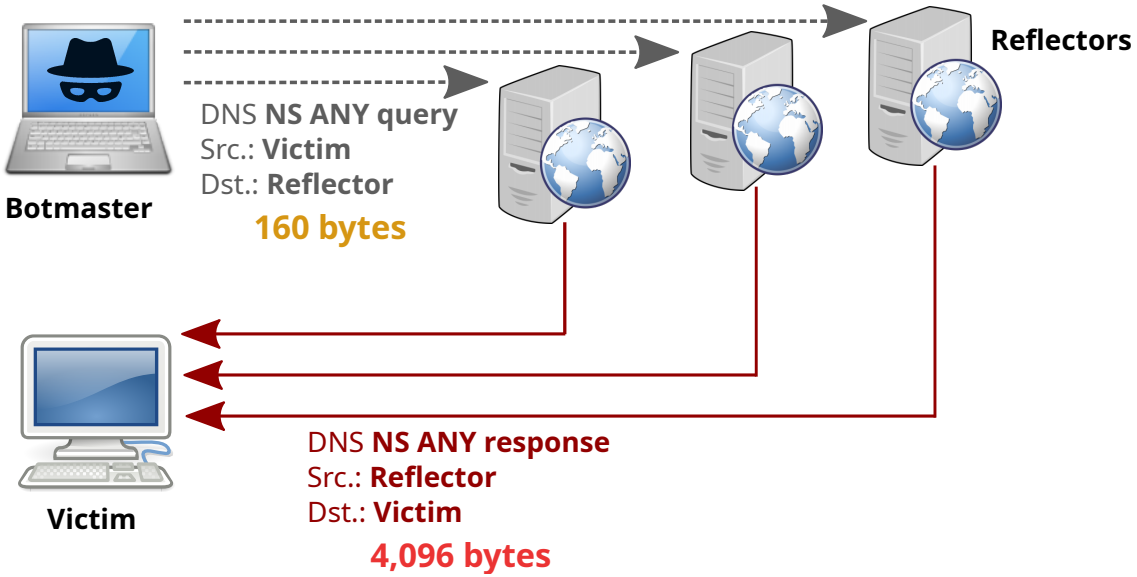
Amplifier Example: Open DNS Resolvers

- **Open resolver**: DNS server that accepts remote requests.
  1. Reachable from *outside* the local network they serve.
  2. Attackers (ab)use them as *reflectors*.
- How can we find them?

[6] https://christian-rossow.de/publications/amplification-ndss2014.pdf.

Amplifier Example: Open DNS Resolvers

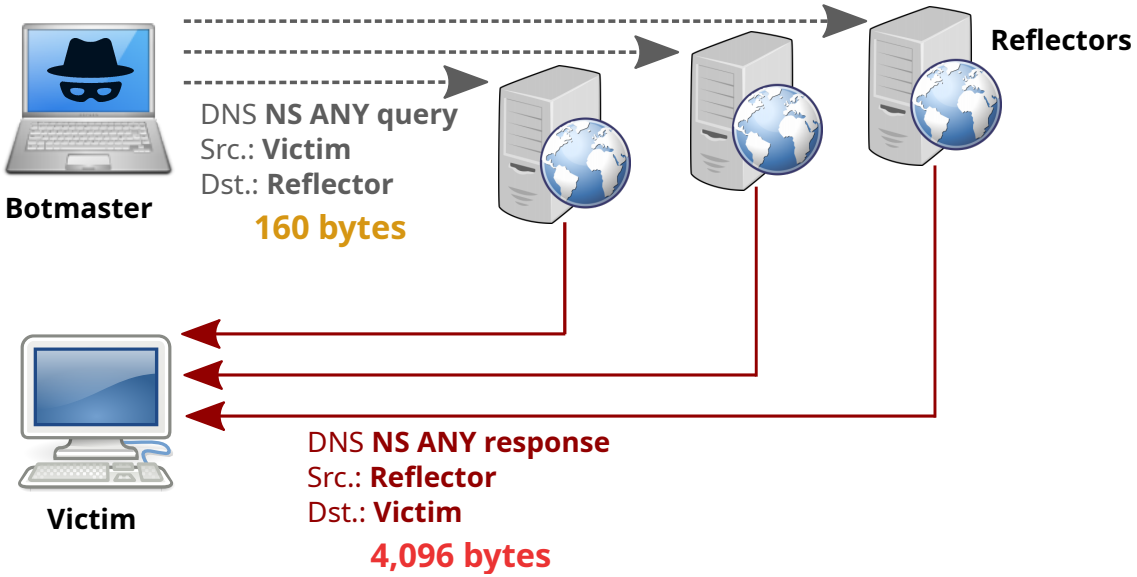
- **Open resolver**: DNS server that accepts remote requests.
  1. Reachable from *outside* the local network they serve.
  2. Attackers (ab)use them as *reflectors*.
- How can we find them? $\Rightarrow$ **Internet scanning!**



[6] https://christian-rossow.de/publications/amplification-ndss2014.pdf.

Amplifier Example: Open DNS Resolvers

- **Open resolver**: DNS server that accepts remote requests.
  1. Reachable from *outside* the local network they serve.
  2. Attackers (ab)use them as *reflectors*.
- How can we find them? $\Rightarrow$ **Internet scanning!**

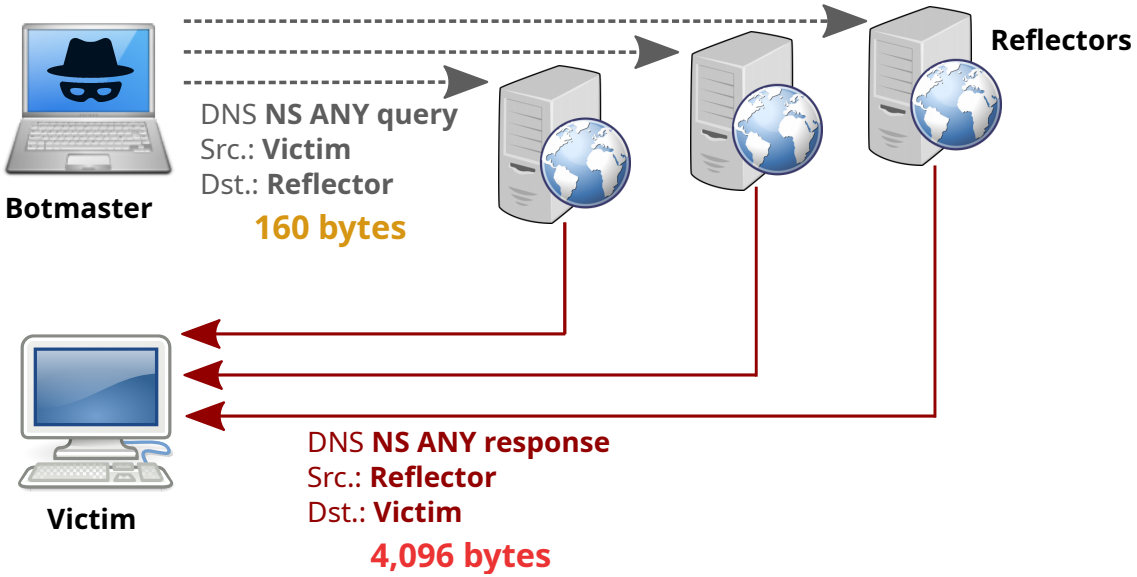


How many open resolvers are accessible on Internet?

[6] https://christian-rossow.de/publications/amplification-ndss2014.pdf.

Amplifier Example: Open DNS Resolvers

- **Open resolver**: DNS server that accepts remote requests.
  1. Reachable from *outside* the local network they serve.
  2. Attackers (ab)use them as *reflectors*.
- How can we find them? $\Rightarrow$ **Internet scanning!**

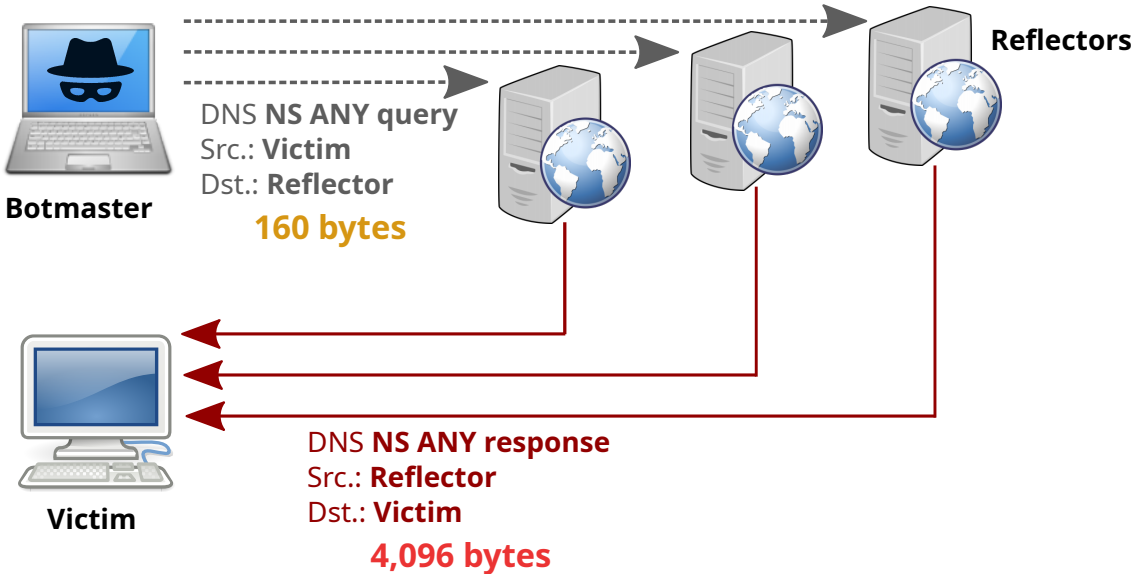


- **2006**: 580,000 open resolvers on Internet (Kaminsky-Shiffman).

[6] https://christian-rossow.de/publications/amplification-ndss2014.pdf.

Amplifier Example: Open DNS Resolvers

- **Open resolver**: DNS server that accepts remote requests.
  1. Reachable from *outside* the local network they serve.
  2. Attackers (ab)use them as *reflectors*.
- How can we find them? $\Rightarrow$ **Internet scanning!**



- **2006**: 580,000 open resolvers on Internet (Kaminsky-Shiffman).
- **2015**: 7.7M open resolvers [NDSS'14][6]

[6] https://christian-rossow.de/publications/amplification-ndss2014.pdf.

