DDoS Attack and Defense

CSIE 7190 Cryptography and Network Security, Spring 2019

https://ceiba.ntu.edu.tw/1072csie_cns

cns@csie.ntu.edu.tw

Hsu-Chun Hsiao

National Taiwan University

Housekeeping

5/07: Project proposal due

5/07: Reading critique #9 due

HW2 released, due in 3 weeks

Reading critique #9

Write a critique on one of the following:

- Adrian, David, et al. “Imperfect forward secrecy: How Diffie-Hellman fails in practice,” in ACM CCS, 2015.
- S. Frolov and E. Wustrow, "The use of TLS in Censorship Circumvention," in NDSS, 2019.

Text only, one page

Agenda

What is DDoS?

Common types of DDoS

Mitigating DDoS

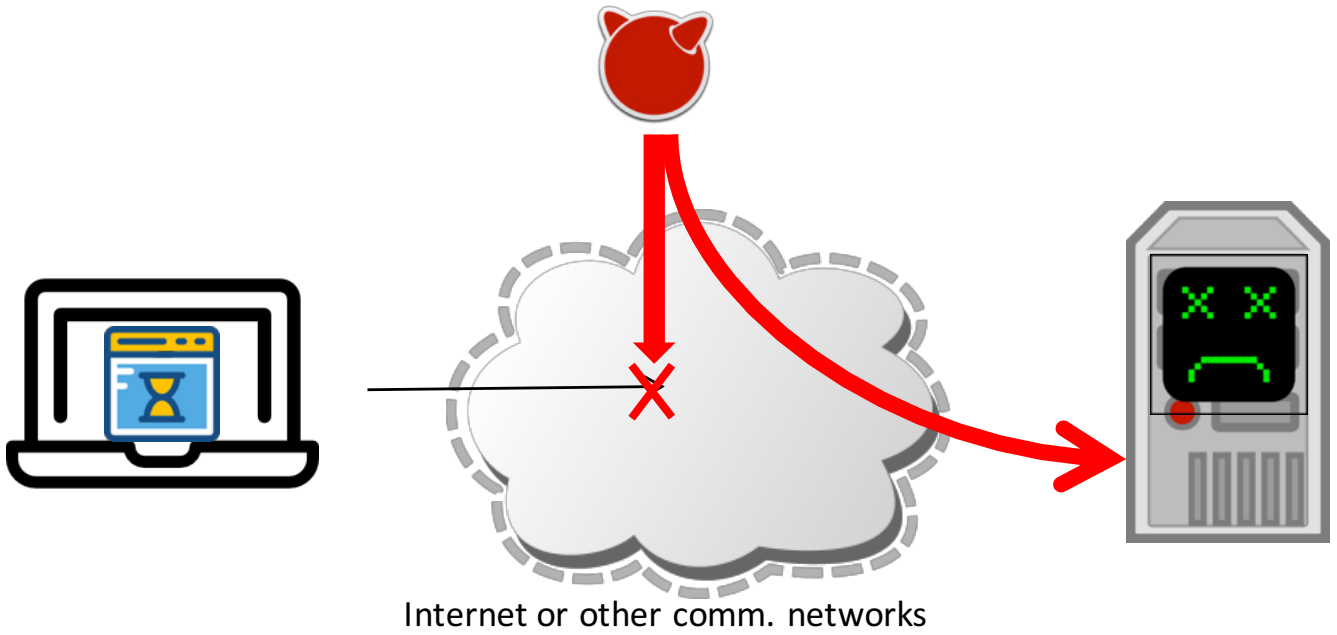
Denial of Service (阻斷服務攻擊)

讓使用者無法使用想要的服務

針對可用性(availability)的攻擊

阻斷服務攻擊常見手法

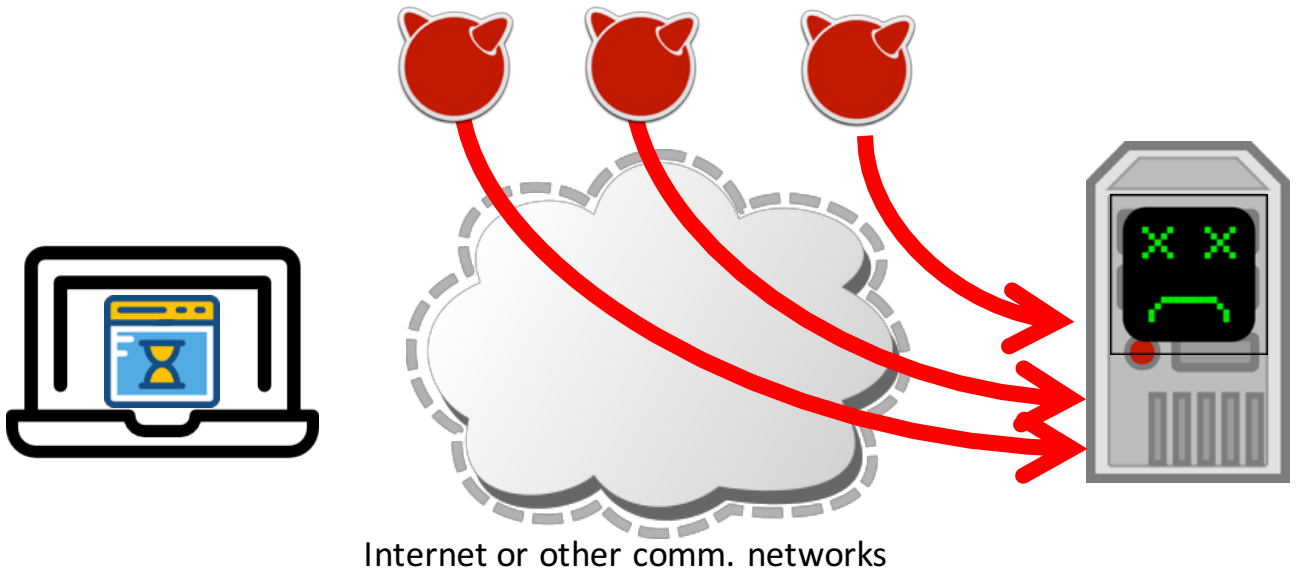
大量消耗共用的資源（e.g. bandwidth, CPU, memory）



分散式阻斷服務攻擊 (Distributed Denial of Service)

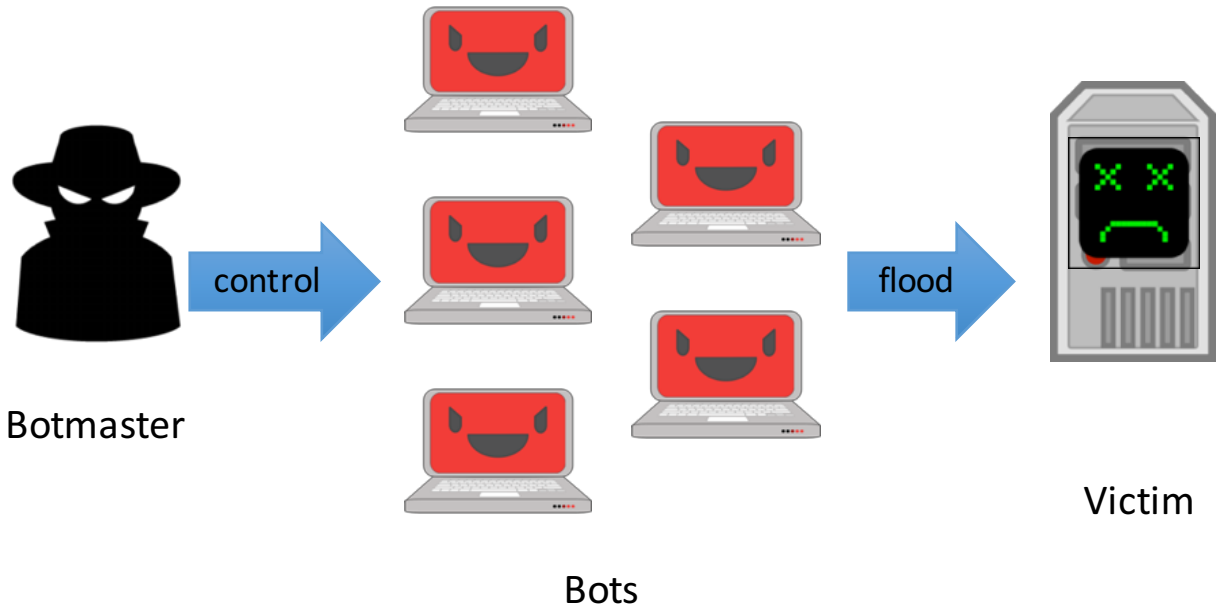
多於一個攻擊來源

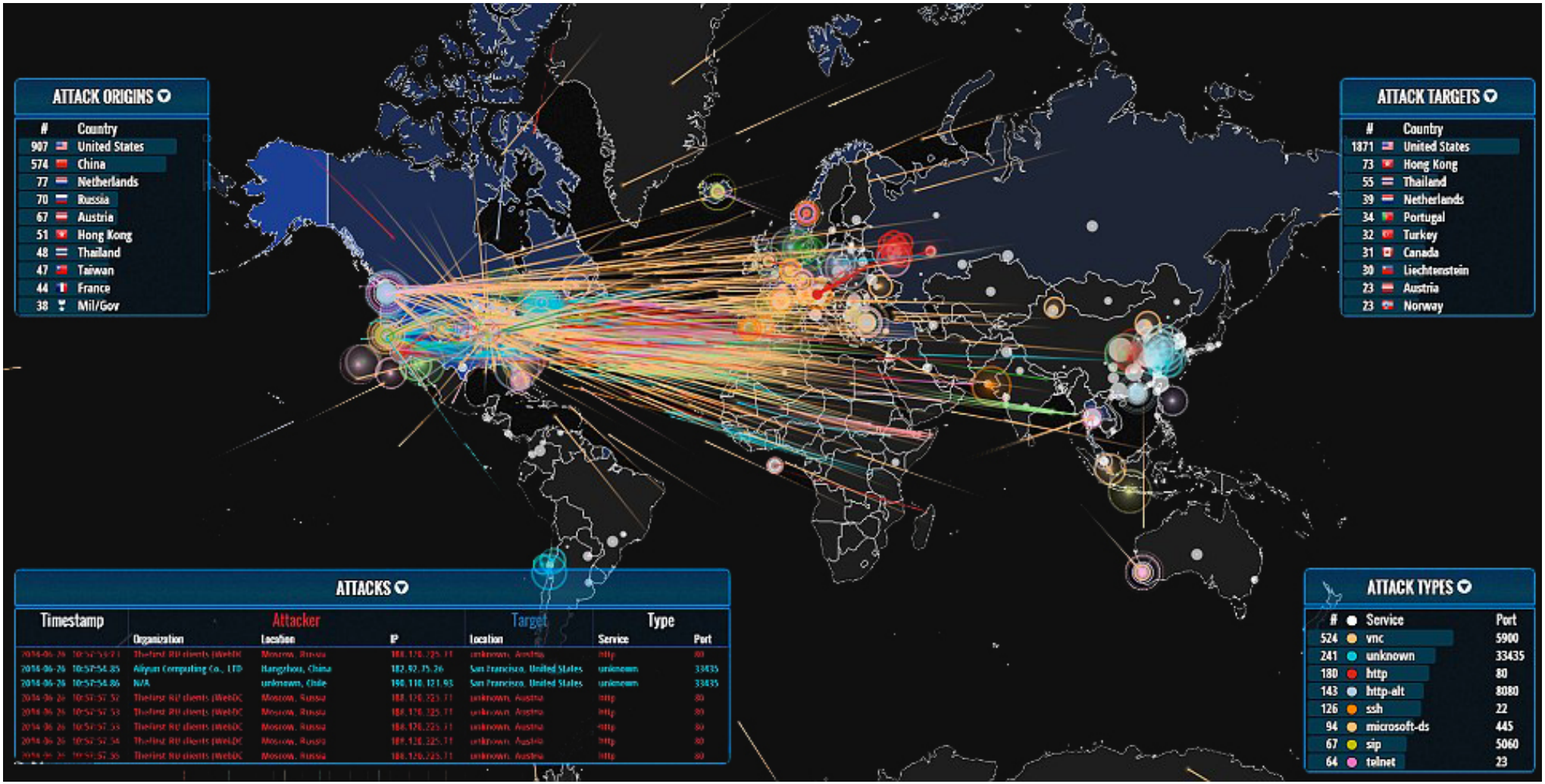
提高攻擊強度、降低單一來源被偵測的風險



Botnet-driven DDoS

Botnet = 殭屍網路





http://map.norsecorp.com/ (link inaccessible)

Why DoS/DDoS?

臺灣史上第一次券商集體遭DDoS攻擊勒索事件

2017年才開春，臺灣就爆發了有史以來第一次券商集體遭DDoS攻擊勒索事件，全臺79家券商中，有13家券商遭到DDoS攻擊勒索，遭攻擊券商單日平均交易金額加總起來近206億元，約臺股單日交易金額的3成，都受到了威脅！

文/ iThome | 2017-02-14 發表

讚 4 萬 按讚加入iThome粉絲團 讚 分享 217 G+1 4

行政院資安處：投票當天中選會遭DDoS攻擊流量不到10Gbps，維持不到10分鐘

中選會投開票的計票系統是獨立的內網形式，並沒有受到DDoS攻擊的波及

文/ 黃彥棻 | 2018-11-30 發表

讚 5.2 萬 按讚加入iThome粉絲團 讚 326 分享

Why DoS/DDoS?

恐嚇勒索

商業競爭

轉移焦點

政治示威

前導攻擊

這種攻擊相對簡單、有效、又難以防禦

- Botnet for hire (botnet as a service)
- Many tools available
- Flash crowds vs. attacks

DDoS-for-hire

ddos booter

Bootyou - DDoS Service

調查局破獲臺灣首家違法提供DDoS攻擊公務機關等單位官方網站

> 調查局破獲臺灣首家違法提供DDoS攻擊公務機關等單位官方網站

發布日期 107-05-30 18:01:08 更新日期 107-05-31 15:19:23 公共事務室

TWDDOS 網站(https://tddos.pw)成員號稱以分散式阻斷服務(DDoS)進行攻擊並製作多個公務機關遭受攻擊影片，放上Youtube供人瀏覽，以招攬會員購買DDoS攻擊服務，並針對公務機關、金融機構等單位進行攻擊，以提升知名度，因受害機關眾多，調查局臺北市調查處與刑事警察局偵九隊分別成立專案小組，經調查相關事證後，於5月28日報請臺北地方法院檢察署蕭奕弘檢察官指揮，依法搜索TWDDOS網站管理者處所，查扣犯案之手機、電腦及比特幣等證物，本案亦為首次查獲從事比特幣非法交易之案件。

FBI大執法，一舉掃蕩15個DDoS租賃服務網站

FBI一舉掃蕩15個專門提供DDoS服務的網站，這些網站在暗網中打廣告，聲稱掌控殭屍網路，可發動DDoS攻擊，受害者遍佈金融機構、大學、網路服務商、政府機構及遊戲平台。

文/ 陳曉莉 | 2018–12–24 發表

讚 5.4 萬 按讚加入iThome粉絲團 讚 295 分享

Free. Open for security community, Free to use and cite.

WebStresser - IP Stresser / Booter | DDoS Tool

https://webstresser.biz/

webstresser.biz is the strongest IP Stresser / **Booter** on the market, we provide strongest and most reliable server stress testing, with up to 500Gbps!

DDoS攻擊大事記

Mar. 2013: **DNS amplification** against Spamhaus at **300Gbps**

Mar. 2015: “Great Cannon” **browser-based** DDoS against GitHub

Sep. 2016: Mirai **IoT botnets** caused DDoS at **620Gbps**

Oct. 2016: Mirai IoT botnets attacked **critical Internet infrastructure** (Dyn DNS service provider), taking down GitHub, Twitter, Reddit, Netflix, Airbnb, etc.

Mar. 2018: **memcached amplification** DDoS against Github at **1.3Tbps**

DDoS attacks grow in volume, frequency and sophistication!

Application-layer DDoS attacks are getting popular

How to DoS?