[NDSS'14] Finding Amplifiers

Protocol	Amplifiers	Tech.	t_{1k}	t_{100k}
SNMP v2	4,832,000	Scan	1.5s	148.9s
NTP	1,451,000	Scan	2.0s	195.1s
DNS_{NS}	255,819	Crawl	35.3s	3530.0s
DNS_{OR}	7,782,000	Scan	0.9s	92.5s
NetBios	2,108,000	Scan	3.4s	341.5s
SSDP	3,704,000	Scan	1.9s	193.5s
CharGen	89,000	Scan	80.6s	n/a
QOTD	32,000	Scan	228.2s	n/a
BitTorrent	5,066,635	Crawl	0.9s	63.6s
Kad	232,012	Crawl	0.9s	108.0s
Quake 3	1,059	Master	0.6s	n/a
Steam	167,886	Master	1.3s	137.1s
ZAv2	27,939	Crawl	1.5s	n/a
Sality	12,714	Crawl	4.7s	n/a
Gameover	2,023	Crawl	168.5s	n/a

TABLE II: Number of amplifiers per protocol, the technique we used to obtain the amplifiers, and the time it took to identify 1000 (t_{1k}) and 100,000 (t_{100k}) amplifiers, respectively.

Source: https://christian-rossow.de/publications/amplification-ndss2014.pdf

[NDSS'14] Finding Amplifiers

Protocol	Amplifiers	Tech.	t_{1k}	t_{100k}
SNMP v2	4,832,000	Scan	1.5s	148.9s
NTP	1,451,000	Scan	2.0s	195.1s
DNS_{NS}	255,819	Crawl	35.3s	3530.0s
DNS_{OR}	7,782,000	Scan	0.9s	92.5s
NetBios	2,108,000	Scan	3.4s	341.5s
SSDP	3,704,000	Scan	1.9s	193.5s
CharGen	89,000	Scan	80.6s	n/a
QOTD	32,000	Scan	228.2s	n/a
BitTorrent	5,066,635	Crawl	0.9s	63.6s
Kad	232,012	Crawl	0.9s	108.0s
Quake 3	1,059	Master	0.6s	n/a
Steam	167,886	Master	1.3s	137.1s
ZAv2	27,939	Crawl	1.5s	n/a
Sality	12,714	Crawl	4.7s	n/a
Gameover	2,023	Crawl	168.5s	n/a

TABLE II: Number of amplifiers per protocol, the technique we used to obtain the amplifiers, and the time it took to identify 1000 (t_{1k}) and 100,000 (t_{100k}) amplifiers, respectively.

Source: https://christian-rossow.de/publications/amplification-ndss2014.pdf

[NDSS'14] Amplification Factors

Protocol	BAF *all*	BAF 50%	BAF 10%	PAF *all*	Scenario
SNMP v2	6.3	8.6	11.3	1.00	*GetBulk* request
NTP	556.9	1083.2	4670.0	10.61	Request "monlist" statistics
DNS_{NS}	54.6	76.7	98.3	2.08	ANY lookup at author. NS
DNS_{OR}	28.7	41.2	64.1	1.32	ANY lookup at open resolv.
NetBios	3.8	4.5	4.9	1.00	Name resolution
SSDP	30.8	40.4	75.9	9.92	*SEARCH* request
CharGen	358.8	n/a	n/a	1.00	Character generation request
QOTD	140.3	n/a	n/a	1.00	Quote request
BitTorrent	3.8	5.3	10.3	1.58	File search
Kad	16.3	21.5	22.7	1.00	Peer list exchange
Quake 3	63.9	74.9	82.8	1.01	Server info exchange
Steam	5.5	6.9	14.7	1.12	Server info exchange
ZAv2	36.0	36.6	41.1	1.02	Peer list and cmd exchange
Sality	37.3	37.9	38.4	1.00	URL list exchange
Gameover	45.4	45.9	46.2	5.39	Peer and proxy exchange

TABLE III: Bandwidth amplifier factors per protocols. *all* shows the average BAF of all amplifiers, 50% and 10% show the average BAF when using the worst 50% or 10% of the amplifiers, respectively.

Source: `https://christian-rossow.de/publications/amplification-ndss2014.pdf`

Estonia DDoS Attack (2007)

- Trigger: relocation of the Bronze Soldier in Tallin.
  - (Estonian's) Russians disagree..
  - On the 27th of April 2007, a 22-days DoS started on several Estonian websites...
- Attack details:
  - ICMP and TCP SYN floods;
  - malicious traffic originated from outside Estonia $\Rightarrow$ ISPs block foreign traffic;
  - targeted critical systems: telephony and financial;
  - attack instruction published on Russian websites.

Sources:
https://en.wikipedia.org/wiki/2007_cyberattacks_on_Estonia and
https://ccdcoe.org/uploads/2018/10/Ottis2008_AnalysisOf2007FromTheInformationWarfarePerspective.pdf

Estonia DDoS Attack (2007)

- Trigger: relocation of the Bronze Soldier in Tallin.
  - (Estonian's) Russians disagree..
  - On the 27th of April 2007, a 22-days DoS started on several Estonian websites...
- Attack details:
  - ICMP and TCP SYN floods;
  - malicious traffic originated from outside Estonia
    $\Rightarrow$ ISPs block foreign traffic;
  - targeted critical systems: telephony and financial;
  - attack instruction published on Russian websites.
- Attribution is hard!
  - State-sponsored cyberwarfare?
    $\Rightarrow$ Many attacks originated from Russia.
  - Cyber-riot?
    $\Rightarrow$ One Estonian student found guilty and fined.

Sources:
https://en.wikipedia.org/wiki/2007_cyberattacks_on_Estonia and
https://ccdcoe.org/uploads/2018/10/Ottis2008_AnalysisOf2007FromTheInformationWarfarePerspective.pdf

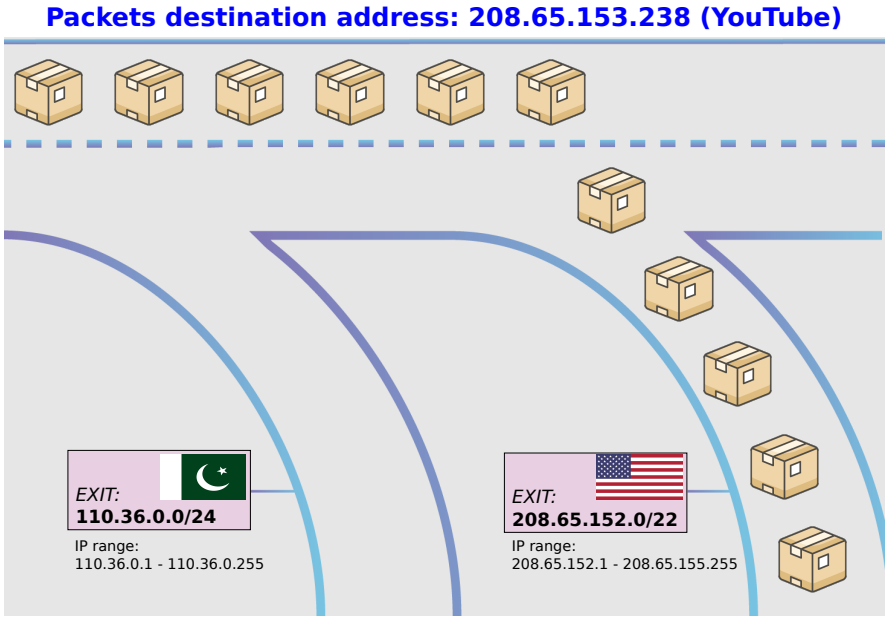
DoS via Routing Attacks

- **BGP hijacking**[7] relies on injecting a fake BGP route to:
  A. drop all traffic for victim's site to attacker's site;
  B. redirect traffic through attacker's network.

[7] https://en.wikipedia.org/wiki/BGP_hijacking
[8] https://wwww.wired.com/2014/08/isp-bitcoin-theft/

DoS via Routing Attacks

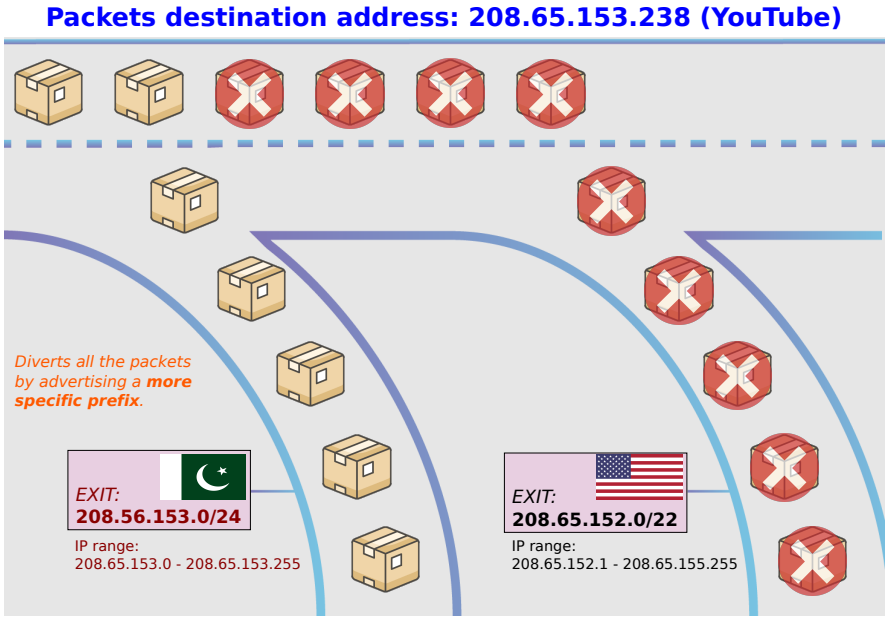
- **BGP hijacking**[7] relies on injecting a fake BGP route to:
  A. drop all traffic for victim's site to attacker's site;
  B. redirect traffic through attacker's network.
- February 2008: YouTube Sinkholing
  - Pakistan telecom advertised a BGP path as subset of the YouTube range.
    $\Rightarrow$ *Most Internet users thought YouTube was in Pakistan.*



[7] https://en.wikipedia.org/wiki/BGP_hijacking
[8] https://wwww.wired.com/2014/08/isp-bitcoin-theft/

DoS via Routing Attacks

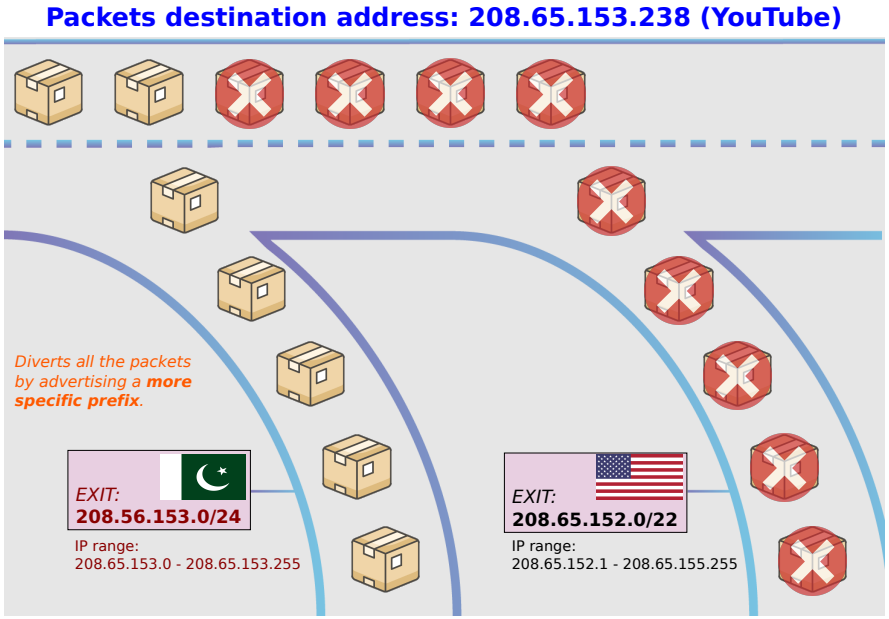
- **BGP hijacking**[7] relies on injecting a fake BGP route to:
  A. drop all traffic for victim's site to attacker's site;
  B. redirect traffic through attacker's network.
- February 2008: YouTube Sinkholing
  - Pakistan telecom advertised a BGP path as subset of the YouTube range.
    $\Rightarrow$ *Most Internet users thought YouTube was in Pakistan.*



[7] https://en.wikipedia.org/wiki/BGP_hijacking
[8] https://wwww.wired.com/2014/08/isp-bitcoin-theft/

DoS via Routing Attacks

- **BGP hijacking**[7] relies on injecting a fake BGP route to:
  A. drop all traffic for victim's site to attacker's site;
  B. redirect traffic through attacker's network.
- February 2008: YouTube Sinkholing
  - Pakistan telecom advertised a BGP path as subset of the YouTube range.
    $\Rightarrow$ *Most Internet users thought YouTube was in Pakistan.*

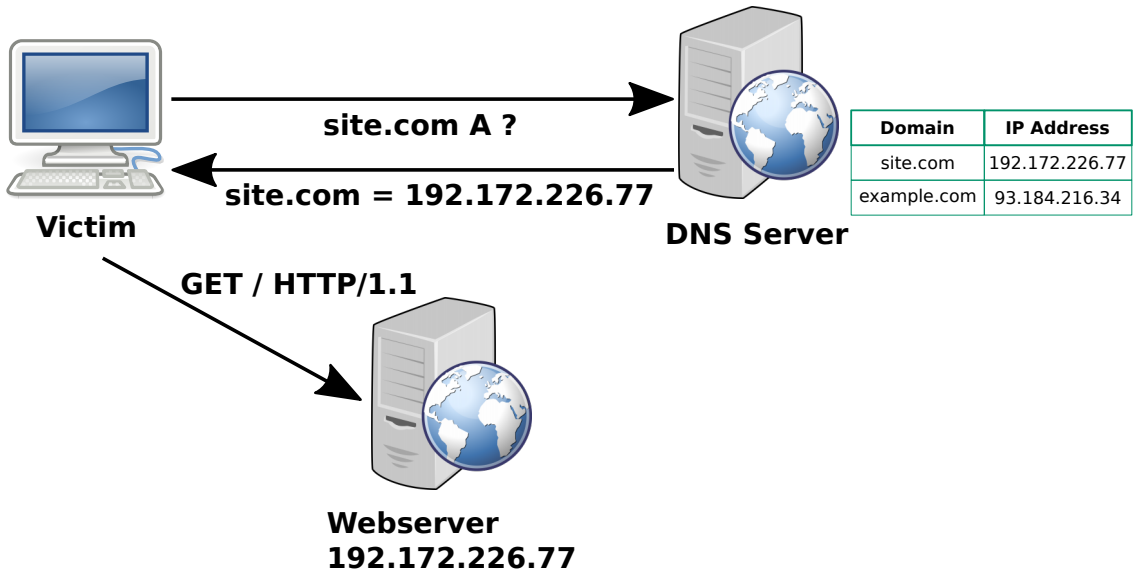


- 2014 **Bitcoin BGP hijacking attack**[8]
  - the attack lasted over 2 months, allowing to harvest $83,000 in cryptocurrency

[7] https://en.wikipedia.org/wiki/BGP_hijacking
[8] https://wwww.wired.com/2014/08/isp-bitcoin-theft/

DNS DDoS

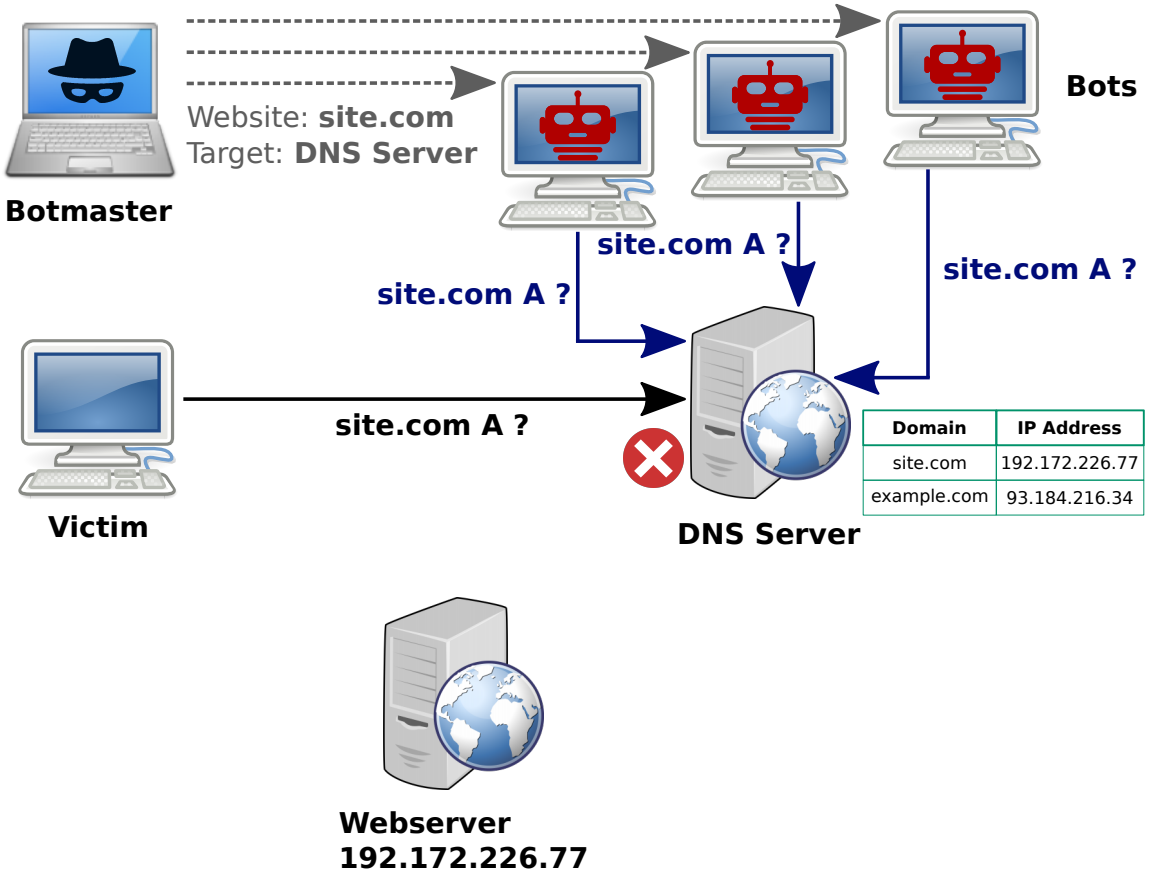
- Idea: instead of taking down the webserver, attack its DNS server



[9] https://en.wikipedia.org/wiki/DDoS_attacks_on_Dyn

DNS DDoS

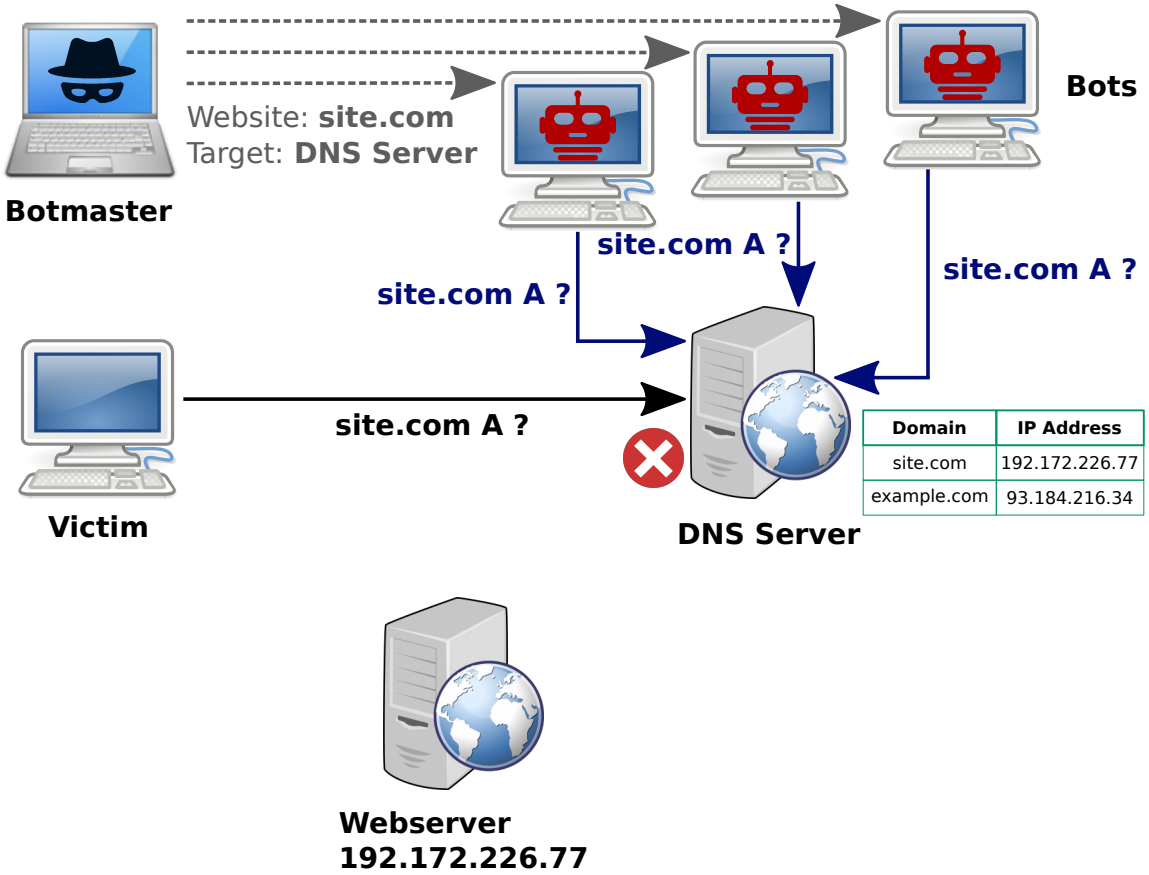
- Idea: instead of taking down the webserver, attack its DNS server
  ⇒ Users will not be able to resolve the domain.