Resource exhaustion (Bandwidth, CPU, memory)

- Flooding hosts
- Flooding infrastructure
- Low-rate attacks exploiting protocol/algorithm specification

System crashed via implementation vulnerabilities

- E.g., Ping of Death

Examples of DoS Attacks

SYN flood

- Sends SYN packets to overflow TCP connection table

Algorithmic complexity attack

- Exploits worst case, e.g., O(n) -> $O(n^2)$ running time

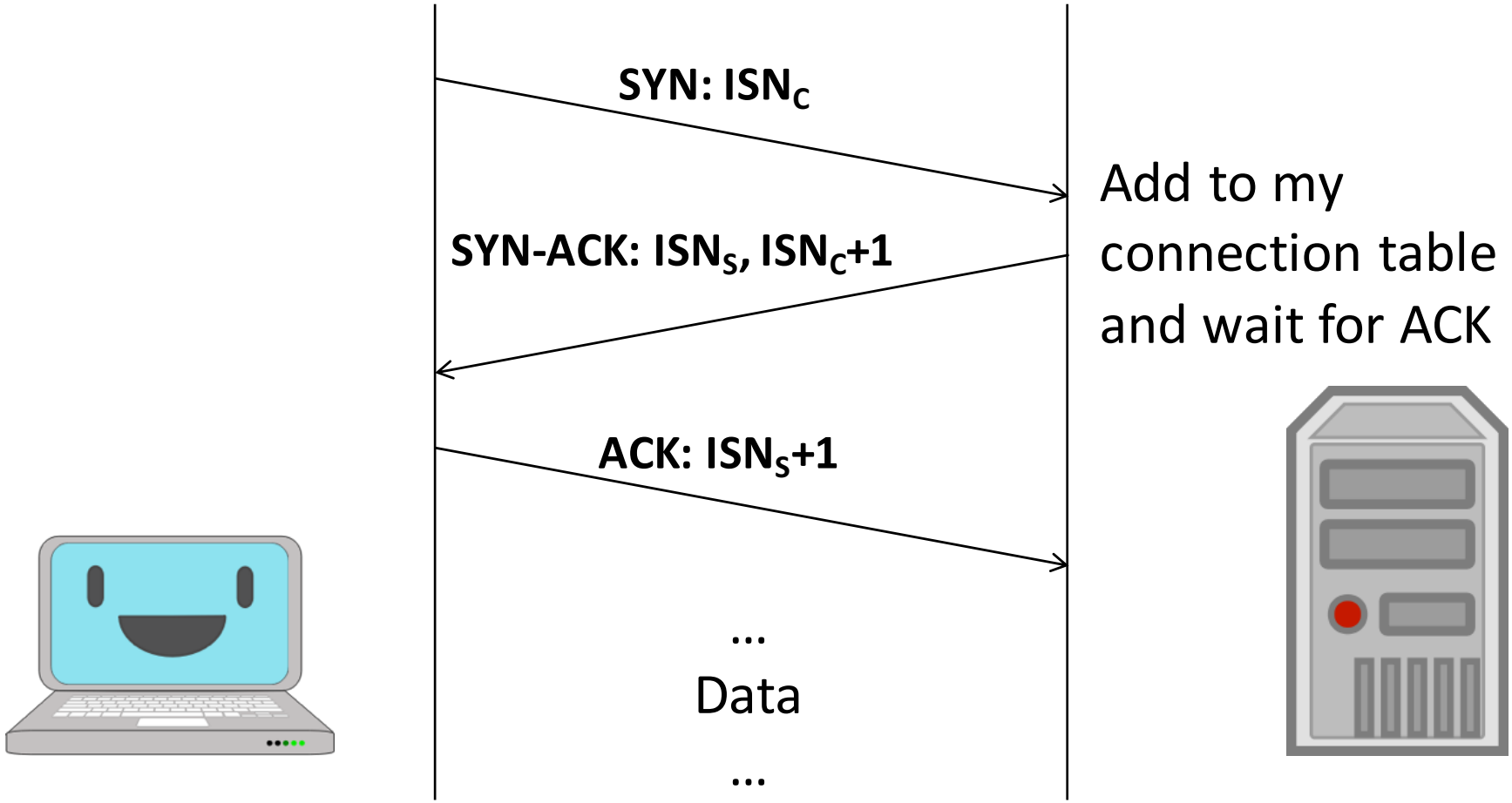
Low-rate pulsing attack

- Exploits TCP congestion control

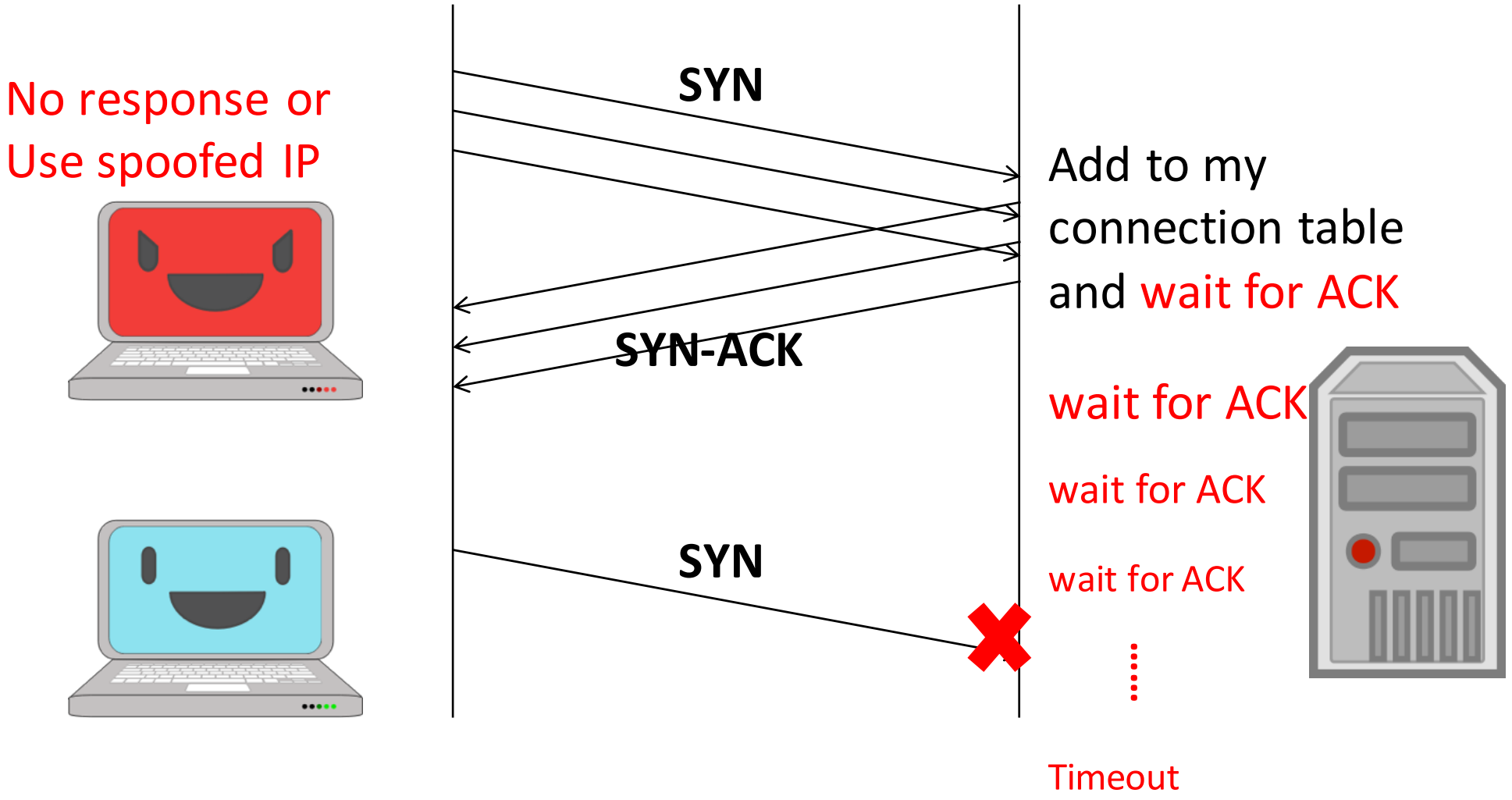
SYN Flood

TCP Connection Establishment: Three-way Handshake

ISNs (initial sequence number) are picked at random



TCP SYN Flood



TCP SYN Flood

First serious DoS attack

Single attacker could tie up server resources to prevent other clients from connecting to server

Problem exploited by SYN Flooding?

- Server needs to keep state after receiving initial SYN
- Connection table has limited size
- Attacker floods server with SYN packets, does not follow up with ACK packet to complete TCP handshake
- Server keeps state, waits for ACK, exhausts resources

Countermeasures

Buy more memory?

Shorter timeout?

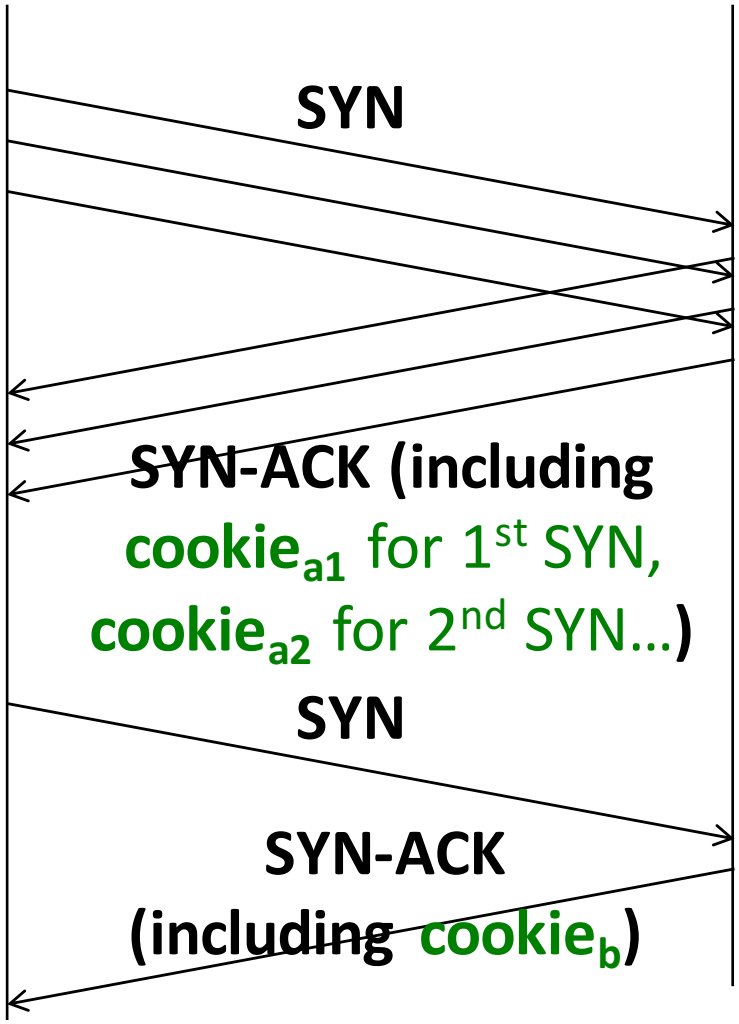
IP-based filtering?

Better solution: TCP SYN Cookie

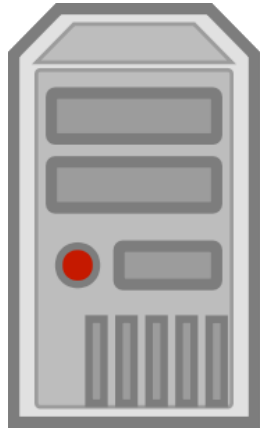
- By D. J. Bernstein, http://cr.yp.to/syncookies.html
- Server keeps no state before handshake completion
- Disadvantages: computational overhead, TCP options lost

TCP SYN Cookies

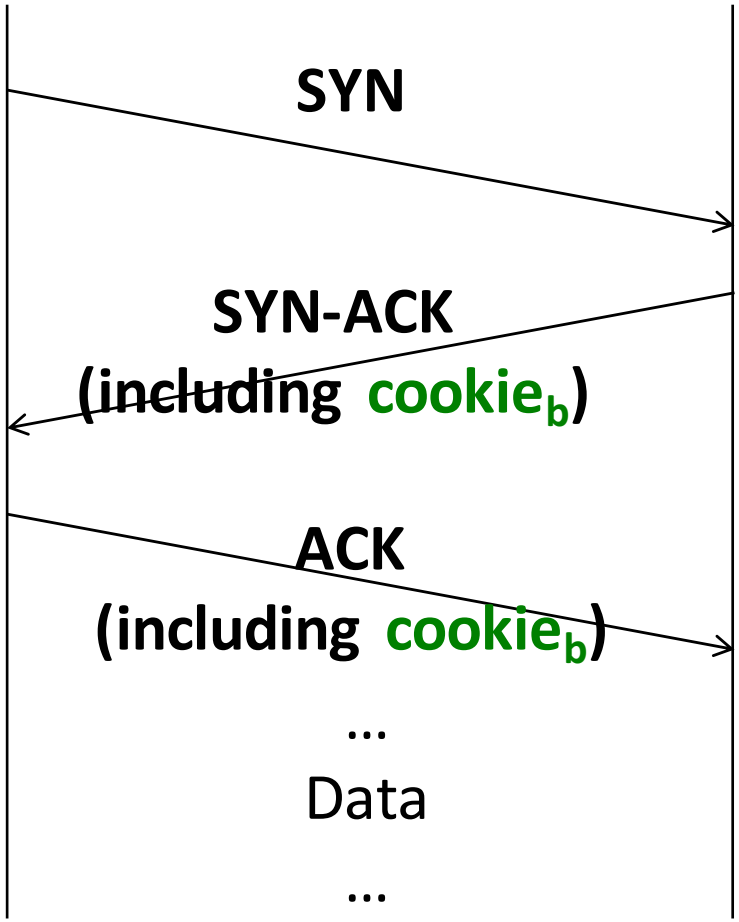
No response or
Use spoofed IP



Compute SYN cookie per SYN request based on connection state, no state kept



TCP SYN Cookies



Compute SYN **cookie$_b$** based on connection state, keep no state

Verify cookie;
If correct, allocate state

TCP SYN Cookies

TCP SYN Cookie is encoded as a particular chosen server ISN (initial sequence number)

- ISN was picked randomly before

How to generate TCP SYN Cookie?

What about?

```
Cookie = H(Server_IP, Client_IP, Server_port,
Client_port, Client_state, time)
```

Better:

```
Cookie = MAC_key(Server_IP, Client_IP, Server_port,
Client_port, Client_state, time)
```

Key is kept to server only

問題：為什麼不乾脆正常地建立連線，把連線表佔滿？

Attack by establishing thousands of connections?

Works too, but less efficient and easier to detect

Less efficient: attacker needs to keep state too

Easier to detect: Server can easily identify the attacker, since it uses its real IP with thousands of open connections

Algorithmic Complexity Attack

Algorithmic Complexity Attack

Induce worst-case behavior in a vulnerable algorithm

The larger the difference between the worst case and average case, the more vulnerable the algorithm

Algorithm	**Average**	**Worst**
Quicksort	O(n log n)	$O(n^2)$
Hash table lookup	constant	O(n)

Crosby, Scott A., and Dan S. Wallach. "Denial of Service via Algorithmic Complexity Attacks." Usenix Security. Vol. 2. 2003.
Smith, Randy, Cristian Estan, and Somesh Jha. "Backtracking algorithmic complexity attacks against a NIDS." ACSAC, 2006

Algorithm Name	Best Case	Worst Case
Optimized Insertion Sort	$\Theta(n)$	$\Theta(n^2)$
Quick Sort	$\Theta(n \log n)$	$\Theta(n^2)$
Optimized Quick Sort	$\Theta(n \log n)$	$\Theta(n^2)$
3-way Quick Sort	$\Theta(n \log n)$	$\Theta(n^2)$
Sequential Search	$\Theta(1)$	$\Theta(n)$
Binary Search	$\Theta(1)$	$\Theta(\log n)$
Binary Search Tree Lookup	$\Theta(1)$	$\Theta(n)$
Red-Black Tree Lookup	$\Theta(1)$	$\Theta(\log n)$
Separate Chain Hash Lookup	$\Theta(1)$	$\Theta(n)$
Linear Probing Hash Lookup	$\Theta(1)$	$\Theta(n)$
NFA Regex Match	$\Theta(m + n)$	$\Theta(mn)$
Booyer-Moore Substring	$\Theta(m + n)$	$\Theta(mn)$
Prim Minimum Spanning Tree	$\Theta(V + E)$	$\Theta(E \log V)$
Bellman-Ford Shortest Path	$\Theta(1)$	$\Theta(V(V + E))$
Dijkstra Shortest Path	$\Theta(1)$	$\Theta(E \log V)$
Alternating Path Bipartite	$\Theta(V)$	$\Theta(V(V + E))$
Hopcroft-Karp Bipartite	$\Theta(V)$	$\Theta(E\sqrt{V})$

Jiayi Wei, Jia Chen, Yu Feng, Kostas Ferles, and Isil Dillig. 2018. Singularity: Pattern Fuzzing for Worst Case Complexity. In Proceedings ofthe 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 18), November 4–9, 2018.

Hash table lookup: Collisions

Collisions are common due to small table size; table resized when load factor > threshold

Collision resolution: chaining, open addressing

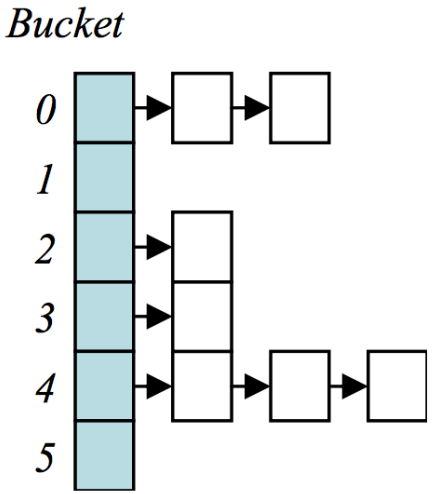


Figure 1: Normal operation of a hash table.

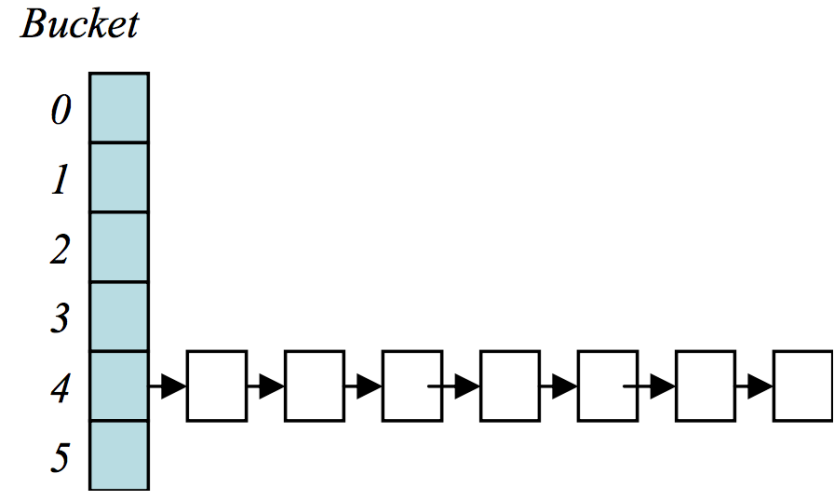


Figure 2: Worst-case hash table collisions.

Hash table lookup: Worst Case is Unlikely…?

Not true with a malicious user comes to play

The attacker can intentionally pick a bad input sequence that causes the worst-case hash table collisions

File version	Perl 5.6.1 program	Perl 5.8.0 program
Perl 5.6.1	6506 seconds	<2 seconds
Perl 5.8.0	<2 seconds	6838 seconds

Table 1: CPU time inserting 90k short attack strings into two versions of Perl.

oCERT Advisories

#2011-003 multiple implementations denial-of-service via hash algorithm collision

Description:

A variety of programming languages suffer from a denial-of-service (DoS) condition against storage functions of key/value pairs in hash data structures, the condition can be leveraged by exploiting predictable collisions in the underlying hashing algorithms.

The issue finds particular exposure in web server applications and/or frameworks. In particular, the lack of sufficient limits for the number of parameters in POST requests in conjunction with the predictable collision properties in the hashing functions of the underlying languages can render web applications vulnerable to the DoS condition. The attacker, using specially crafted HTTP requests, can lead to a 100% of CPU usage which can last up to several hours depending on the targeted application and server performance, the amplification effect is considerable and requires little bandwidth and time on the attacker side.

The condition for predictable collisions in the hashing functions has been reported for the following language implementations: Java, JRuby, PHP, Python, Rubinius, Ruby. In the case of the Ruby language, the 1.9.x branch is not affected by the predictable collision condition since this version includes a randomization of the hashing function.

The vulnerability outlined in this advisory is practically identical to the one reported in 2003 and described in the paper Denial of Service via Algorithmic Complexity Attacks which affected the Perl language.

The reporters own advisory can be found at http://www.nruns.com/_downloads/advisory28122011.pdf

http://www.ocert.org/advisories/ocert-2011-003.html

Hash table lookup: Severe collision may lead to denial of service

Hash table collision in Python

```
n = 20000
d=2**64-1
h={}
for i in xrange(n):
  h[i * d] = i
```

Similar attacks work in PHP, ASP, Python, Ruby, C++, Java...

Hash table lookup : Countermeasures

Universal hashing

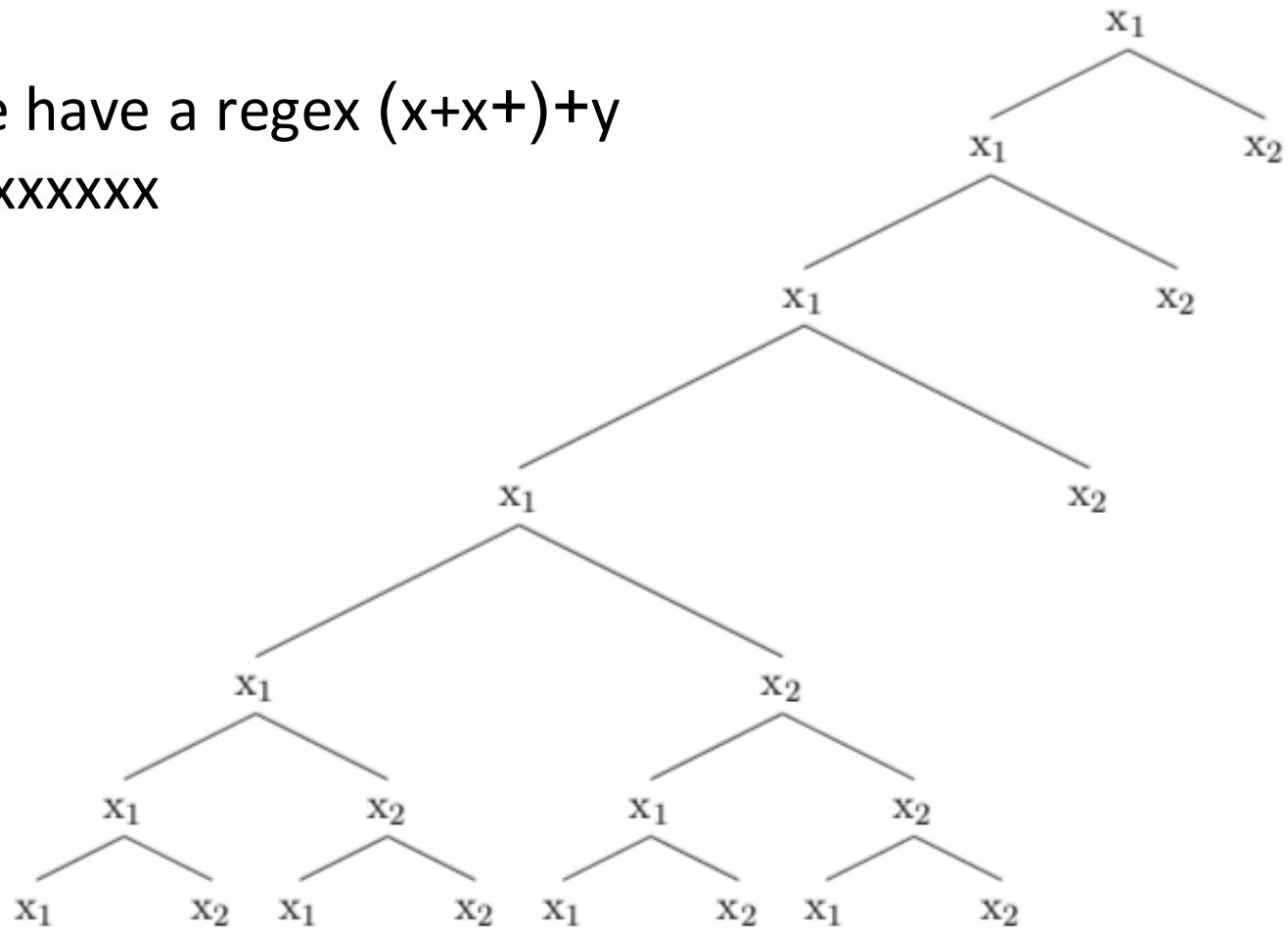
- Hash functions guarantee 0 or low collision for any input

Hash randomization

- Harder for attacker to find out the worst-case input
- E.g., use a secret hash function for each hash table

Regular Expression Denial of Service (ReDoS)

Suppose we have a regex (x+x+)+y and input xxxxxxx



https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS

Regular Expression Denial of Service (ReDoS)

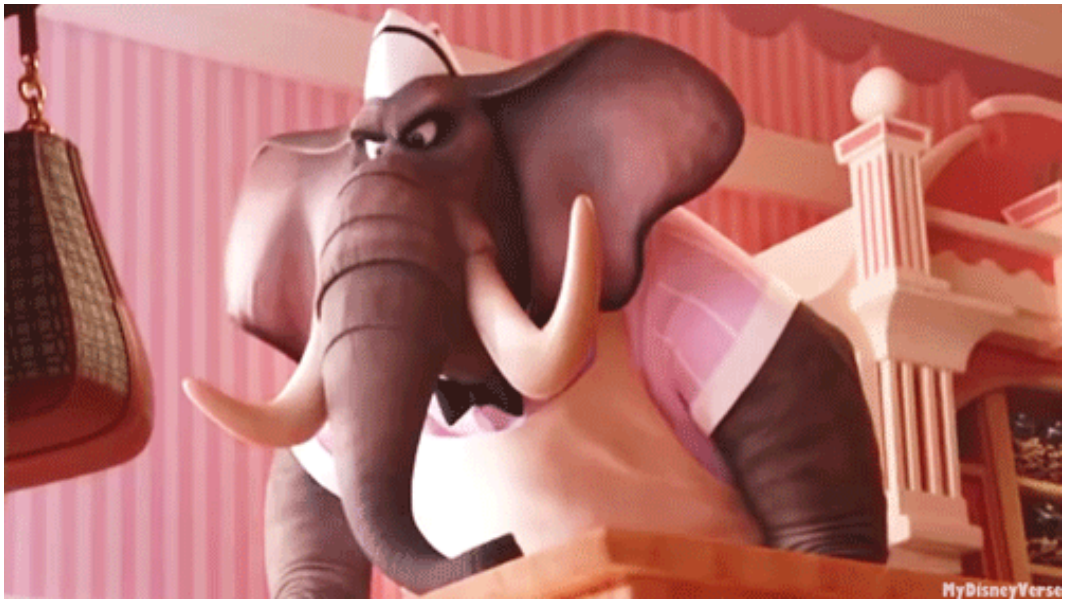
A real example of email validation regex from RegExLib:

```
([a-zA-Z0-9])(([\-.]+)?([a-zA-Z0-9]+))*(@){1}[a-z0-
9]+[.]{1}(([a-z]{2,3})|([a-  z]{2,3}[.]{1}[a-z]{2,3}))
```

Evil input: aaaaaa!@gmail.com

Shrew DoS Attack

Elephant vs. Shrew

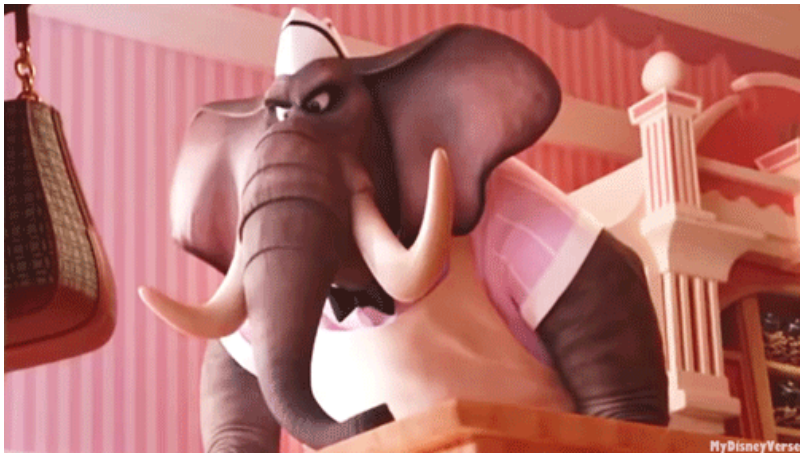


Shrew = 鼩鼱（ㄑㄩˊㄐㄧㄥ）一種形似小鼠的哺乳動物。頭部和背部呈棕褐色，腹部為棕灰色或灰白色。體被短黑毛，具有臭味，以逼退天敵。多生活在山林、田野、沼澤中，捕食昆蟲、蝸牛、蚯蚓等，也吃植物種子和穀物。

Elephant vs. Shrew

Conventional bandwidth-based DoS requires sending **high-rate** attack traffic (like an elephant)

Can we achieve the same effect by sending **low-rate** attack traffic (like a fierce shrew)?



≈

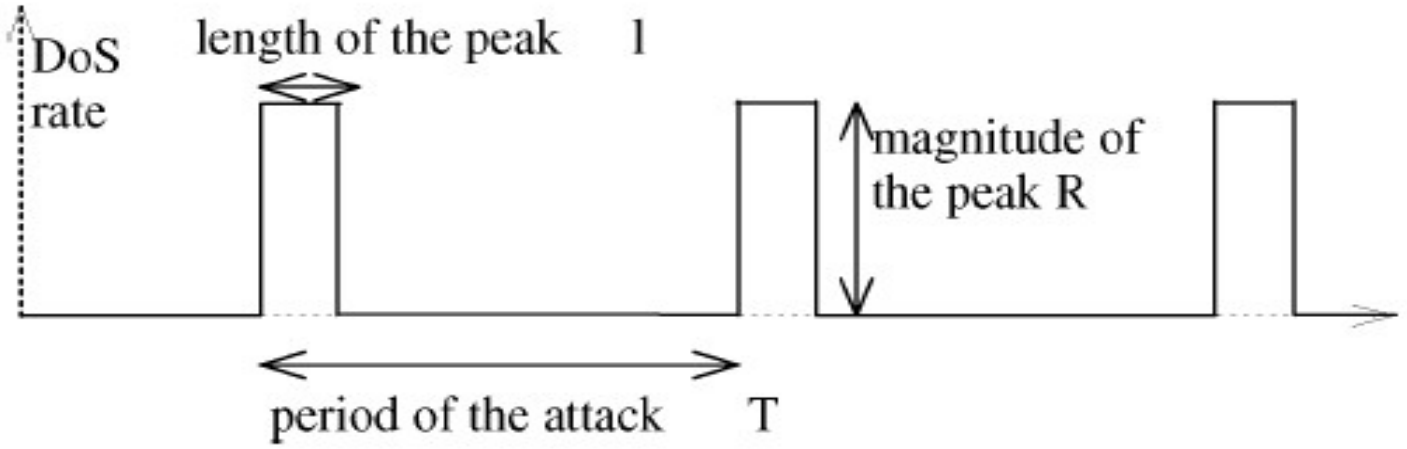
Shrew DoS Attack

Exploits **TCP congestion control** feature

Sends periodic short burst to the target link/router

- => Low traffic volume

Denies the bandwidth of legitimate TCP flows as it makes TCP believe there is a long-term congestion



Retransmission Timeout in TCP Congestion Control

Exponential backoff timeout

- If packet dropped, retransmit in 1 sec
- If resent packet dropped, retransmit in 2 secs
- If resent packet dropped again, retransmit in 4 secs
- …

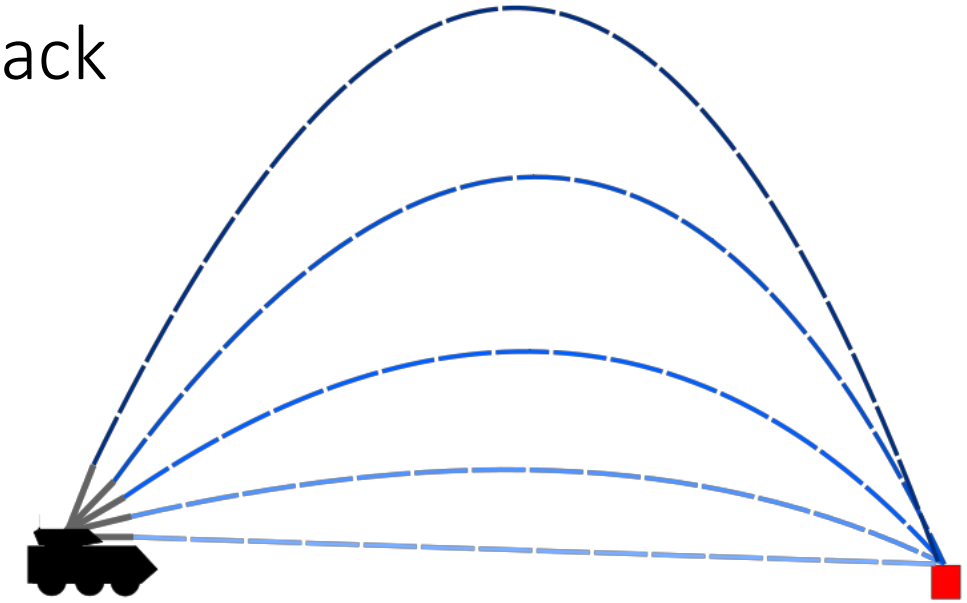
The attacker forces TCP flows to repeatedly enter a retransmission timeout state by sending high-rate but short-duration bursts!