[9] https://en.wikipedia.org/wiki/DDoS_attacks_on_Dyn

DNS DDoS

- Idea: instead of taking down the webserver, attack its DNS server
  ⇒ Users will not be able to resolve the domain.



- Mirai-infected devices attack Dyn (2016)
  - Affected services: Amazon, PayPal, Slack, Twitter, CNN, NYT[9].

[9] https://en.wikipedia.org/wiki/DDoS_attacks_on_Dyn

JavaScript DDoS Attack

- Application layer attack.
- Attack: inject DDoS JavaScript into website's response.
  $\Rightarrow$ Everyone visiting the (malicious) website becomes a DDoS bot.

How to serve to clients malicious JS code:

```
function imgflood() {
  var TARGET = 'victim-website.com'
  var URI = '/index.php?'
  var pic = new Image()
  var rand = Math.floor(Math.random() * 1000)
  pic.src = 'http://'+TARGET+URI+rand+'=val'
}
setInterval(imgflood, 10)
```

[a] https://www.eff.org/deeplinks/2014/11/verizon-x-uidh

Source: https://blog.cloudflare.com/an-introduction-to-javascript-based-ddos/

JavaScript DDoS Attack

- Application layer attack.
- Attack: inject DDoS JavaScript into website's response.
  $\Rightarrow$ Everyone visiting the (malicious) website becomes a DDoS bot.

```
function imgflood() {
  var TARGET = 'victim-website.com'
  var URI = '/index.php?'
  var pic = new Image()
  var rand = Math.floor(Math.random() * 1000)
  pic.src = 'http://'+TARGET+URI+rand+'=val'
}
setInterval(imgflood, 10)
```

How to serve to clients malicious JS code:

1. create a dedicated website for the attack;

[a] https://www.eff.org/deeplinks/2014/11/verizon-x-uidh

Source: https://blog.cloudflare.com/an-introduction-to-javascript-based-ddos/

JavaScript DDoS Attack

- Application layer attack.
- Attack: inject DDoS JavaScript into website's response.
  $\Rightarrow$ Everyone visiting the (malicious) website becomes a DDoS bot.

```
function imgflood() {
  var TARGET = 'victim-website.com'
  var URI = '/index.php?'
  var pic = new Image()
  var rand = Math.floor(Math.random() * 1000)
  pic.src = 'http://'+TARGET+URI+rand+'=val'
}
setInterval(imgflood, 10)
```

How to serve to clients malicious JS code:

1. create a dedicated website for the attack;
2. compromise servers hosting popular libraries;

[a] https://www.eff.org/deeplinks/2014/11/verizon-x-uidh

Source: https://blog.cloudflare.com/an-introduction-to-javascript-based-ddos/

JavaScript DDoS Attack

- Application layer attack.
- Attack: inject DDoS JavaScript into website's response.
  $\Rightarrow$ Everyone visiting the (malicious) website becomes a DDoS bot.

```
function imgflood() {
  var TARGET = 'victim-website.com'
  var URI = '/index.php?'
  var pic = new Image()
  var rand = Math.floor(Math.random() * 1000)
  pic.src = 'http://'+TARGET+URI+rand+'=val'
}
setInterval(imgflood, 10)
```

How to serve to clients malicious JS code:

1. create a dedicated website for the attack;
2. compromise servers hosting popular libraries;
3. Man-In-The-Middle attacks[a]
   $\rightarrow$ solution?

[a] https://www.eff.org/deeplinks/2014/11/verizon-x-uidh

Source: https://blog.cloudflare.com/an-introduction-to-javascript-based-ddos/

JavaScript DDoS Attack

- Application layer attack.
- Attack: inject DDoS JavaScript into website's response.
  $\Rightarrow$ Everyone visiting the (malicious) website becomes a DDoS bot.

```
function imgflood() {
  var TARGET = 'victim-website.com'
  var URI = '/index.php?'
  var pic = new Image()
  var rand = Math.floor(Math.random() * 1000)
  pic.src = 'http://'+TARGET+URI+rand+'=val'
}
setInterval(imgflood, 10)
```

How to serve to clients malicious JS code:

1. create a dedicated website for the attack;
2. compromise servers hosting popular libraries;
3. Man-In-The-Middle attacks[a]
   $\rightarrow$ solution? encryption + authentication

[a] https://www.eff.org/deeplinks/2014/11/verizon-x-uidh

Source: https://blog.cloudflare.com/an-introduction-to-javascript-based-ddos/

DoS Defenses

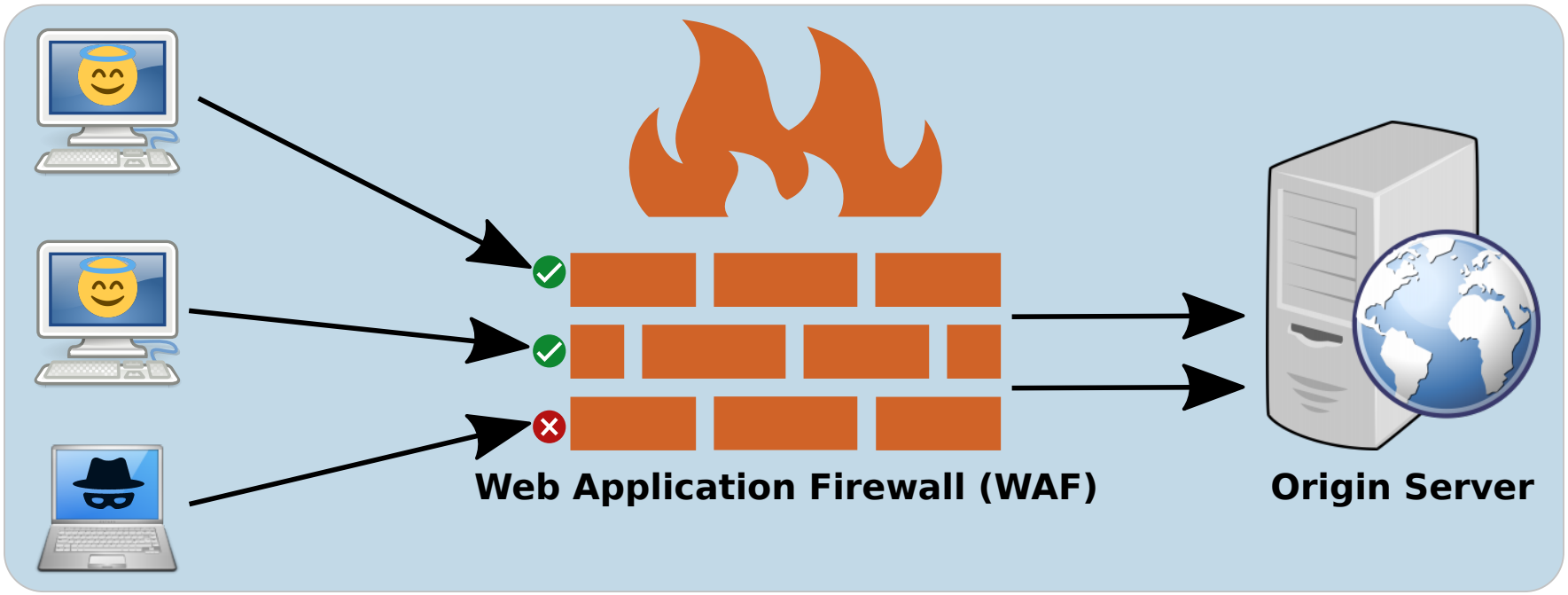
- Ingress Filtering
- SYN Cookies

DoS Defenses

- Ingress Filtering
- SYN Cookies

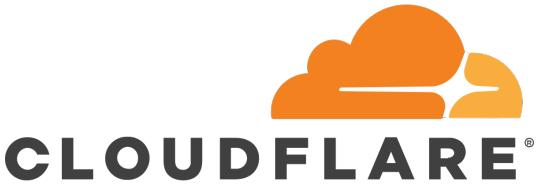
Web Application Firewall (WAF)

- WAF: a reverse proxy that intercepts HTTP traffic.
  - Filters malicious traffic (e.g., DDoS).
  - Forward valid requests to the website.



Dedicated DoS Protection Services

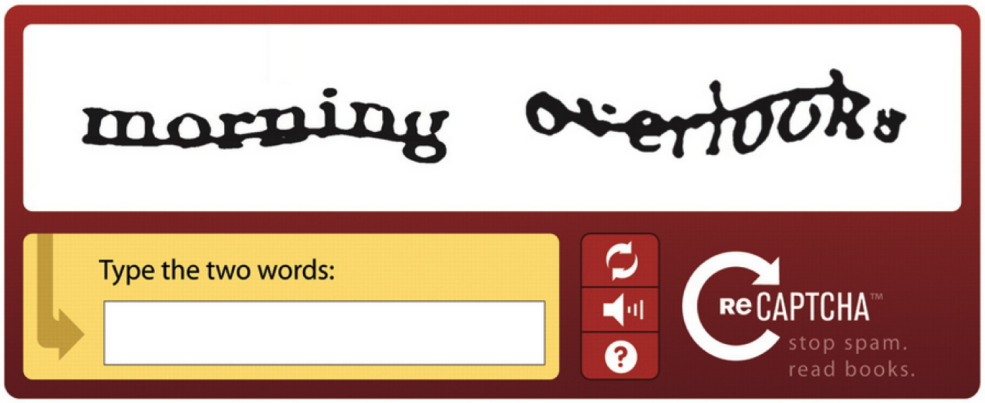
- Usually offered in conjunction with CDN service.
- These services route users to the closest WAF/CDN server that:
  1. filters malicious traffic;
  2. caches static content.