Temporal Lensing

Idea: “multiple rounds simultaneous impact”

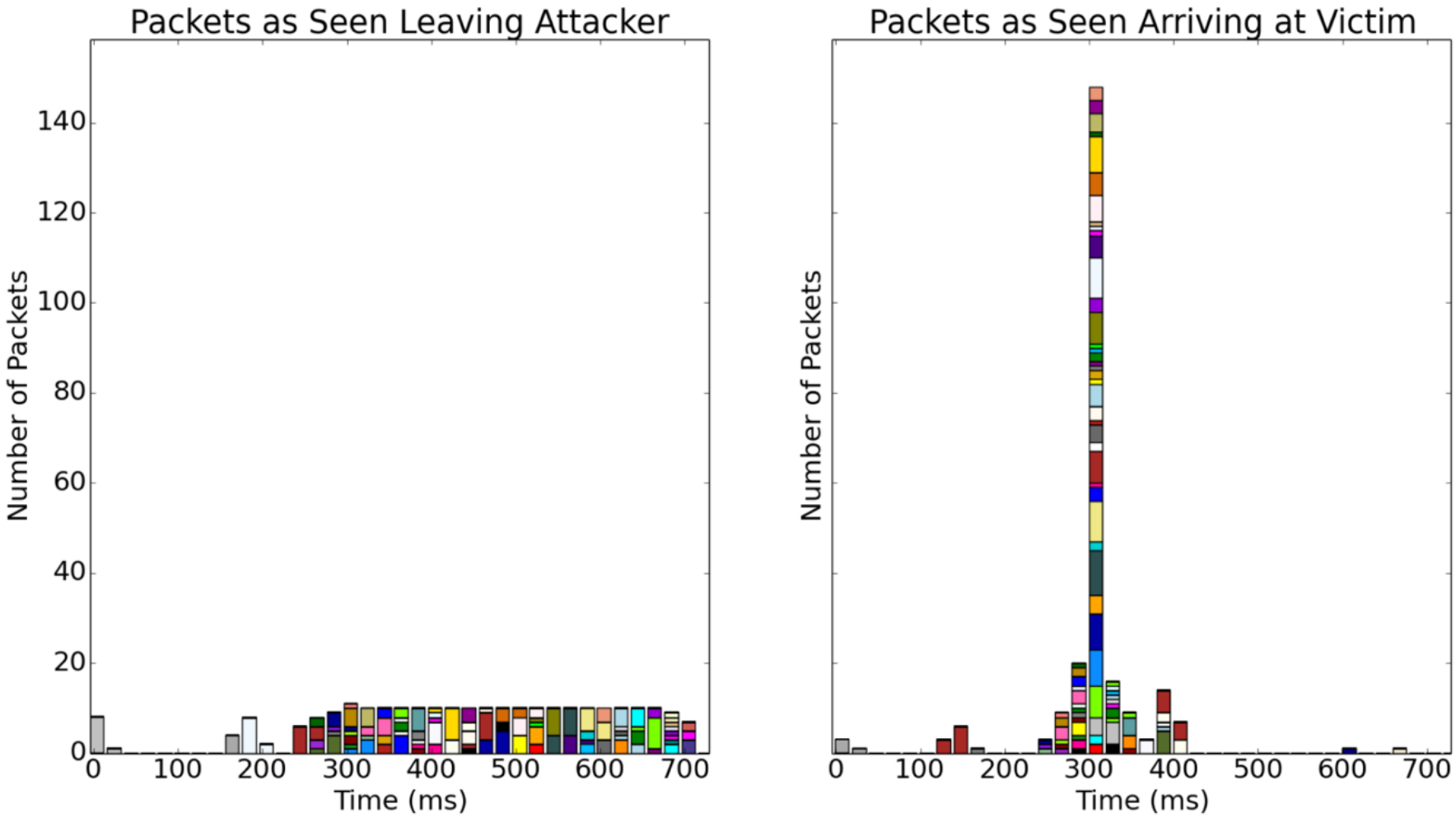
Concentrate a flow in time

So that a low-bandwidth source can also perform a shrew attack



R. Rasti, M. Murthy, and V. Paxson, “Temporal Lensing and its Application in Pulsing Denial of Service Attacks,” in IEEE Symposium on Security and Privacy, 2015.

Temporal Lensing



R. Rasti, M. Murthy, and V. Paxson, "Temporal Lensing and its Application in Pulsing Denial of Service Attacks," in IEEE Symposium on Security and Privacy, 2015.

Recent trends of DDoS

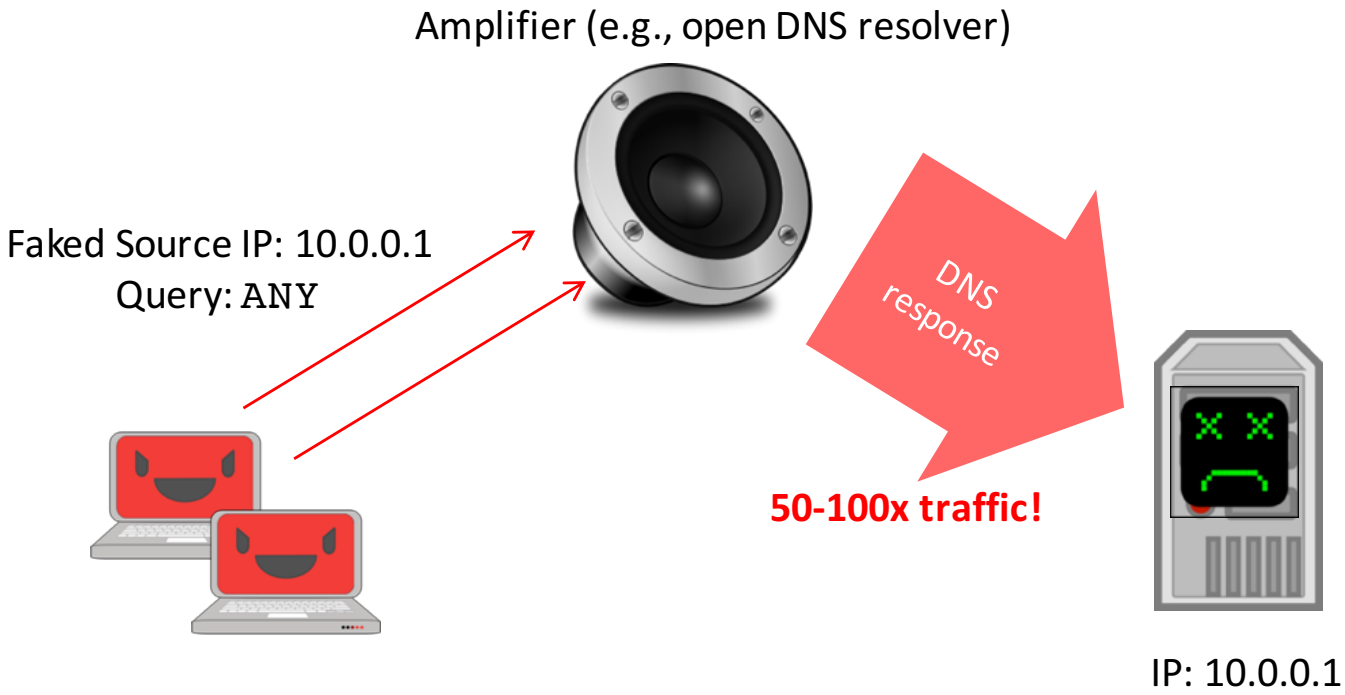
Amplification attack

IoT-based botnets

Browser-based DDoS attack

DDoS attacking Internet infrastructures

DDoS Amplification



Amplification Factor

Memcached: 51,200x

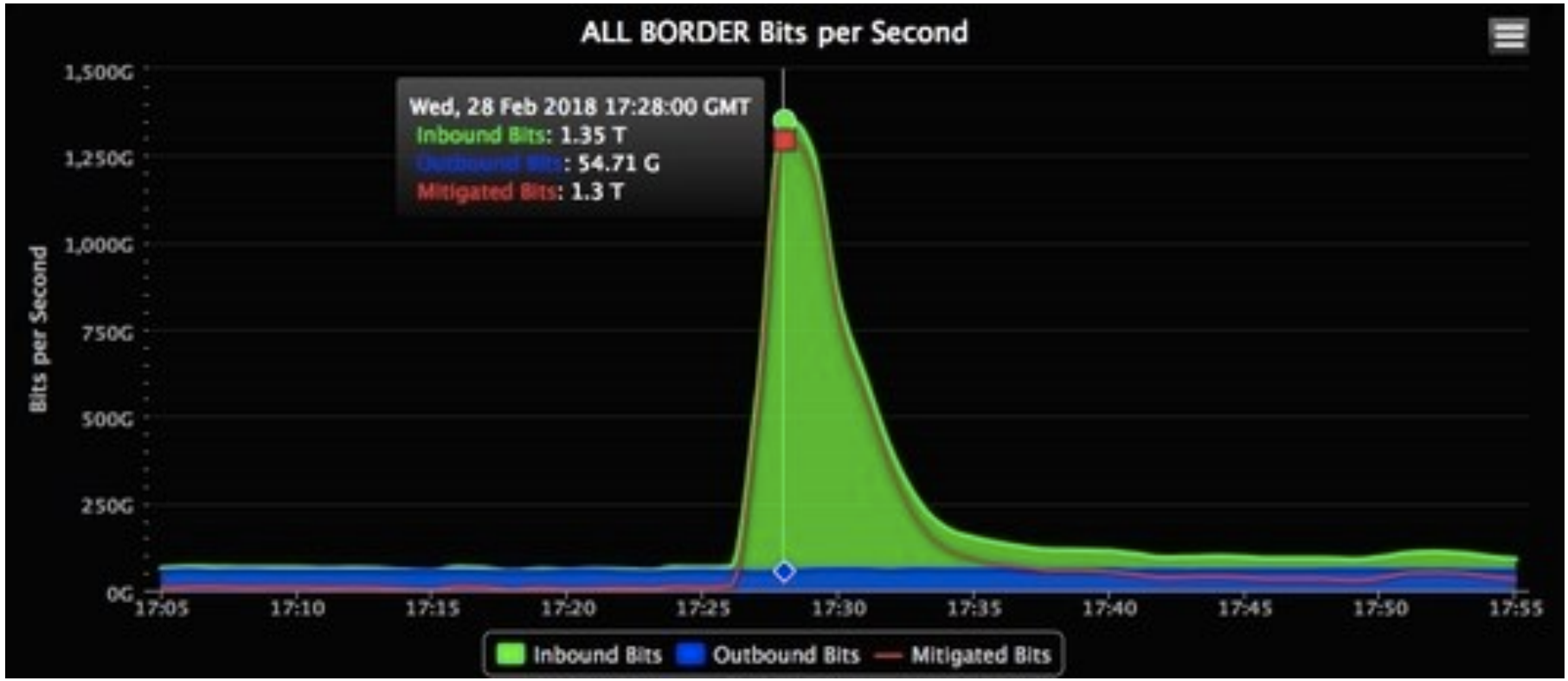
Protocol	BAF *all*	BAF 50%	BAF 10%	PAF *all*	Scenario
SNMP v2	6.3	8.6	11.3	1.00	*GetBulk* request
NTP	556.9	1083.2	4670.0	3.84	Request client statistics
DNS_{NS}	54.6	76.7	98.3	2.08	ANY lookup at author. NS
DNS_{OR}	28.7	41.2	64.1	1.32	ANY lookup at open resolv.
NetBios	3.8	4.5	4.9	1.00	Name resolution
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

Rossow, Christian. "Amplification hell: Revisiting network protocols for DDoS abuse." Symposium on Network and Distributed System Security (NDSS). 2014.

How can we mitigate amplification attacks?

Memcrashed:
DDoS amplification using memcached

Mar. 2018: **memcached amplification** DDoS against Github at **1.3Tbps**



Mirai IoT Botnets

Infected ~1 million IoT devices

- Simple tech: Scan IPv4 space & try 62 default passwords
- Vulnerable devices hacked in 6 mins after going online

Launched largest DDoS in history

- Sep 2016: DDoS at 620Gbps
- Oct 2016: attacked Dyn DNS service provider, taking down GitHub, Twitter, Reddit, Netflix, Airbnb, etc.

“Great Cannon” browser-based DDoS

DDoS Attack Targets Popular Anti-censorship Projects on Github

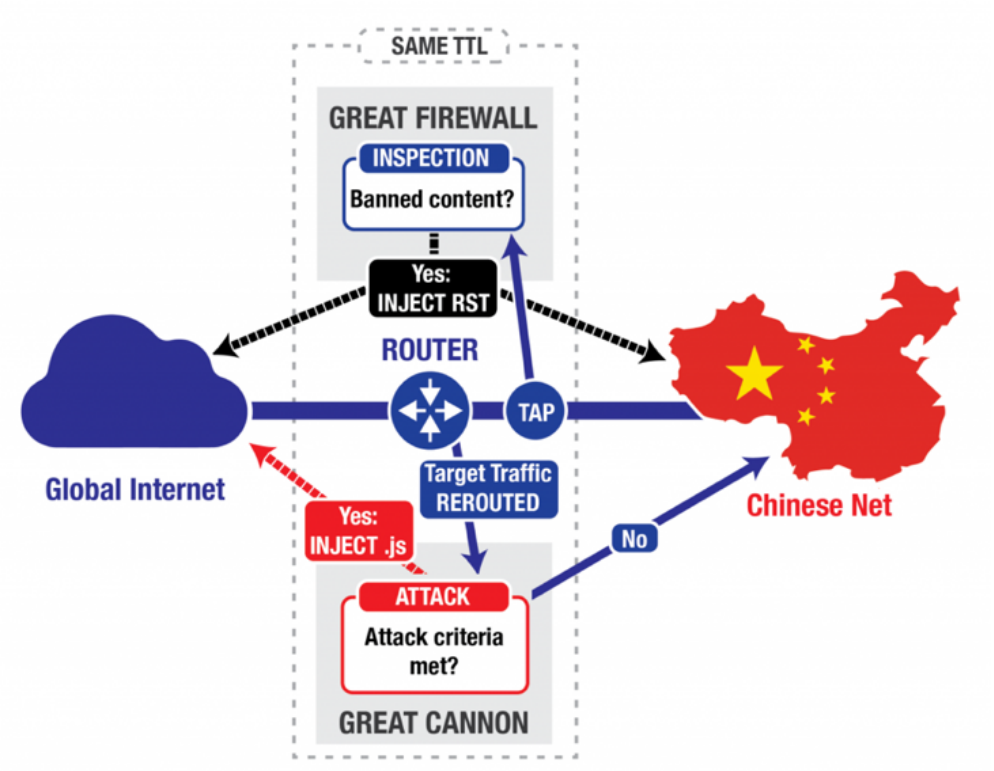
3 days into attack since Thursday March 26, 2015

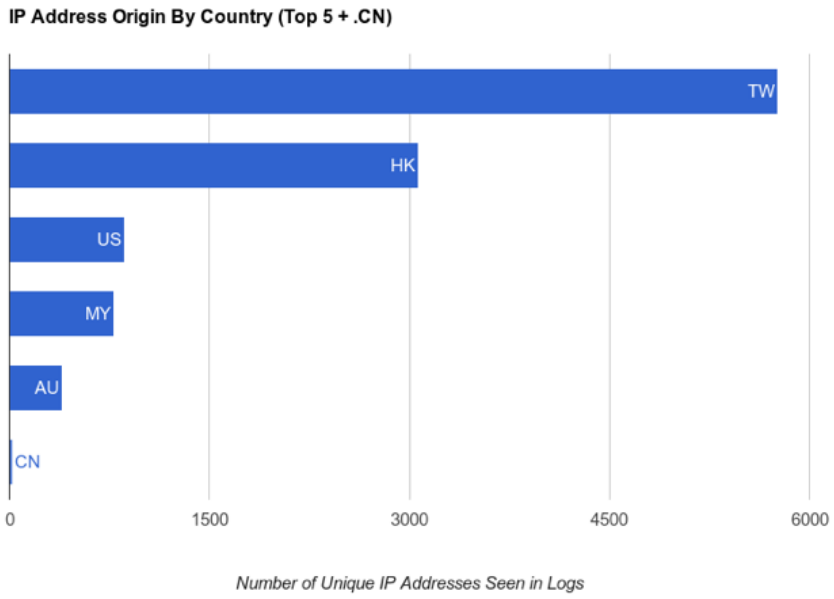
Malicious JavaScript executed when users outside China visited sites with Baidu’s user tracking code

Load https://github.com/greatfire/ and https://github.com/cn-nytimes/ every two seconds

http://insight-labs.org/?p=1682
http://www.wsj.com/articles/u-s-coding-website-github-hit-with-cyberattack-1427638940

“Great Cannon” browser-based DDoS





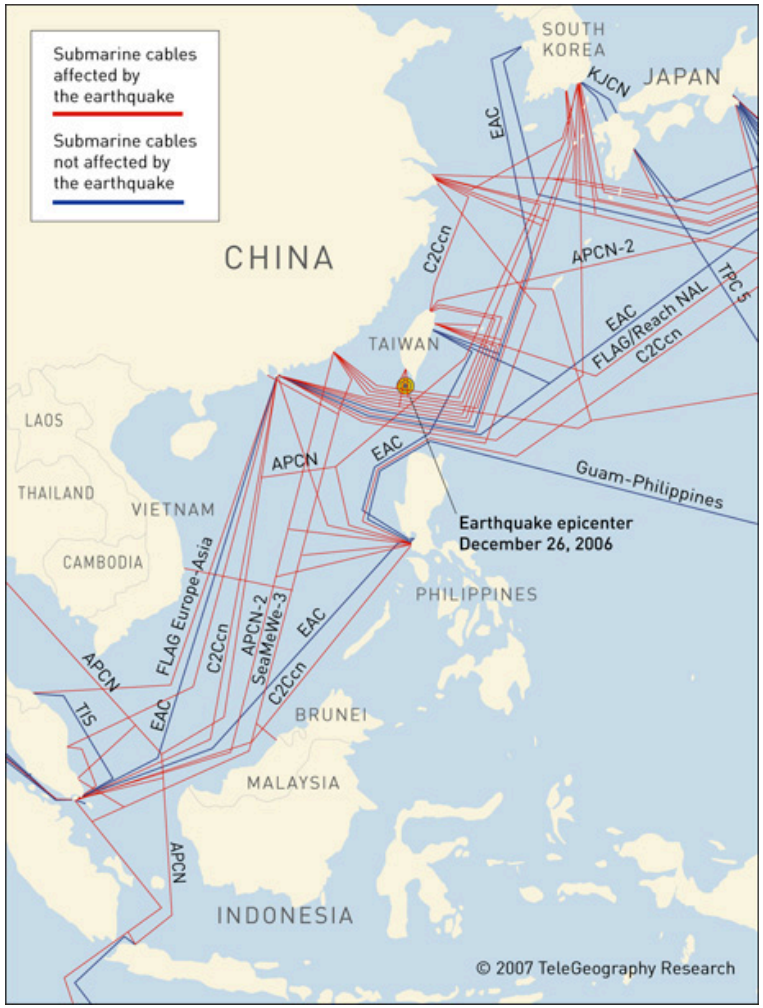
https://citizenlab.org/2015/04/chinas-great-cannon/

DDoS能切斷台灣對外的連線嗎？

2006 恆春地震

- Impaired seven out of nine geographically co-located cables in the Luzon Strait
- A six-hour outage for more than two thousand IP prefixes

「根據台灣網路資訊中心（TWNIC）進行的台灣網際網路連線頻寬調查結果顯示，截至2016年3月底止，我國對外連線總頻寬達1,715,085Mbps」



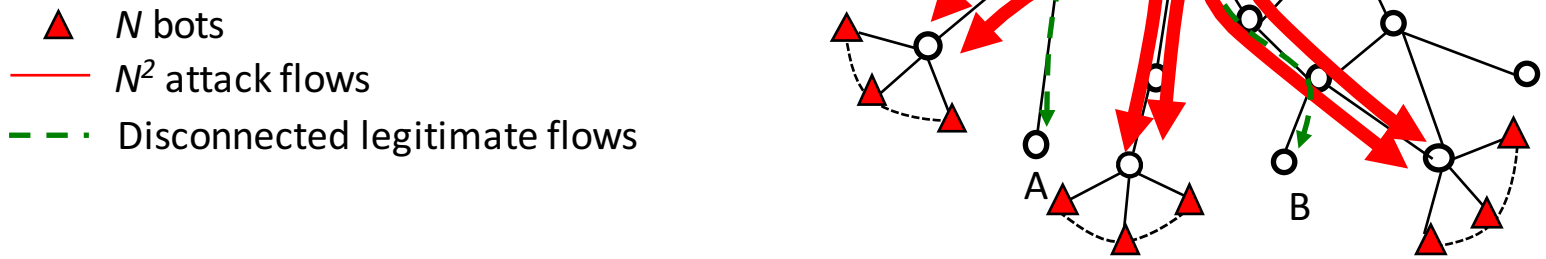
DDoS against Core Links

Coremelt attack (2009), Crossfire attack (2013)

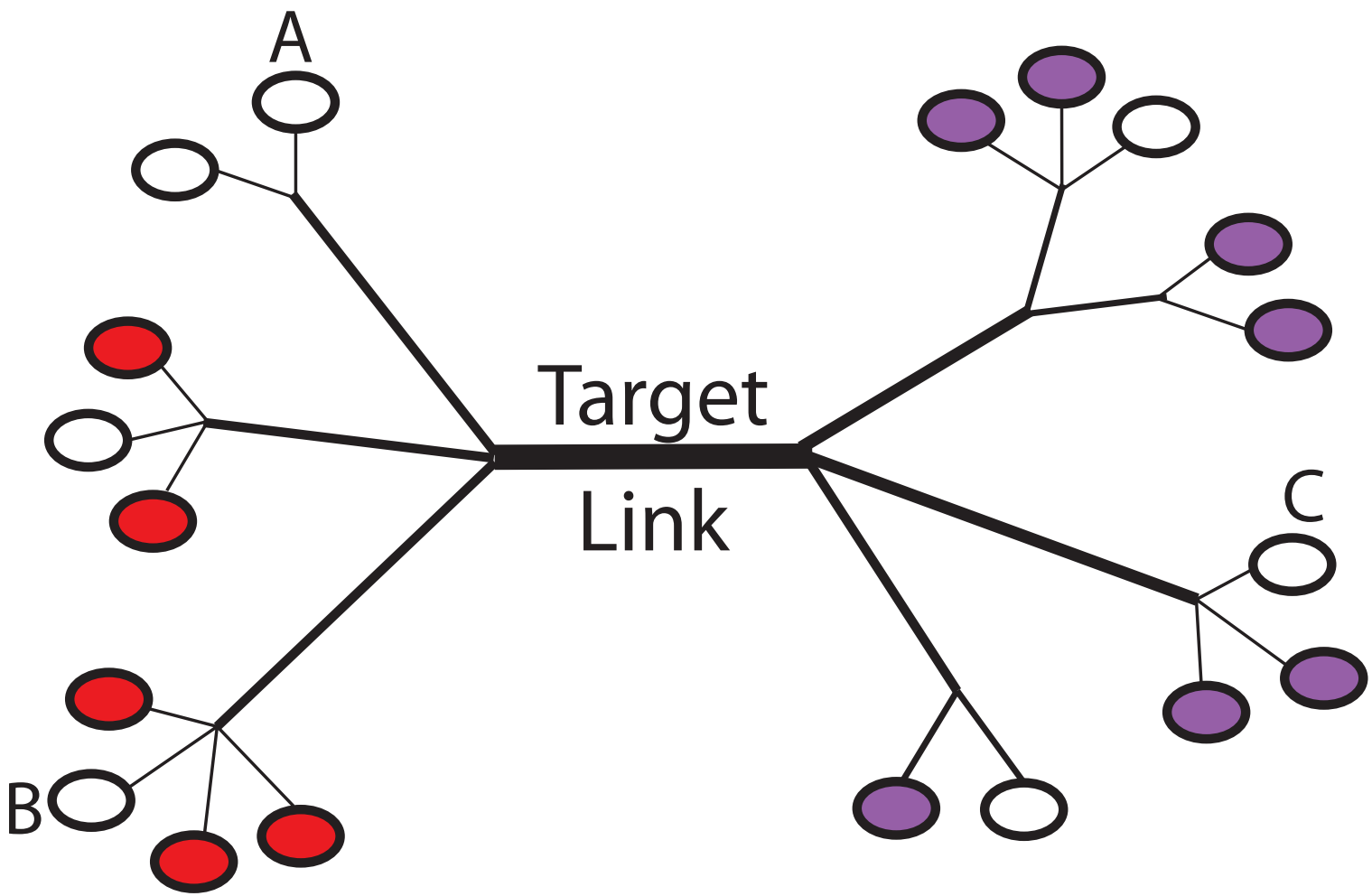
- Congest Internet core links
- Bots send traffic to each other or to decoy servers
- Possible to isolate an entire area by congesting critical links

Bypasses existing DoS defenses

- Traffic is "wanted" so capabilities are acquired
- No obvious reason to filter, looks legitimate
  - Each bot contributes a small amount of traffic



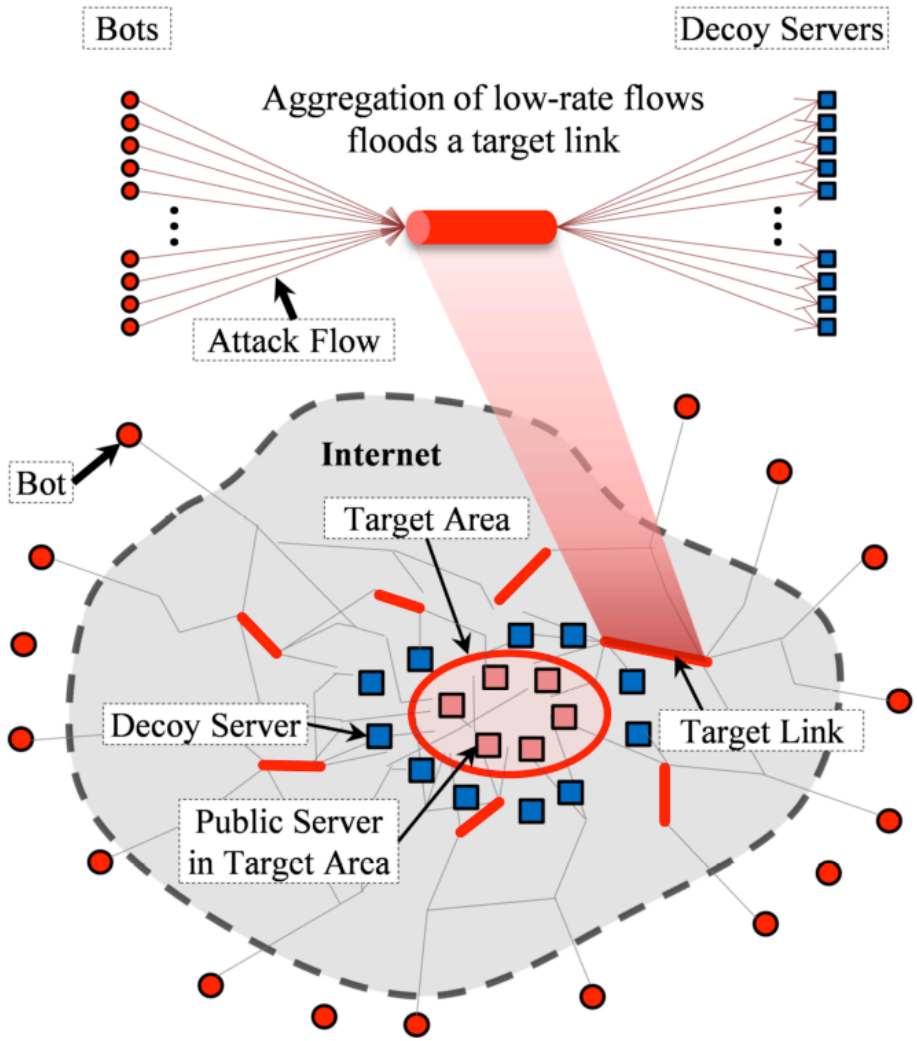
The Coremelt Attack



Crossfire Attack

Cut off a target area by flooding a few selected links

Exploits current Internet's characteristic—a few critical links are responsible for a large portion of traffic to a target area



M. S. Kang, S. B. Lee, and V. D. Gligor, "The Crossfire Attack," in Proceedings of IEEE Symposium on Security and Privacy, 2013.

DDoS Defense

How to mitigate DDoS

硬碰硬比誰資源多？

- 防禦小型攻擊ok
- 面對大型攻擊，就算是Google 也不保證撐得住

香港公投網站DDoS攻擊內幕大公開，連Google、亞馬遜都擋不住

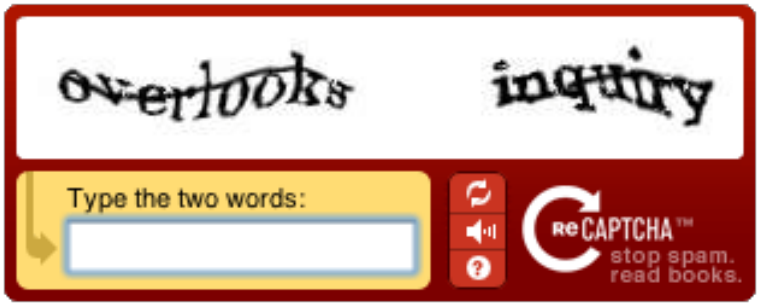
一開始投票，PopVote線上投票網站就遭遇到了超大規模的DDoS攻擊，攻擊流量達網路史上第二高，連找Amazon或Google網路服務支援都擋不住，CloudFlare最後靠著全球網路服務業者聯手，才撐過了這10天投票過程。

http://www.ithome.com.tw/news/90246

How to mitigate DDoS

CAPTCHAs擋bot-based DDoS？

- CAPTCHAs可以外包
- CAPTCHAs有時還會擋人類
- 要先成功建立連線
- Cloud幫忙插CAPTCHAs有隱私問題



How to mitigate DDoS

防火牆把攻擊流量過濾掉就好啦～？?

- 如何準確分辨誰是好人誰是壞人
- 有時打到家門口再過濾已經來不及了
- 防火牆本身也是bottleneck

It’s harder than you might think!

General DDoS Defense Strategies

Overprovisioning/replication
- 硬碰硬，比誰資源多；分散攻擊力道

Traffic differentiation
- 分辨 “好與壞”，移除惡意的連線

Fair sharing
- 公平分配資源，不讓壞人佔便宜

Source identification and takedown
- 從源頭根除

Many mitigation mechanisms combine two or more strategies

DDoS defense mechanisms

Ingress filtering

- Removes packets with illegitimate source IPs

Computational puzzles

- Slows down attacks, achieves per-computation fairness

Cloud- or ISP-based filtering

- Delegates defense to cloud or ISP

Network capabilities

- Allows victim to block unwanted traffic closer to the source

IP traceback

- Reveals the real source IPs of packets

Ingress Filtering

IP Spoofing

Forging the source IP address of an IP packet

Why IP spoofing?

- Concealing the sender's identity
- Impersonating another entity

Exploited by many DDoS attacks

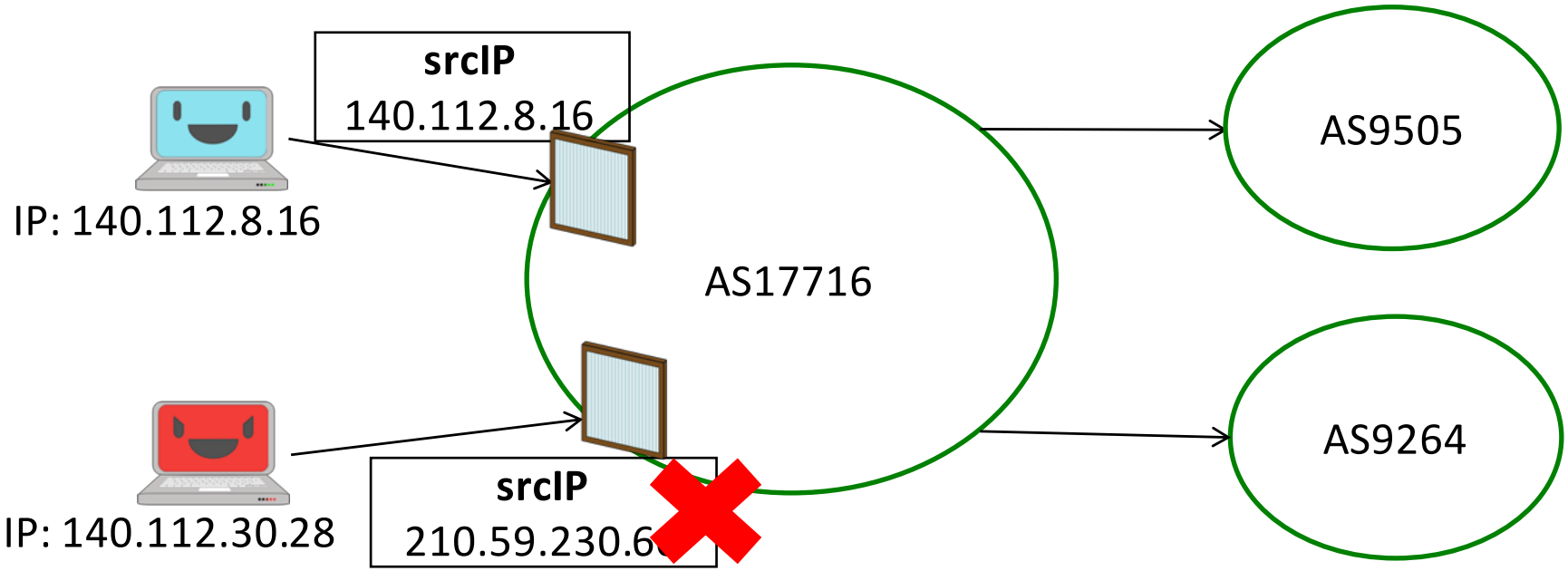
A fundamental problem in the current Internet architecture

- Packets are routed based on destination IP addresses
- No explicit authentication of source IP addresses