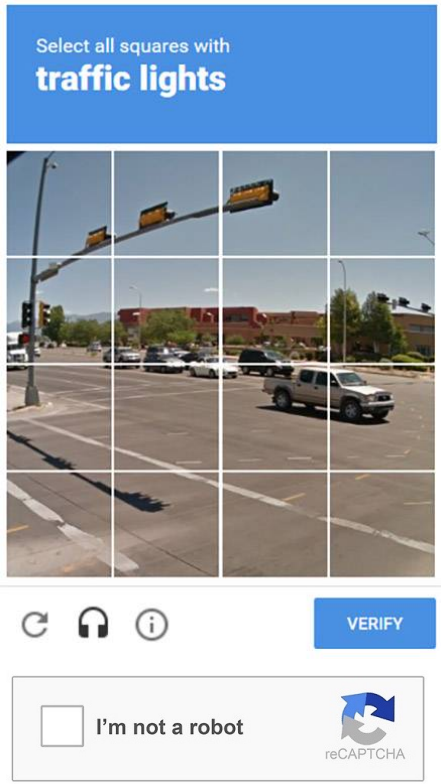


DoS protection service providers:
`https://www.esecurityplanet.com/products/distributed-denial-of-service-ddos-protection-vendors/`

CAPTCHAs

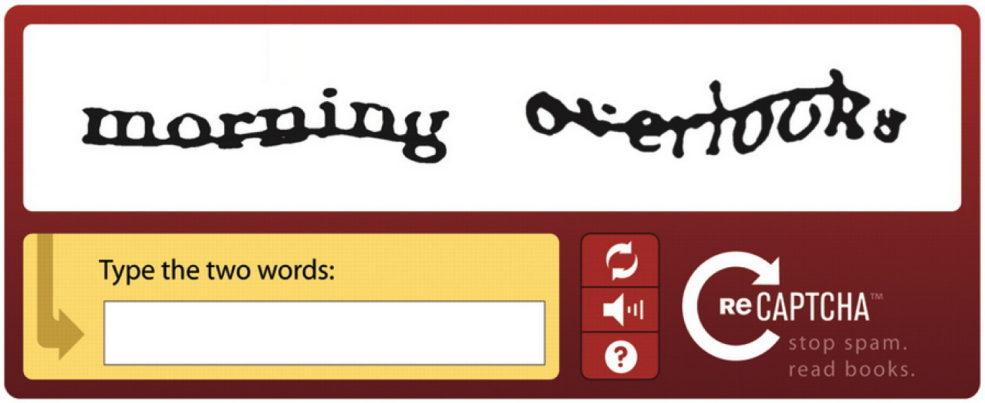
- Idea: verify that the connection comes **from a human**.
  - Differentiate *humans* and *bots*.
- Load increases $\Rightarrow$ serve a CAPTCHA.
  - Accept *only* connections with a correct answer.

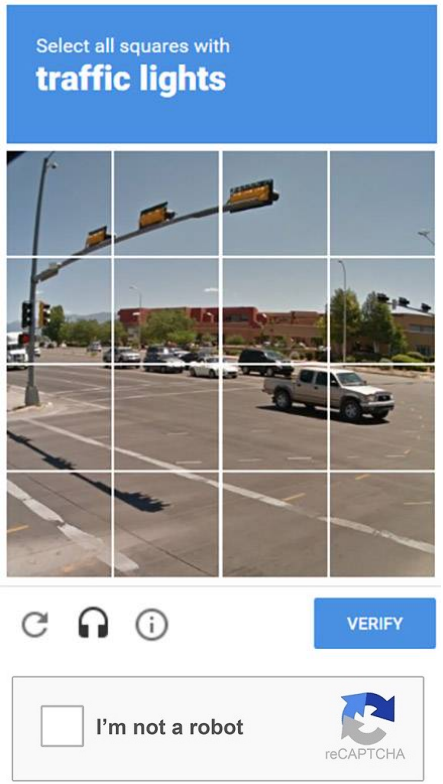




CAPTCHAs

- Idea: verify that the connection comes **from a human**.
  - Differentiate *humans* and *bots*.
- Load increases $\Rightarrow$ serve a CAPTCHA.
  - Accept *only* connections with a correct answer.

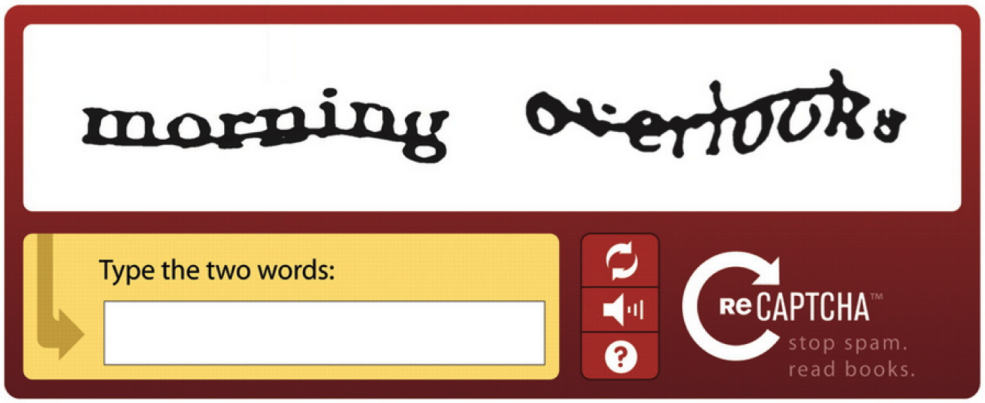


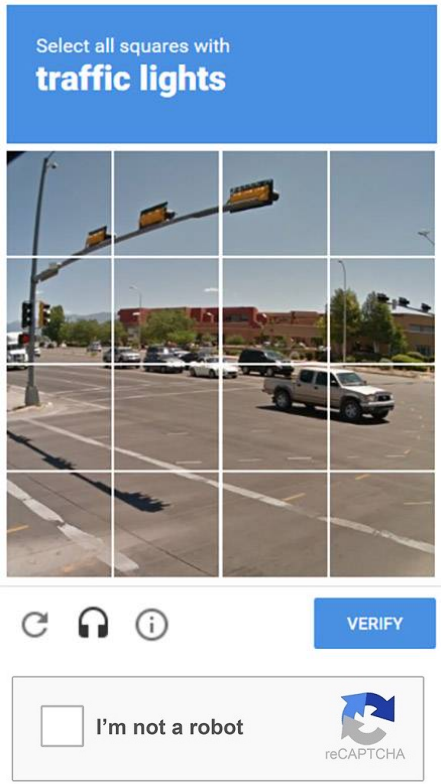


- **Limitations:**

CAPTCHAs

- Idea: verify that the connection comes **from a human**.
  - Differentiate *humans* and *bots*.
- Load increases ⇒ serve a CAPTCHA.
  - Accept *only* connections with a correct answer.

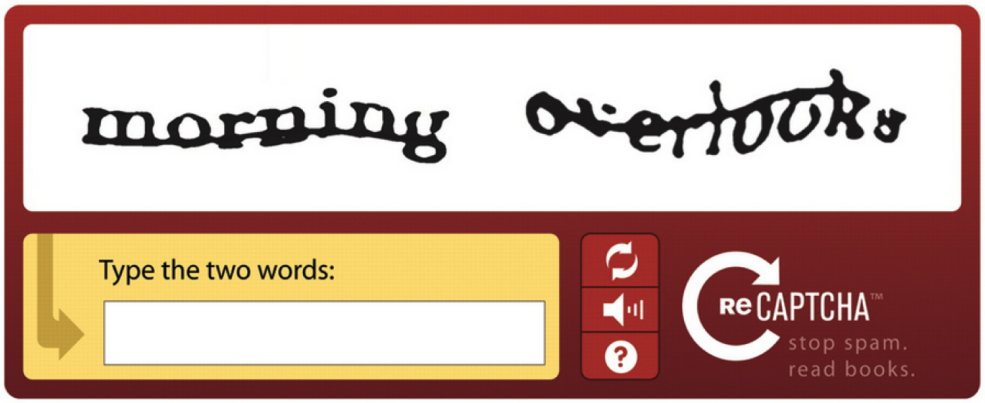


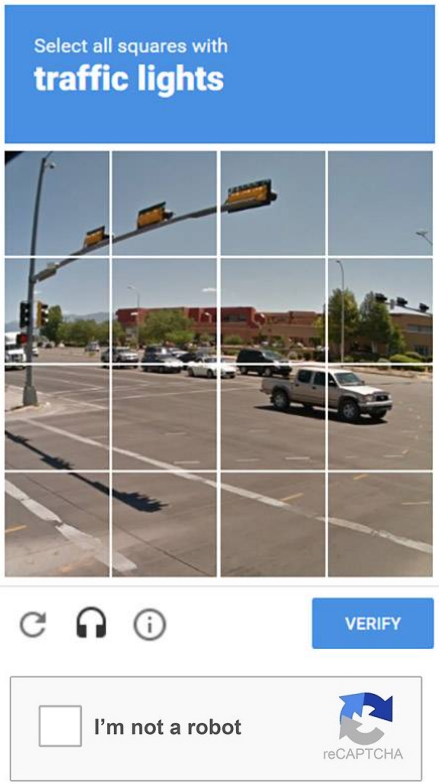


- **Limitations:**
  - works only for *application layer* attacks;

CAPTCHAs

- Idea: verify that the connection comes **from a human**.
  - Differentiate *humans* and *bots*.
- Load increases ⇒ serve a CAPTCHA.
  - Accept *only* connections with a correct answer.

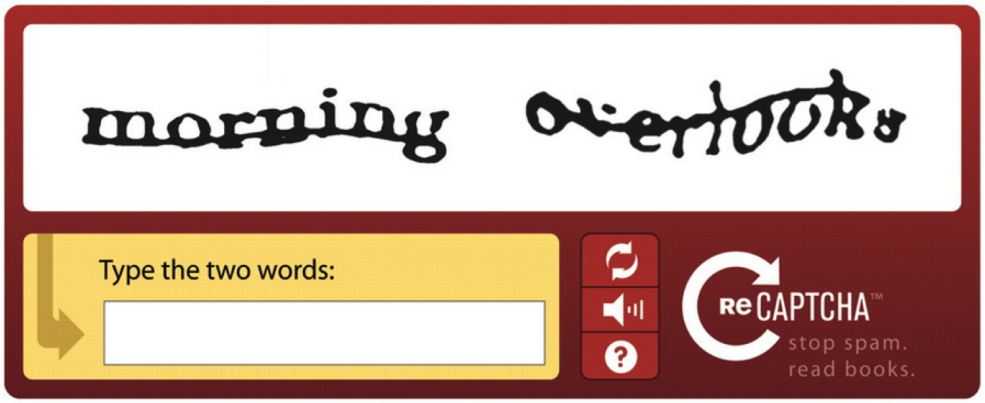


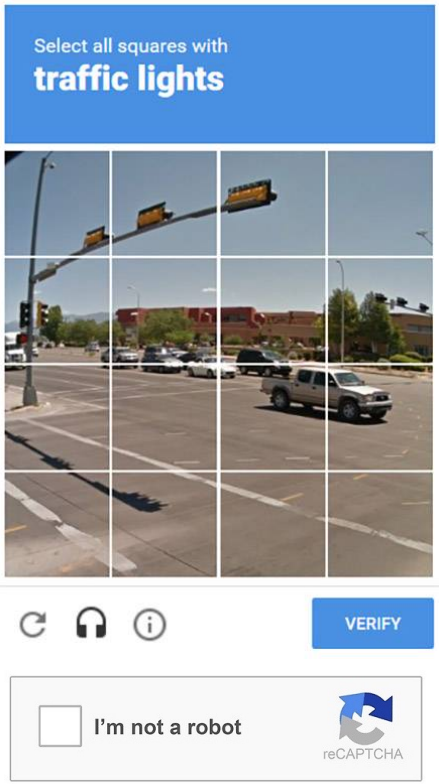


- **Limitations:**
  - works only for *application layer* attacks;
  - no defense against flooding;

CAPTCHAs

- Idea: verify that the connection comes **from a human**.
  - Differentiate *humans* and *bots*.
- Load increases ⇒ serve a CAPTCHA.
  - Accept *only* connections with a correct answer.