Ingress Filtering

Aims to mitigate IP spoofing

RCF 2827 / BCP 38 - Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing



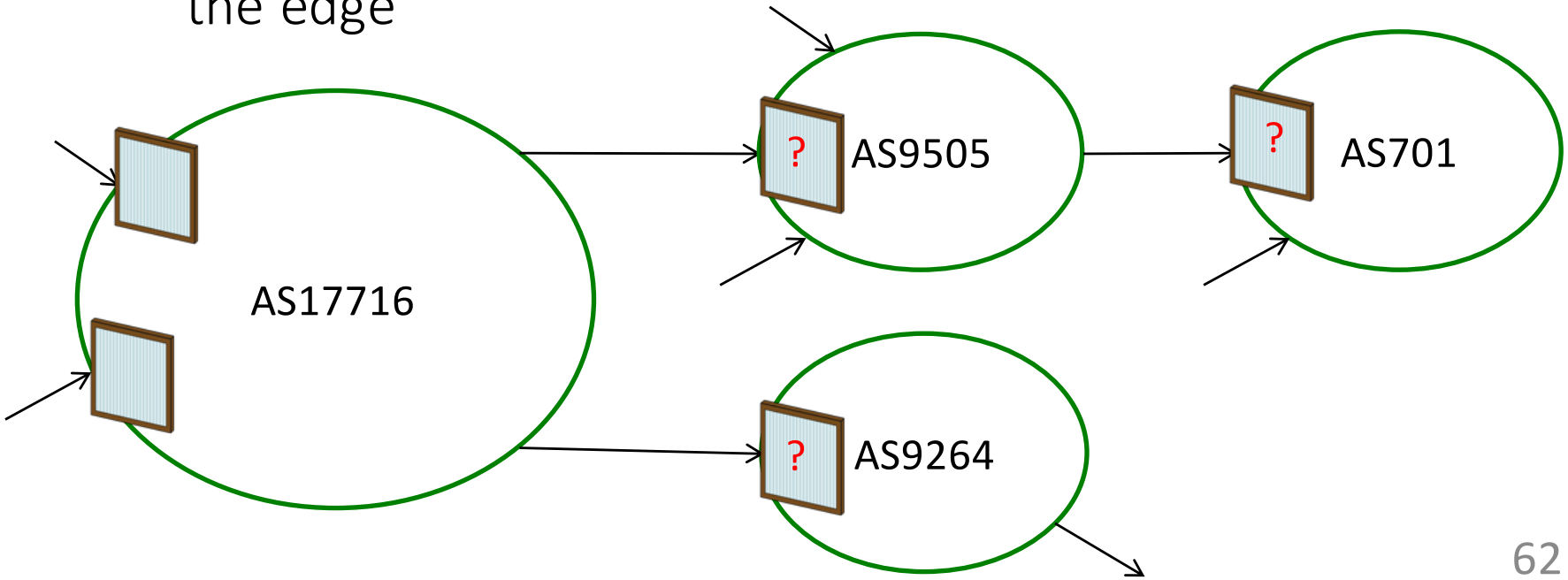
Ingress Filtering

Only forwards packets with **legitimate** source IPs

- "Legitimate" is defined as "expected at the ingress point"

Challenge: How can an Autonomous System (AS) know which IPs are legitimate?

- Hard to determine unambiguously as it moves away from the edge



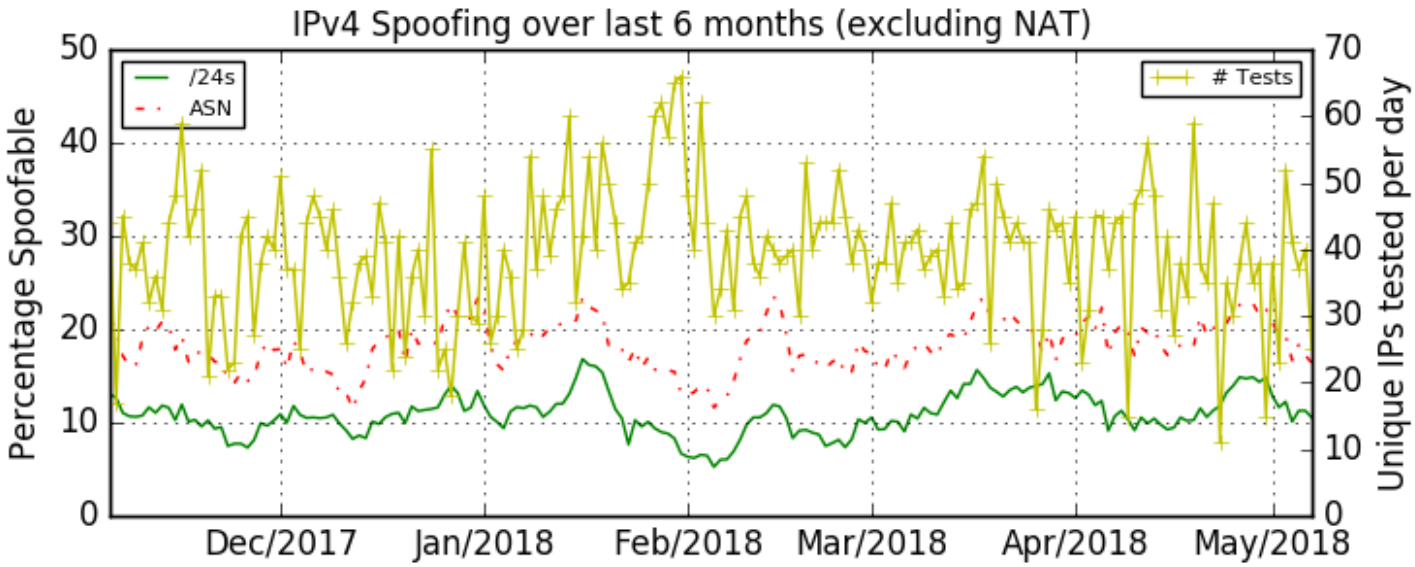
Deployment Issues of Ingress Filtering

Require 100% deployment to be effective

- Less effective as moving away from the source

No incentive for ISPs to be an early adopter

- Preventing spoofing does not necessarily makes one's own network less vulnerable



https://spoofer.caida.org/

Discussion: Ingress Filtering

Widespread (if not universal) deployment is required for ingress filtering to be effective

Even if ingress filtering were universally deployed, the attacker can still spoof the source addresses within the same subnet

DDoS can be effective without IP spoofing given the size of today's botnets

Proof of Work

Client Puzzles

Goal: prevent attacker from consuming excessive server resources

Idea: slow down attacker with *puzzles*

- Puzzle is a proof-of-work
- Server can quickly generate and verify a puzzle
- Server sends a unique puzzle to client
- Attacker & clients have to spend time computing puzzle solution
- Server only accepts requests with a solved puzzle

Juels, Ari, and John G. Brainard. "Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks." NDSS. Vol. 99. 1999.

Approach #1

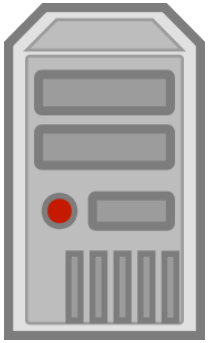
1. C->S: request

2. S creates random number x, computes its hash and takes the last r bits : $y = [H(x)]_r$

2. S->C: H, y, r

3. C searches x s.t. last r bits match y: $[H(x)]_r = [y]_r$

4. C->S: H, y, r, x



Client's effort?

How can the server verify?

Problems?

Approach #2

Idea: server S uses a secret key K to generate puzzle challenge and to verify authenticity of puzzle solution without keeping any state

- Secret `T = MAC(K, Client_IP || Client_port)`
- Use T for puzzle generation and solution verification

Approach #2

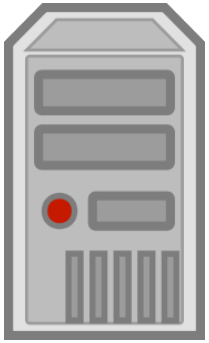
1. C->S: request

2. S computes:
T = MAC(K,Client_IP||Client_port)

2. S->C: T, r

3. C searches x s.t. last r bits match T: $[H(x)]_r = [T]_r$

4. C->S: T, r, x



Client's effort?

How can the server verify?

Problems?

Approach #3

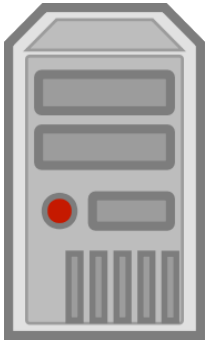
1. C->S: request

2. S->C: T, r, t

3. C searches x s.t. last r bits match T: $[H(x)]_r = [T]_r$

4. C->S: T, r, x, t

2. S computes:
T = MAC(K,
Client_IP||Client_port||t||r)



Client's effort?

How can the server verify?

Problems?

Discussion: Client Puzzles

Advantages

- Simple
- Can be done automatically
- Puzzle's difficulty level is tunable

Disadvantages

- Consume client's CPU
- Require changes to both client and server
- Not so fair for less-powerful devices
- Puzzle difficulty grows exponentially

More Proof-of-* Schemes

Memory-based

- A. Biryukov and D. Khovratovich, “Equihash: Asymmetric proof-of-work based on the Generalized Birthday problem,” in Proceedings of NDSS, 2016.

Bandwidth-based

- M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, “DDoS Defense by Offense,” in Proceedings of ACM SIGCOMM, 2006.

Proof-of-stake

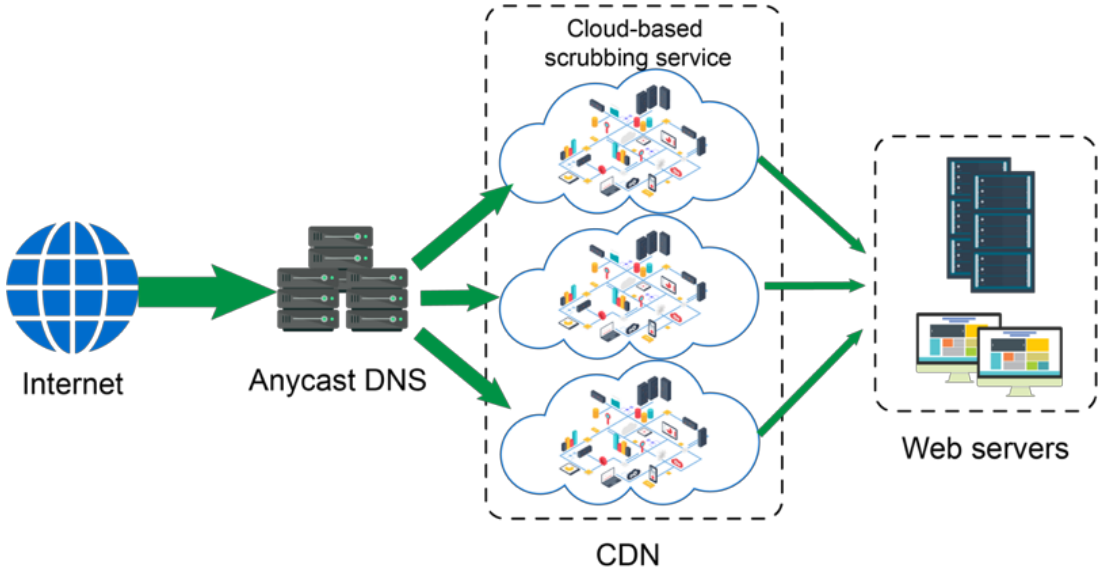
Proof-of-elapsed-time

Cloud- or ISP-based Filtering

Cloud-based DDoS Mitigation Service

By changing BGP or DNS of web server, the traffic is redirected to the provider as a middle-man

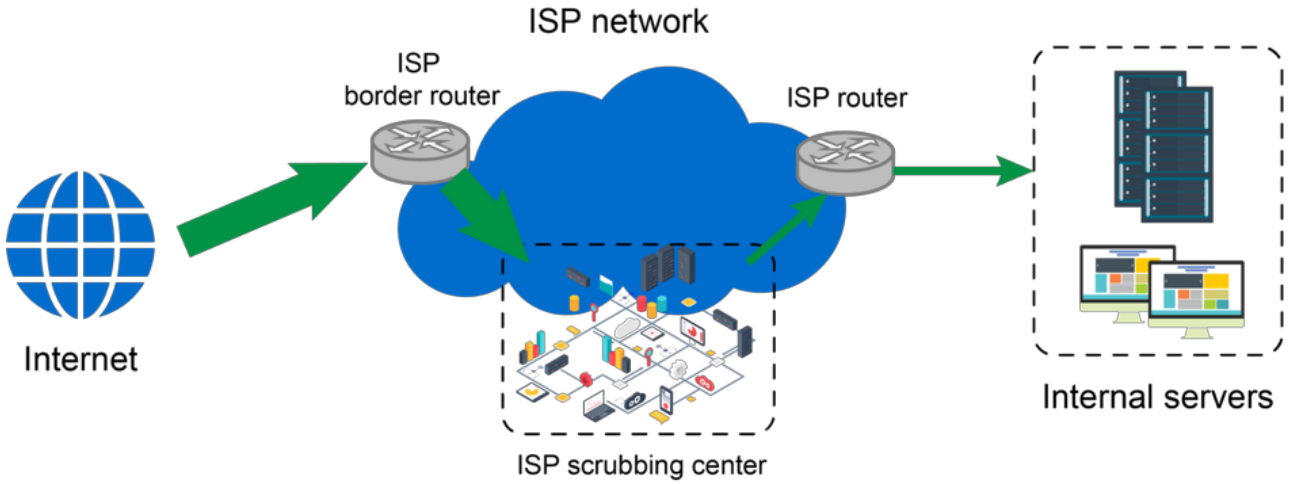
Some provide Content Delivery Network (CDN) service to achieve diversion of traffic



ISP-based DDoS Mitigation Service

ISP redirect the traffic to the scrubbing center, then send good traffic back to the destination

Scrubbing center uses Deep packet inspection (DPI) and connection pattern to filter malicious traffic



Discussion: Cloud- or ISP-based Filtering

Cloud-based security provider can be easily bypassed

- Because most cloud use DNS to redirect traffic, attackers can easily bypass the proxies if the victim's IP is exposed

Privacy violation

- E.g., Radware decrypts HTTPS and injects CAPTCHAs to client
- An untrusted or compromised cloud could expose users' sensitive data

No destination control

- 攻擊越早擋掉越好、但資訊要越靠victim server才越多，特別是application-layer attacks
- 使用者不知道也無法控制cloud filter policy

High cost for small-, medium-size organizations

- How about our department?

Network Capabilities

Acknowledgment: some slides are provided by Prof. Adrian Perrig

Fundamental Problem of DDoS

DDoS attacks exploit a fundamental problem of the current Internet: receiver has no control over who can send traffic to it

How to enable receiver to stop misbehaving senders as early as possible?

Challenges

- Need per flow state in network?
- Where to filter?
- Need trust relationships between ISPs?
- Routers need to authenticate receiver requests to stop flows?

Network Capabilities

Goal: enable receiver to control its traffic w/o installing stateful filters

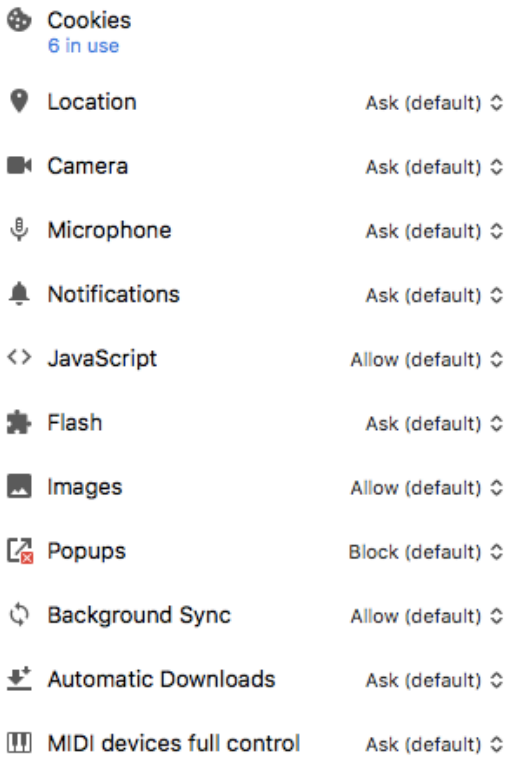
Key ideas

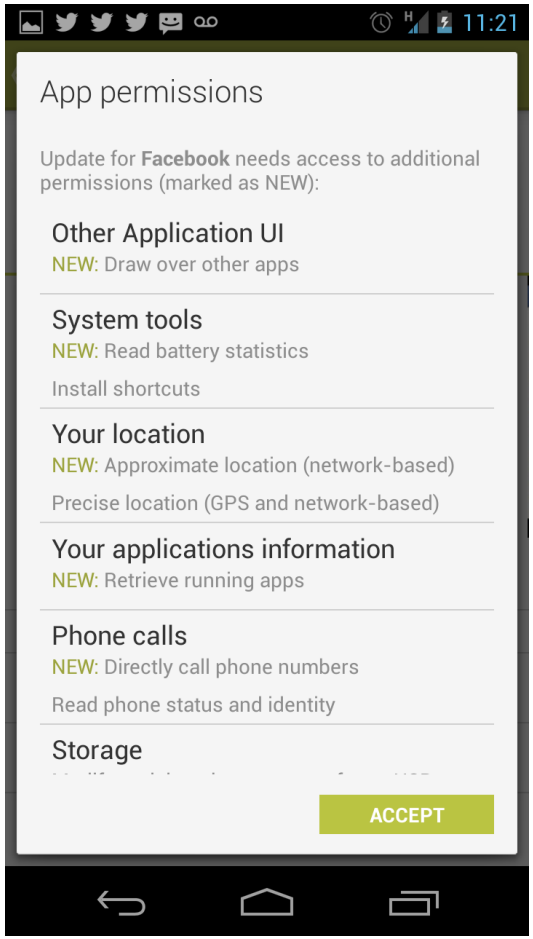
- Network capability for traffic authorization
- Only clients with valid capability get authorization
- Authorized or "privileged" packets get priority over non-privileged packets

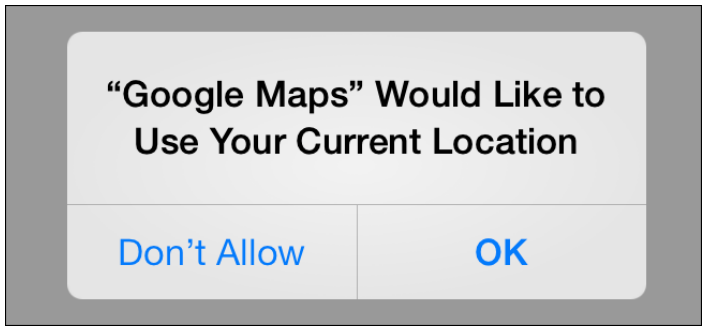
A. Yaar, A. Perrig, and D. Song, "SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks," in Proceedings of IEEE Symposium on Security and Privacy, 2004.