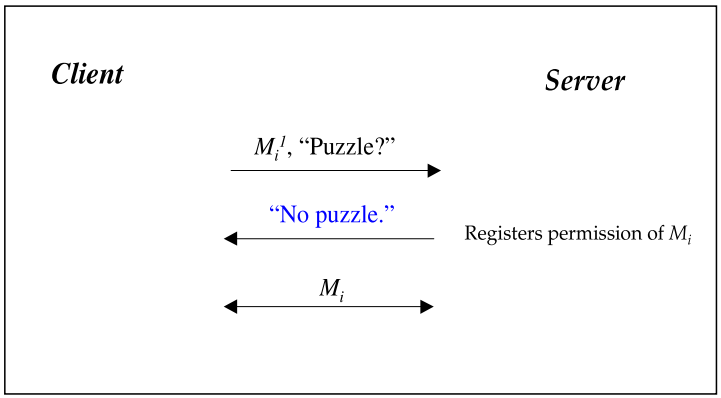


- **Limitations:**
  - works only for *application layer* attacks;
  - no defense against flooding;
  - False Positives (e.g., VPN, NAT).

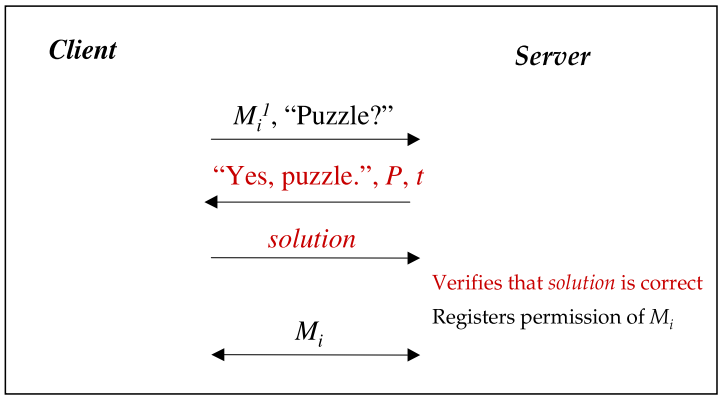
Client Puzzle

- Idea: request client(s) to *do some work*, to slow down attacker(s).
  1. Moderately expensive to compute (client side).
  2. Cheap to verify (server side).
- Block in case the provided solution is incorrect.

Webserver operating normally:



Webserver under attack:

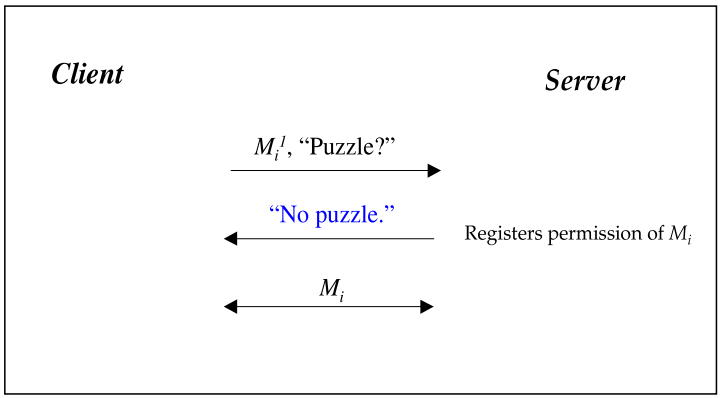


[10] http://www.arijuels.com/wp-content/uploads/2013/09/JB99.pdf

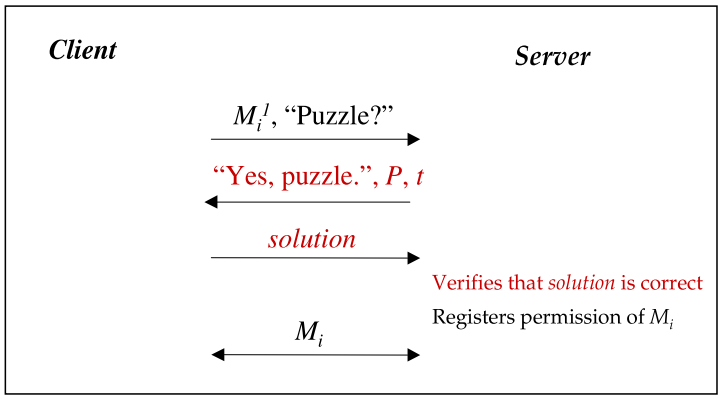
Client Puzzle

- Idea: request client(s) to *do some work*, to slow down attacker(s).
  1. Moderately expensive to compute (client side).
  2. Cheap to verify (server side).
- Block in case the provided solution is incorrect.

Webserver operating normally:



Webserver under attack:



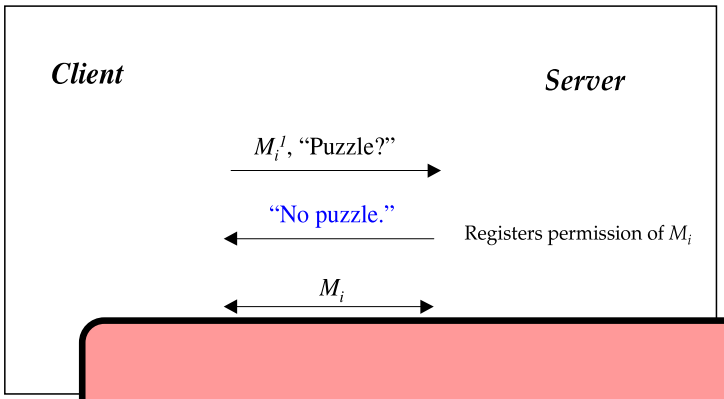
- Example: TCP SYN flood[10]:
  - Given the challenge **C**, find **X** such that:
    $LSB_n(\ SHA1(\ C\ ||\ X\ )\ ) = 0^n$
  - requires 2^n to solve $\Rightarrow$ for $n = 16$ it requires 0.3 second on a 1 Ghz machine

[10] http://www.arijuels.com/wp-content/uploads/2013/09/JB99.pdf

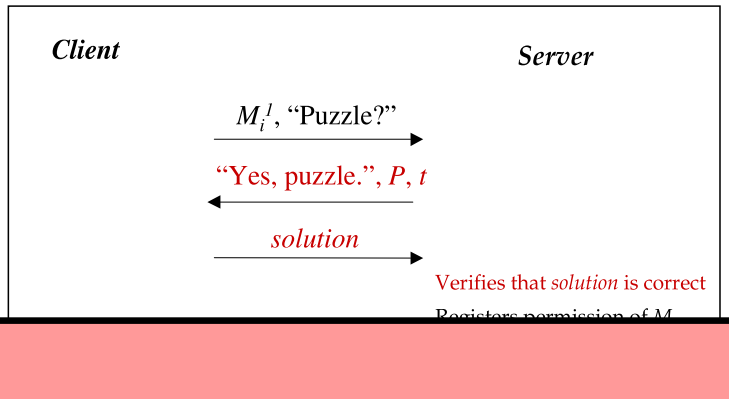
Client Puzzle

- Idea: request client(s) to *do some work*, to slow down attacker(s).
  1. Moderately expensive to compute (client side).
  2. Cheap to verify (server side).
- Block in case the provided solution is incorrect.

Webserver operating normally:



Webserver under attack:



- chine

Limitation: **not deployed in practice**. Why?

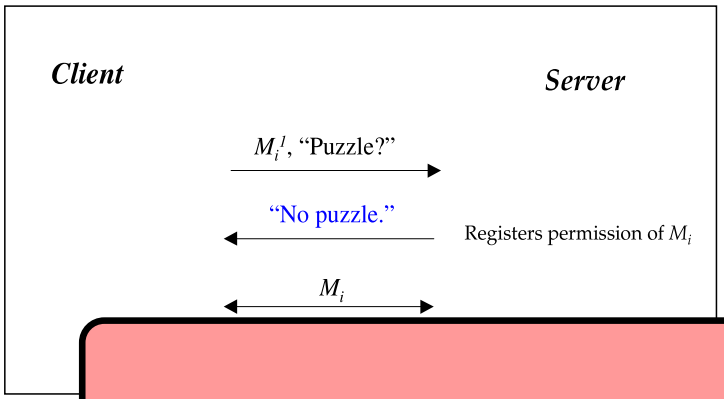
1.
2.
3.

[10] http://www.arijuels.com/wp-content/uploads/2013/09/JB99.pdf

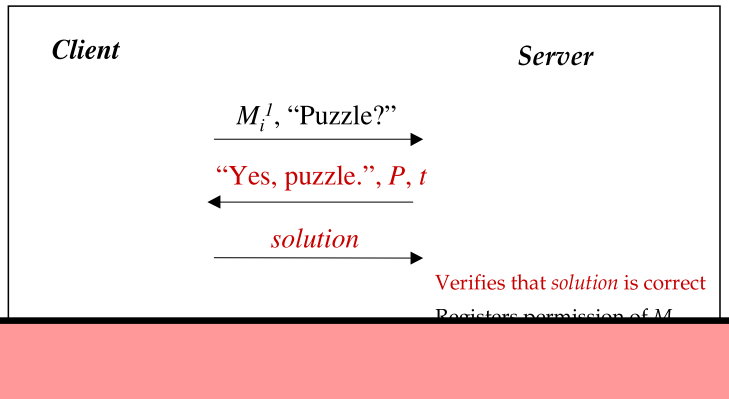
Client Puzzle

- Idea: request client(s) to *do some work*, to slow down attacker(s).
  1. Moderately expensive to compute (client side).
  2. Cheap to verify (server side).
- Block in case the provided solution is incorrect.

Webserver operating normally:



Webserver under attack:



-

Limitation: **not deployed in practice**. Why?

1. Protects against *SYN flooding*, but not against *link flooding*.
2. It requires changes on both *client* and *servers*.
3. Is hard for low-powered clients/devices (e.g., IoT).

chine

[10] http://www.arijuels.com/wp-content/uploads/2013/09/JB99.pdf

Traceback

- Goal: determine packet source, despite spoofing.
- How: configure routers to include path information in each packet.

Traceback

- Goal: determine packet source, despite spoofing.
- How: configure routers to include path information in each packet.

Simple approach:

1. Each router appends its own IP address to the packet.
2. The destination (victim) can reconstruct the path to the source (attacker).

Traceback

- Goal: determine packet source, despite spoofing.
- How: configure routers to include path information in each packet.

Simple approach:

1. Each router appends its own IP address to the packet.
2. The destination (victim) can reconstruct the path to the source (attacker).

Issue: *Where do we log this information?*.

- Paths can be very long.
- No usable fields in the IP datagrams.

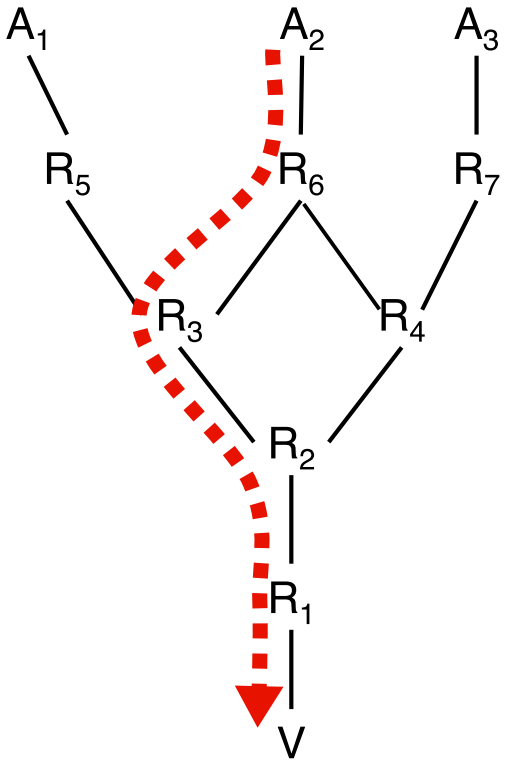
Proposed solutions:

- *"Practical Network Support for IP Traceback", Savage, Wetherall, Karlin, Anderson [SIGCOMM'00]*
- "Hash-Based IP Traceback", Snoeren, Partridge, Sanchez, Jones, Tchakountio, Kent, Strayer [SIGCOMM'01]
- "An algebraic approach to IP traceback", Stubblefield, Dean, Franklin [NDSS'01]

Traceback Intuition [SIGCOMM'00]

Assumptions:

1. Most routers (on the path) are not compromised.
2. The attacker sends **many packets**.
3. The **route** attacker $\rightarrow$ victim **remains stable**.



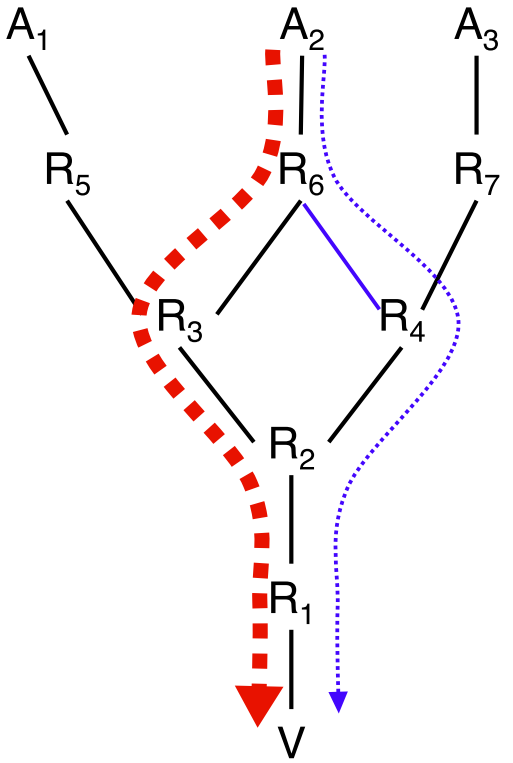
A_i = attacker
R_j = router
V = victim

Source: `https://www.cs.unc.edu/~jeffay/courses/nidsS05/traceback/savage-traceback-sigcomm00.pdf`

Traceback Intuition [SIGCOMM'00]

Assumptions:

1. Most routers (on the path) are not compromised.
2. The attacker sends **many packets**.
3. The **route** attacker $\rightarrow$ victim **remains relatively stable**.



A_i = attacker
R_j = router
V = victim

Source: `https://www.cs.unc.edu/~jeffay/courses/nidsS05/traceback/savage-traceback-sigcomm00.pdf`

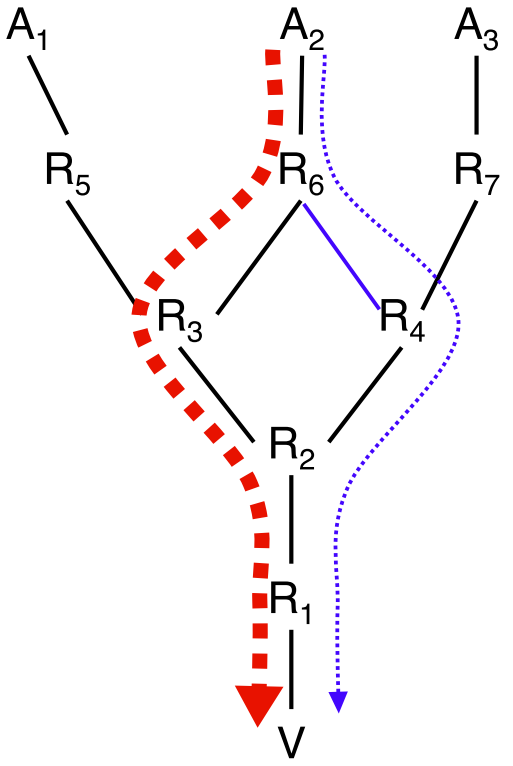
Traceback Intuition [SIGCOMM'00]

Assumptions:

1. Most routers (on the path) are not compromised.
2. The attacker sends **many packets**.
3. The **route** attacker → victim **remains relatively stable**.

Approach: store *only one link* in each packet.

- Each router probabilistically stores only its own IP address.
  ⇒ Fixed space needed, regardless of the path.
- The full path (attacker → victim) can be reconstructed with enough packets.



A_i = attacker
R_j = router
V = victim

Source: https://www.cs.unc.edu/~jeffay/courses/nidsS05/traceback/savage-traceback-sigcomm00.pdf

Traceback - Limitations

Traceback is not deployed in practice:

1. It requires changes to the routers.

Traceback - Limitations

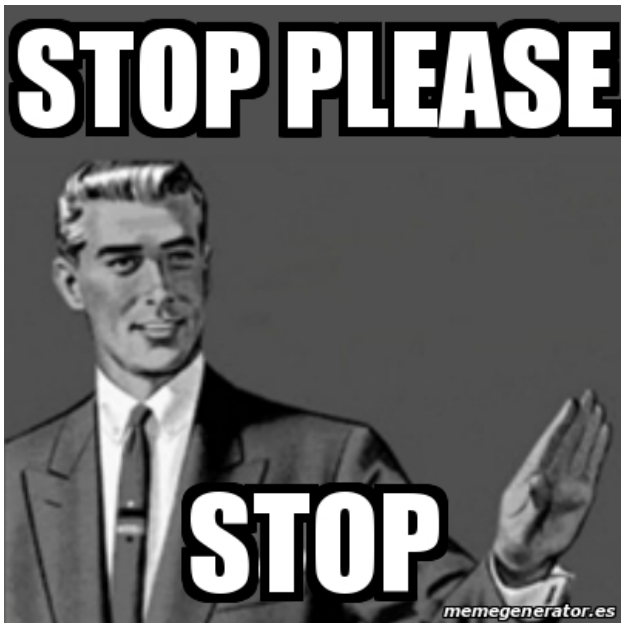
Traceback is not deployed in practice:

1. It requires changes to the routers.
2. Does knowing the source will stop the attack?

Traceback - Limitations

Traceback is not deployed in practice:

1. It requires changes to the routers.
2. Does knowing the source will stop the attack?
   Attacking packets can originate from:
   - Bullet-proof hosting providers.
   - Reflectors.
   - Bots.
   - States that deny any cooperation.



Pushback Filtering

Idea: dynamic traffic filtering on the backbone[11].

How: by detecting and controlling a high-bandwidth subsets of traffic.

[11] https://www.microsoft.com/en-us/research/publication/2017/01/Controlling-High-Bandwidth-Aggregates-in-the-Network.pdf

Pushback Filtering

Idea: dynamic traffic filtering on the backbone[11].

How: by detecting and controlling a high-bandwidth subsets of traffic.

- Move filtering *toward the attack sources*!
- The router implements the following procedure:
  1. *Am I seriously congested?*
     - *Check if packet drop rate above the threshold.*

[11] https://www.microsoft.com/en-us/research/publication/2017/01/Controlling-High-Bandwidth-Aggregates-in-the-Network.pdf

Pushback Filtering

Idea: dynamic traffic filtering on the backbone[11].

How: by detecting and controlling a high-bandwidth subsets of traffic.

- Move filtering *toward the attack sources*!
- The router implements the following procedure:
  1. *Am I seriously congested?*
     - *Check if packet drop rate above the threshold.*
  2. **If so**, can I identify and traffic subset responsible for an appreciable portion of the congestion?
     - *Cluster a sample of the high-volume traffic.*

[11] https://www.microsoft.com/en-us/research/publication/2017/01/Controlling-High-Bandwidth-Aggregates-in-the-Network.pdf

Pushback Filtering

Idea: dynamic traffic filtering on the backbone[11].

How: by detecting and controlling a high-bandwidth subsets of traffic.

- Move filtering *toward the attack sources*!
- The router implements the following procedure:
  1. *Am I seriously congested?*
     - *Check if packet drop rate above the threshold.*
  2. **If so**, can I identify and traffic subset responsible for an appreciable portion of the congestion?
     - *Cluster a sample of the high-volume traffic.*
  3. **If so**, to what degree do I limit the traffic subset?
     Do I also ask *upstream* routers to limit?

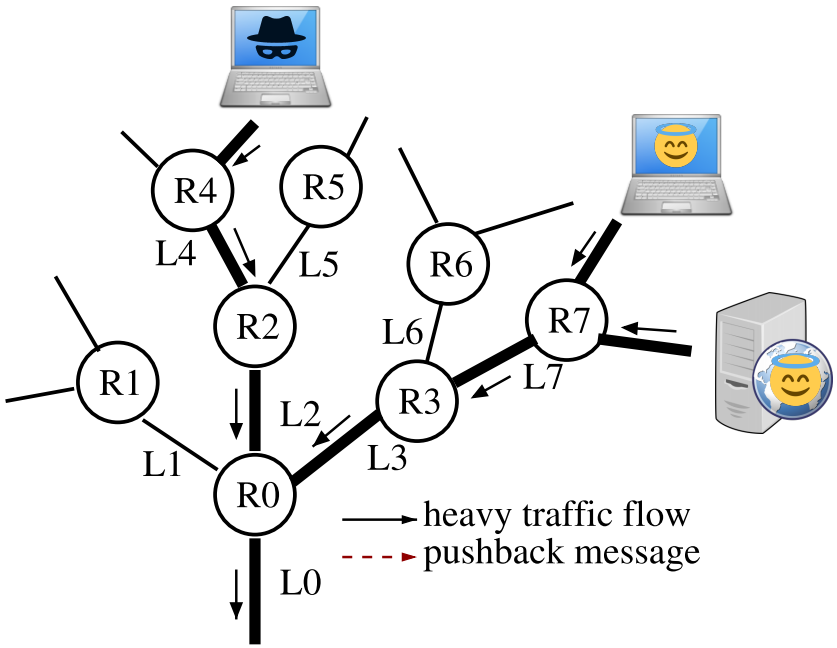
[11] https://www.microsoft.com/en-us/research/publication/2017/01/Controlling-High-Bandwidth-Aggregates-in-the-Network.pdf

Pushback Filtering

Idea: dynamic traffic filtering on the backbone[11].