Capabilities

Borrow the term *capabilities* from OS security



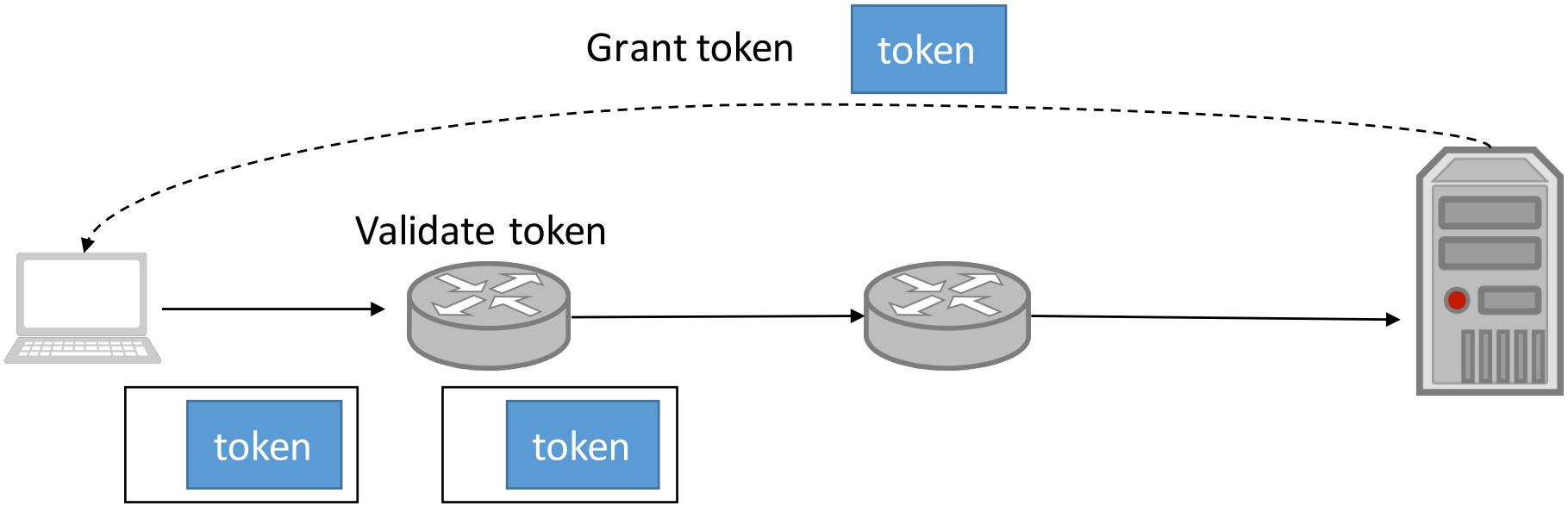




Network Capabilities

Borrow the term *capabilities* from OS

Network capability = a token carried by a packet that indicates the privilege level of the packet



SIFF: Stateless Internet Flow Filter

Create two Internet packet classes

- Unprivileged (best-effort): Signaling and legacy traffic
- Privileged: Receiver-controlled traffic flows

Privileged packets are given *priority* at routers

- Privileged packets are never dropped by unprivileged flooding

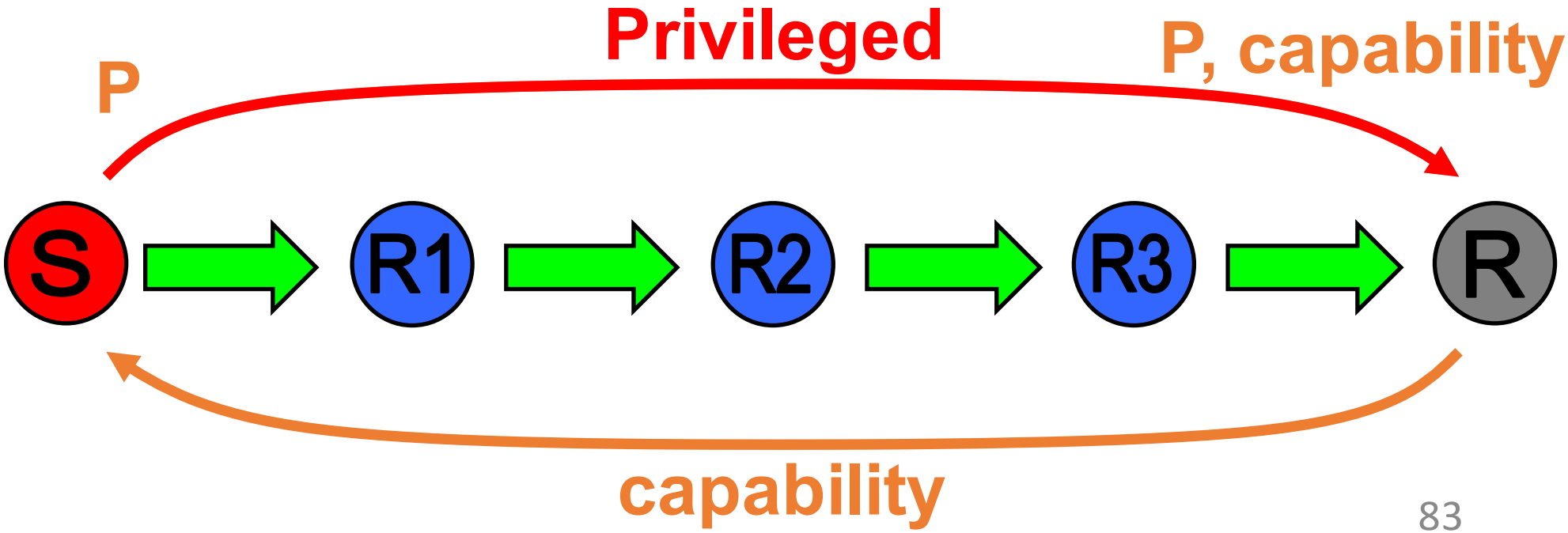
Privileged packet flooding is *accountable*

SIFF Handshake

Sender *S* sends best effort packet to receiver *R*, arriving packet accumulates capability

If *R* wants to allow *S* to send privileged traffic, *R* sends capability back to *S*

S includes capability in packets to send at privileged level



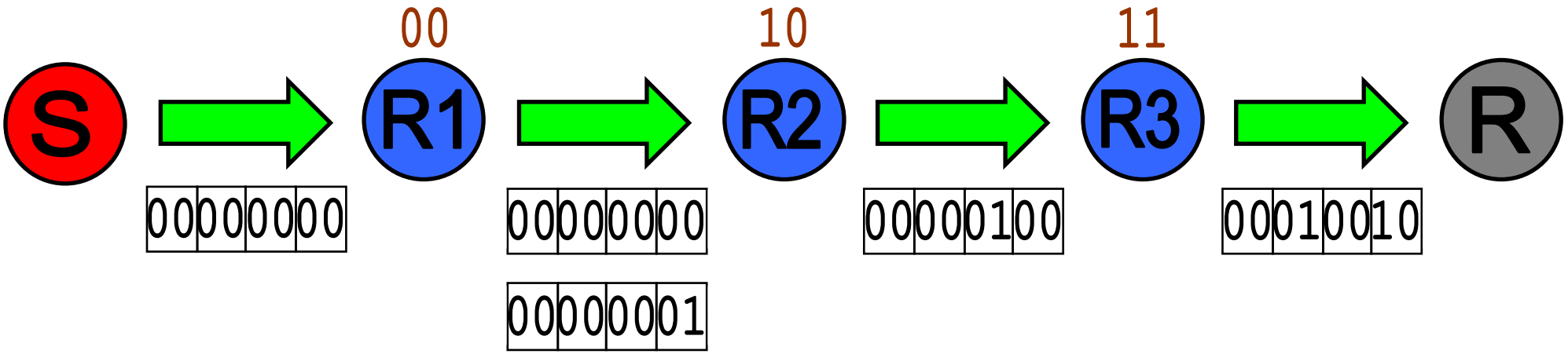
SIFF Marking: Unprivileged Packets

SIFF routers mark unprivileged packets

- With all zeros in the capability field, router pushes extra 1 bit to signal the length

Markings unique to Sender/Receiver pair

- Add the source IP and destination IP to hash
- i.e. MAC(K, currIP | prevIP | senderIP | recvIP | TTL)



SIFF Marking: Privileged Packets

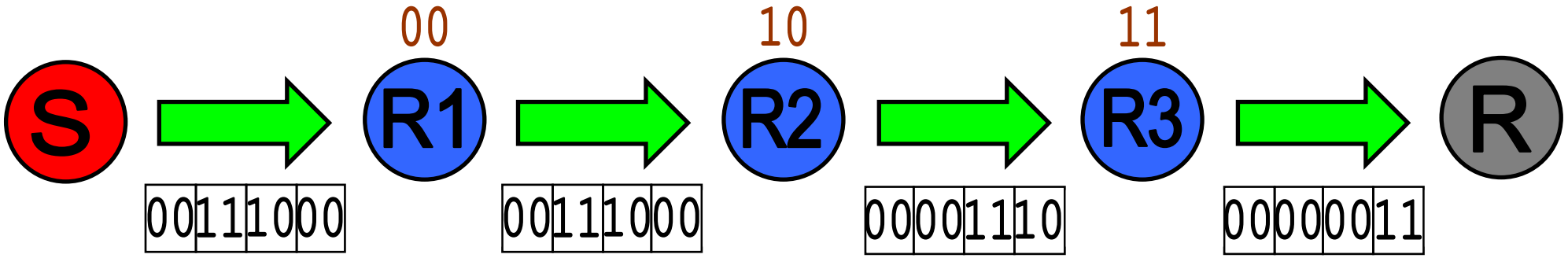
SIFF routers verify marking in the header

- Correct marking: Router pushes zeros into MSB
- Incorrect marking: Router drops packet

Sender cannot learn capability without Receiver's help, and thus can't send privileged traffic

IP Spoofing

- Receiver's capability does not reach attacker



Problem: Static Privilege

Once received, sender can abuse the capability

Goal: Dynamic Privilege

- Expire capabilities over time

Solution: Key Rotation

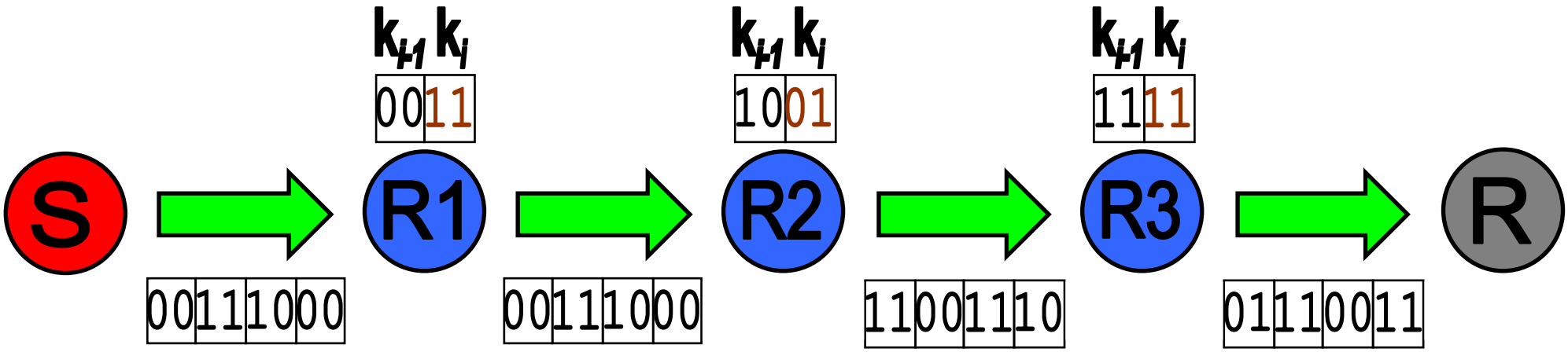
- Routers change keys periodically
- Receiver automatically gets new capabilities

SIFF Marking: Multiple Keys

Each router has two keys: an old key and a new key

On each router, either old marking or new marking enables packet to pass

Routers push new marking into MSB



Receiver-controlled Flows

As packet flow carries on, receiver receives updated markings

If receiver wants to continue to enable sender to send privileged traffic, receiver sends updated marking as capability to sender

If receiver wants to terminate malicious flow, receiver simply stops updating sender with new capability, and routers will soon stop the flow early in network

SIFF Discussion

DoS-less sender/receiver communication

- Receivers can stop malicious flows
- One unprivileged packet establishes privileged connection

Routers need to update every packet

Denial of capabilities

SIFF requires more header space

- Marking Field (128 bits)
- Flags Field (3 bits)
- [Optional] Update Field (128 bits)

Can an attacker forge a valid capability?

IP Traceback

Acknowledgment: some slides are provided by Prof. Adrian Perrig

IP Traceback

We would like to trace attack packets (even with spoofed source addresses) back toward their *origin*

1. The bot machines initiate attack traffic
2. The Command and Control (C&C) server controls bots
3. The criminal administrates the C&C server

S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," ACM SIGCOMM Comput. Commun. Rev., vol. 30, no. 4, pp. 295–306, 2000.

- First IP traceback work

Approximate Traceback Problem

V: victim

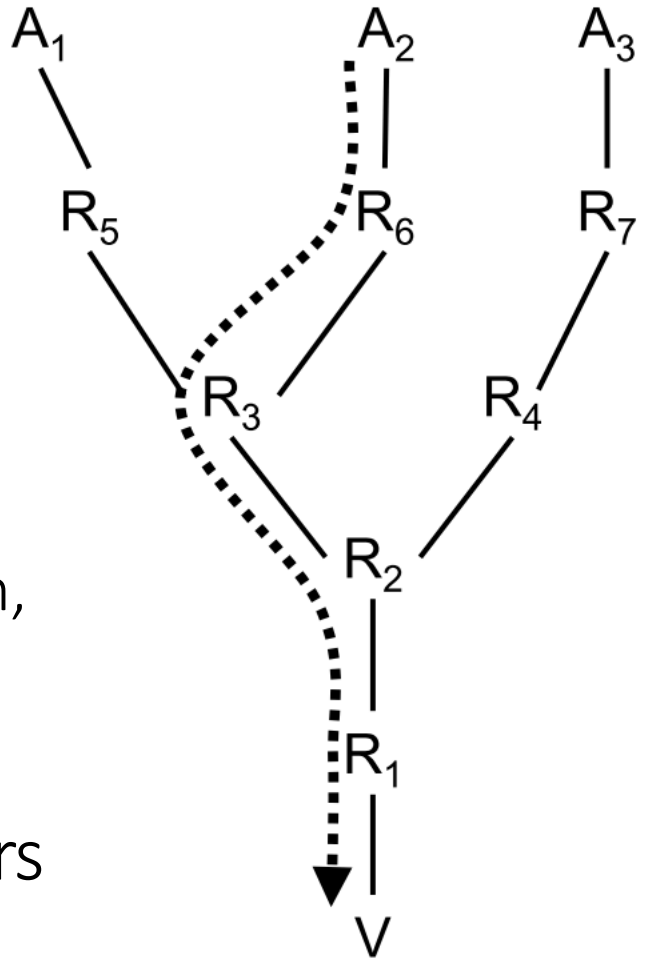
R: router

A: attacker

Approximate traceback problem:
Finding a path whose suffix is a real attack path

- If (R6, R3, R2, R1) is a real attack path, (R5, R6, R3, R2, R1) has a valid suffix

Exact traceback is hard because attackers can inject additional routers



Network seen from the victim

Threat Models and Assumptions

An attacker may generate any packet

Multiple attackers may conspire

Attackers may be aware they are being traced

Packets may be lost or reordered

Assumptions:

- Attackers send numerous packets
- The route between attacker and victim is fairly stable
- Routers are not widely compromised

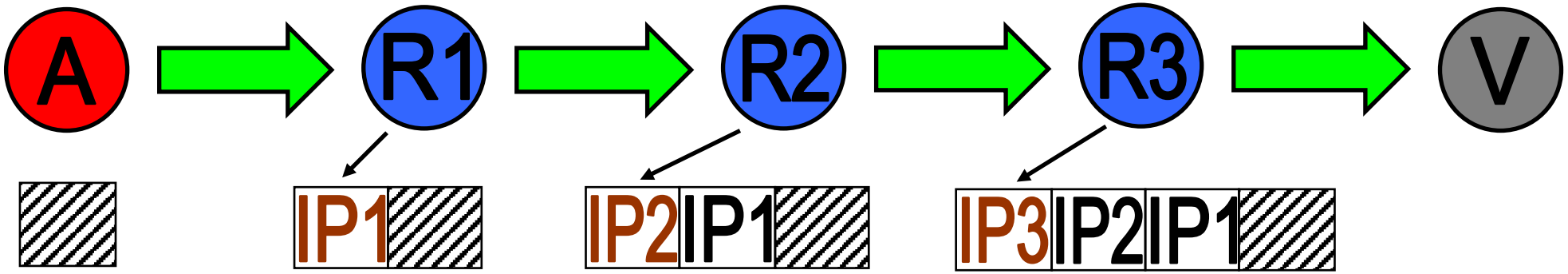
Think about what would happen if any of the assumptions is violated

Simple IP Traceback

Approach: routers add their IP address to each forwarded packet

Problems?