How: by detecting and controlling a high-bandwidth subsets of traffic.

- Move filtering *toward the attack sources*!
- The router implements the following procedure:
  1. *Am I seriously congested?*
     - *Check if packet drop rate above the threshold.*
  2. **If so**, can I identify and traffic subset responsible for an appreciable portion of the congestion?
     - *Cluster a sample of the high-volume traffic.*
  3. **If so**, to what degree do I limit the traffic subset?
     Do I also ask *upstream* routers to limit?



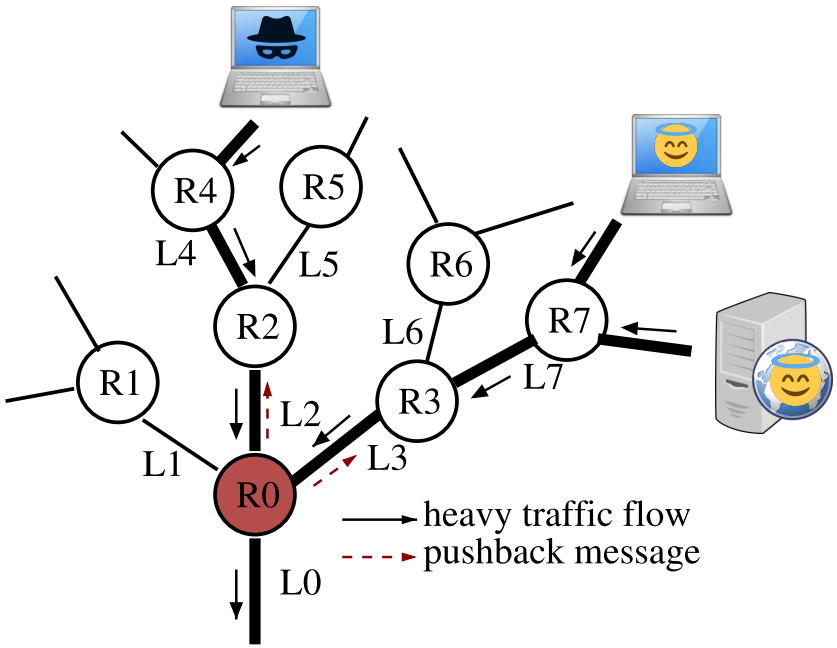
[11] https://www.microsoft.com/en-us/research/publication/2017/01/Controlling-High-Bandwidth-Aggregates-in-the-Network.pdf

Pushback Filtering

Idea: dynamic traffic filtering on the backbone[11].

How: by detecting and controlling a high-bandwidth subsets of traffic.

- Move filtering *toward the attack sources*!
- The router implements the following procedure:
  1. *Am I seriously congested?*
     - *Check if packet drop rate above the threshold.*
  2. **If so**, can I identify and traffic subset responsible for an appreciable portion of the congestion?
     - *Cluster a sample of the high-volume traffic.*
  3. **If so**, to what degree do I limit the traffic subset?
     Do I also ask *upstream* routers to limit?



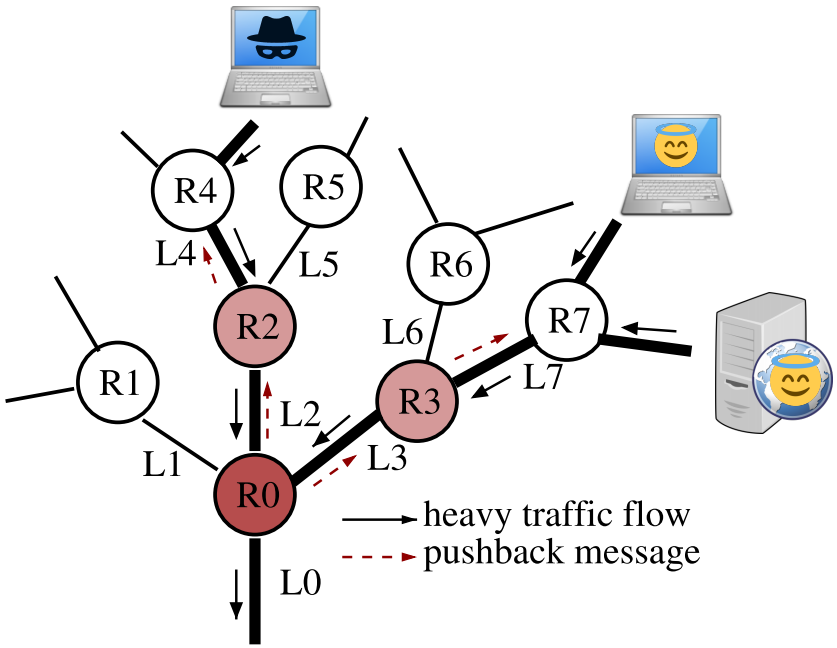
[11] https://www.microsoft.com/en-us/research/publication/2017/01/Controlling-High-Bandwidth-Aggregates-in-the-Network.pdf

Pushback Filtering

Idea: dynamic traffic filtering on the backbone[11].

How: by detecting and controlling a high-bandwidth subsets of traffic.

- Move filtering *toward the attack sources*!
- The router implements the following procedure:
  1. *Am I seriously congested?*
     - *Check if packet drop rate above the threshold.*
  2. **If so**, can I identify and traffic subset responsible for an appreciable portion of the congestion?
     - *Cluster a sample of the high-volume traffic.*
  3. **If so**, to what degree do I limit the traffic subset?
     Do I also ask *upstream* routers to limit?



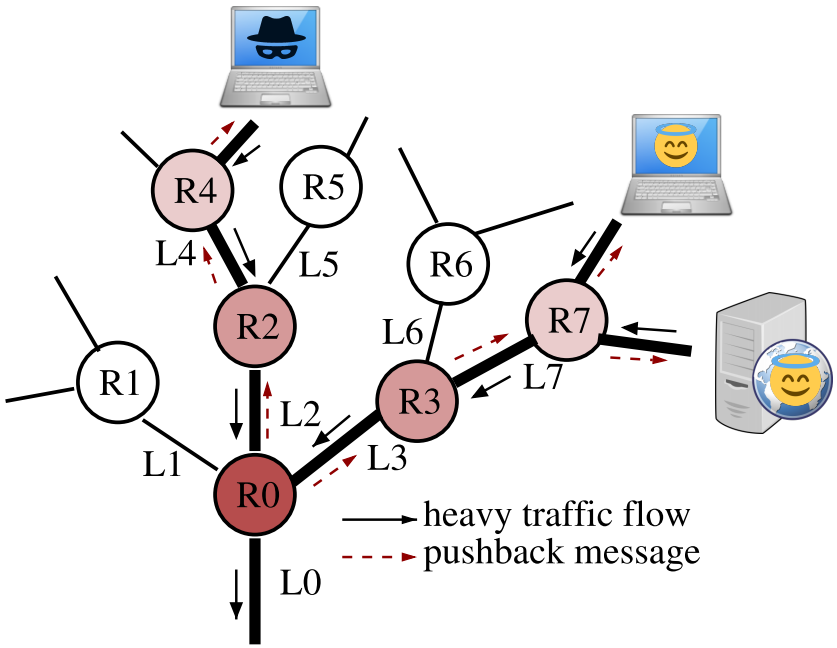
[11] https://www.microsoft.com/en-us/research/publication/2017/01/Controlling-High-Bandwidth-Aggregates-in-the-Network.pdf

Pushback Filtering

Idea: dynamic traffic filtering on the backbone[11].

How: by detecting and controlling a high-bandwidth subsets of traffic.

- Move filtering *toward the attack sources*!
- The router implements the following procedure:
  1. *Am I seriously congested?*
     - *Check if packet drop rate above the threshold.*
  2. **If so**, can I identify and traffic subset responsible for an appreciable portion of the congestion?
     - *Cluster a sample of the high-volume traffic.*
  3. **If so**, to what degree do I limit the traffic subset?
     Do I also ask *upstream* routers to limit?



[11] https://www.microsoft.com/en-us/research/publication/2017/01/Controlling-High-Bandwidth-Aggregates-in-the-Network.pdf

Capabilities

Idea: before sending a packet, a node must first obtain the "permission to send" from the destination[12].

How: the receiver provides tokens, or capabilities, to those senders whose traffic it agrees to accept.

[12] https://people.inf.ethz.ch/troscoe/pubs/netcap_hotnets.pdf
[13] https://users.ece.cmu.edu/~adrian/projects/siff.pdf

Capabilities

Idea: before sending a packet, a node must first obtain the "permission to send" from the destination[12].

How: the receiver provides tokens, or capabilities, to those senders whose traffic it agrees to accept.

- Process:
  1. Sender requests capability.
     - Path identifier to limit the *number of requests from one source*.
  2. If desired, the receiver responds with a capability.
  3. The sender includes the capability in *all* future packets.

[12] https://people.inf.ethz.ch/troscoe/pubs/netcap_hotnets.pdf
[13] https://users.ece.cmu.edu/~adrian/projects/siff.pdf

Capabilities

Idea: before sending a packet, a node must first obtain the "permission to send" from the destination[12].

How: the receiver provides tokens, or capabilities, to those senders whose traffic it agrees to accept.

- Process:
  1. Sender requests capability.
     - Path identifier to limit the *number of requests from one source*.
  2. If desired, the receiver responds with a capability.
  3. The sender includes the capability in *all* future packets.

- Routers forward only:
  A. request packets;
  B. packets with a *valid capability*.
     - Capabilities need to be updated across time[13]
       $\Rightarrow$ receivers notify senders.

[12] https://people.inf.ethz.ch/troscoe/pubs/netcap_hotnets.pdf
[13] https://users.ece.cmu.edu/~adrian/projects/siff.pdf

DoS - Conclusions

- (D)DoS has been a problem since 2000's
  ... and is still an important problem.

- Different solutions have been deployed:
  - ingress filtering;
  - SYN cookies;
  - DDoS protection services.

- Some problems still remain:
  - vulnerable and unpached **reflectors**;
  - spoofing is possible due to incomplete adoption of ingress filtering.

- Many researchers have proposed more robust solutions, that require a re-design of Internet services
  $\Rightarrow$ Traceback, pushback filtering, capabilities.

Mandatory readings

- RFC4987 - "TCP SYN Flooding Attacks and Common Mitigations"
  `https://datatracker.ietf.org/doc/html/rfc4987`

- Paper - "An analysis of using reflectors for distributed denial-of-service attacks"
  `https://ccr.sigcomm.org/archive/2001/jul01/ccr-200107-paxson.pdf`
  alternative download link
  `https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=16089cc1668f937fa4b0b24c2cc29a9a91d8ec39`

- Paper - "Amplification Hell: Revisiting Network Protocols for DDoS Abuse"
  `https://christian-rossow.de/publications/amplification-ndss2014.pdf`