- Can't pre-allocate space
- Increasing length of packet is slow at routers
- More fragmentation



IP Traceback: Probabilistic Packet Marking

Idea: 把attack path的資訊分散存在多個封包裡

Routers mark packets: A router adds additional information to packets about the path they are traveling

- Marking = adding information

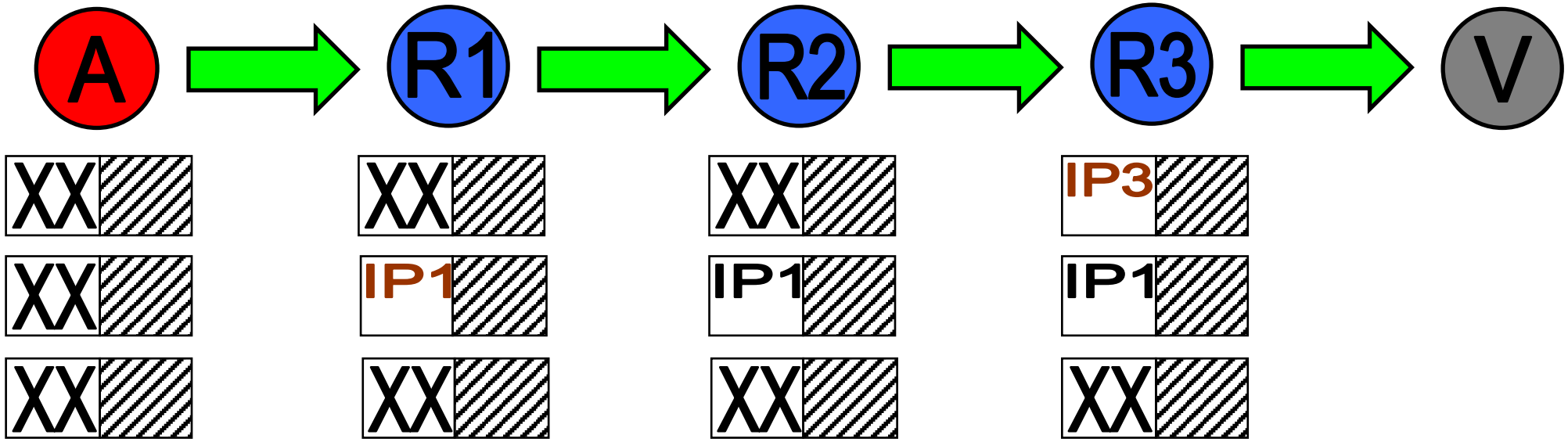
The victim reconstructs the attack path: using information in marked packets to reconstruct attack paths

Evaluation metrics: Convergence time, reconstruction overhead, required space per packet

Approach #1: Node Sampling

Routers mark a static 32-bit field with their IP address with probability *p*

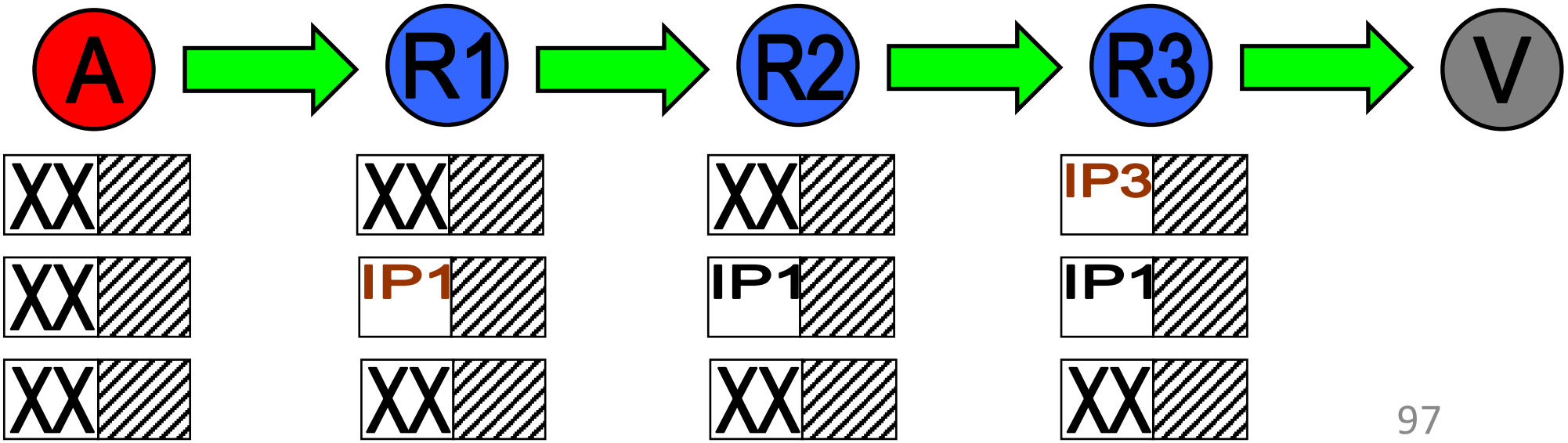
How can the victim reconstruct the path?



Approach #1: Node Sampling

If every router marks a packet with probability *p*, what is the probability that the victim sees a packet marked by R1? How about R2 and R3?

- Pr[a packet marked by R1] = $p(1-p)^2$
- Pr[a packet marked by R2] = $p(1-p)$
- Pr[a packet marked by R3] = p



Approach #1: Node Sampling

Marking: Routers mark a static 32-bit field with their IP address with probability *p*

Reconstruction: The victim orders the routers based on the the number of marked packets

Problems?

- Not robust against multiple attackers (there can be multiple routers at the same distance)
- Inferring router order from sampling distribution is slow

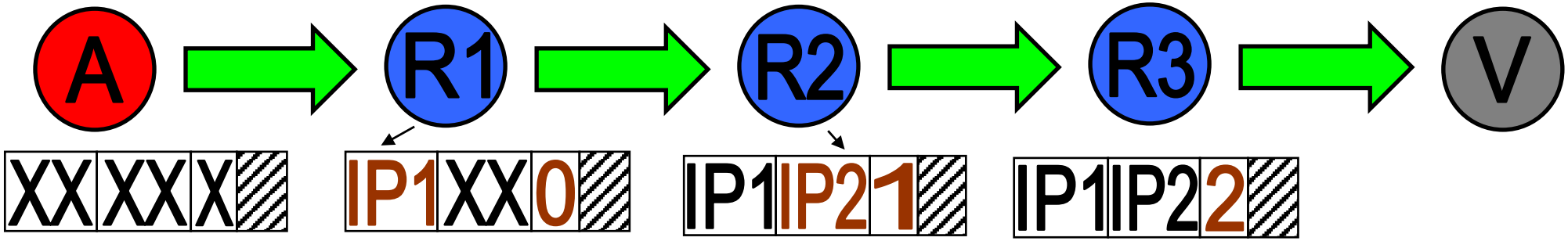
Approach #2: Edge Sampling

A router probabilistically adds its IP address and TTL

Else, router checks if previous router added IP address (current TTL == Marked TTL?)

- If yes, router adds its IP address
- Else, router does nothing
- Router increments TTL

How can victim reconstruct attack path?



Marking procedure at router R:

for each packet w
  let x be a random number from [0..1)
  if $x < p$ then
    write R into w.start and 0 into w.distance
  else
    if w.distance = 0 then
      write R into w.end
    increment w.distance

Path reconstruction procedure at victim v:

let G be a tree with root v
let edges in G be tuples (start,end,distance)
for each packet w from attacker
  if w.distance = 0 then
    insert edge (w.start,v,0) into G
  else
    insert edge (w.start,w.end,w.distance) into G
remove any edge (x,y,d) with $d \neq$ distance from x to v in G
extract path (R_i..R_j) by enumerating acyclic paths in G

Biggest problem here:
extra space (72 bits) needed in packets

32-bit IPs x2

8 bits for distance

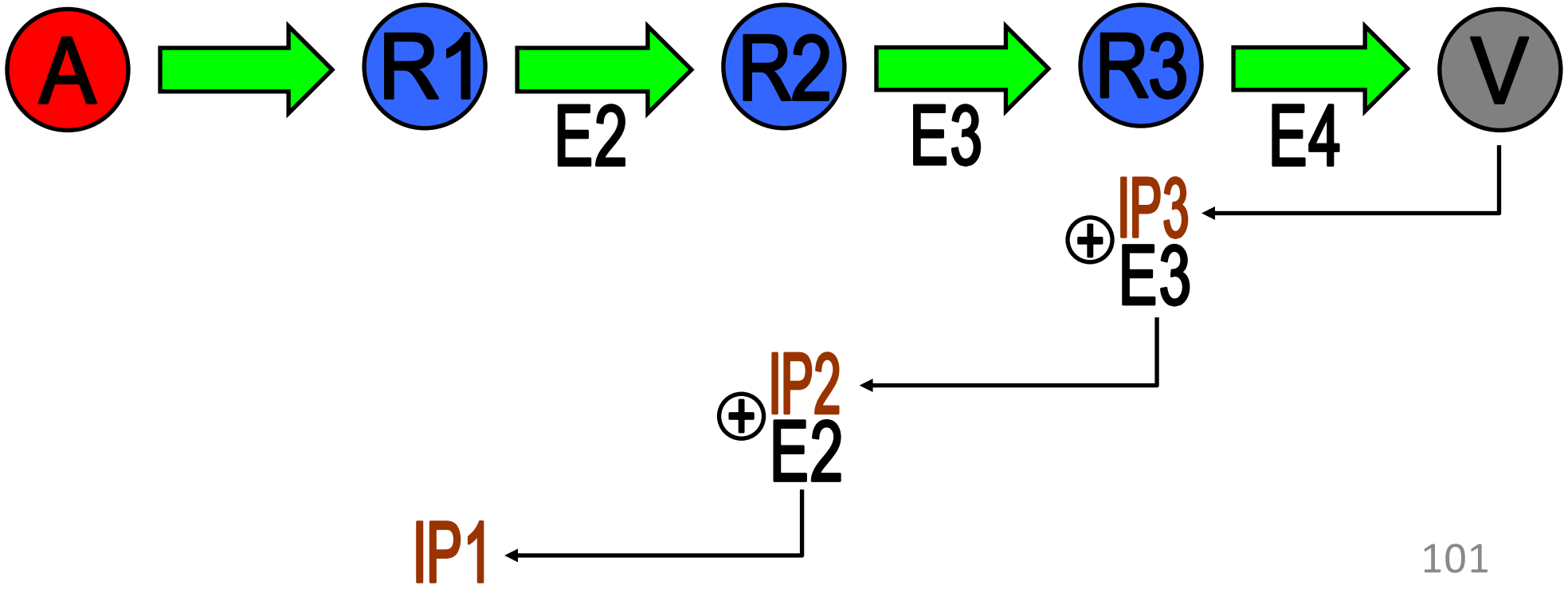
Edge Sampling Improvement I: XOR

Represent an edge (R_x, R_y) as R_x XOR R_y to save 32 bits

Reconstruction:

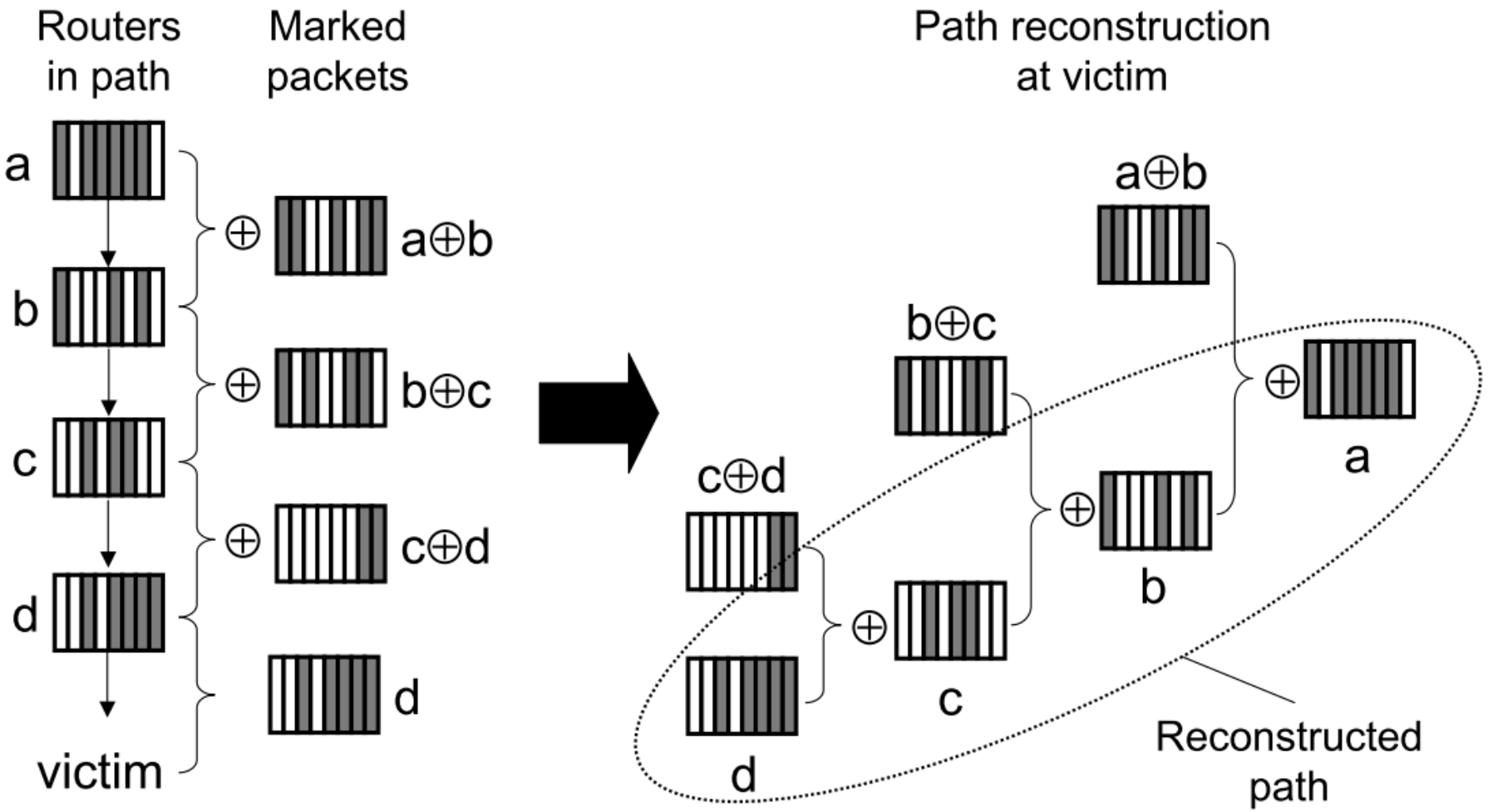
- For distance *i* = 0 to *end*

  take incomplete edge, *i,* and XOR with edge *i*+1, for IP_{i+1}



Marking

Reconstruction

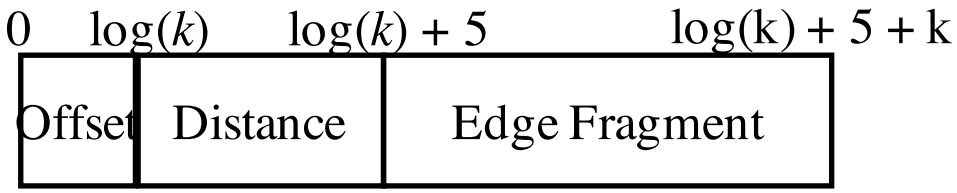


Edge Sampling Improvement II: Fragment Sampling

Split each IP address into *k* smaller fragments. When marking, randomly select fragment to mark with

Problem?

- IP fragments are not unique. No way to tell which fragments belong together in multiple path attack

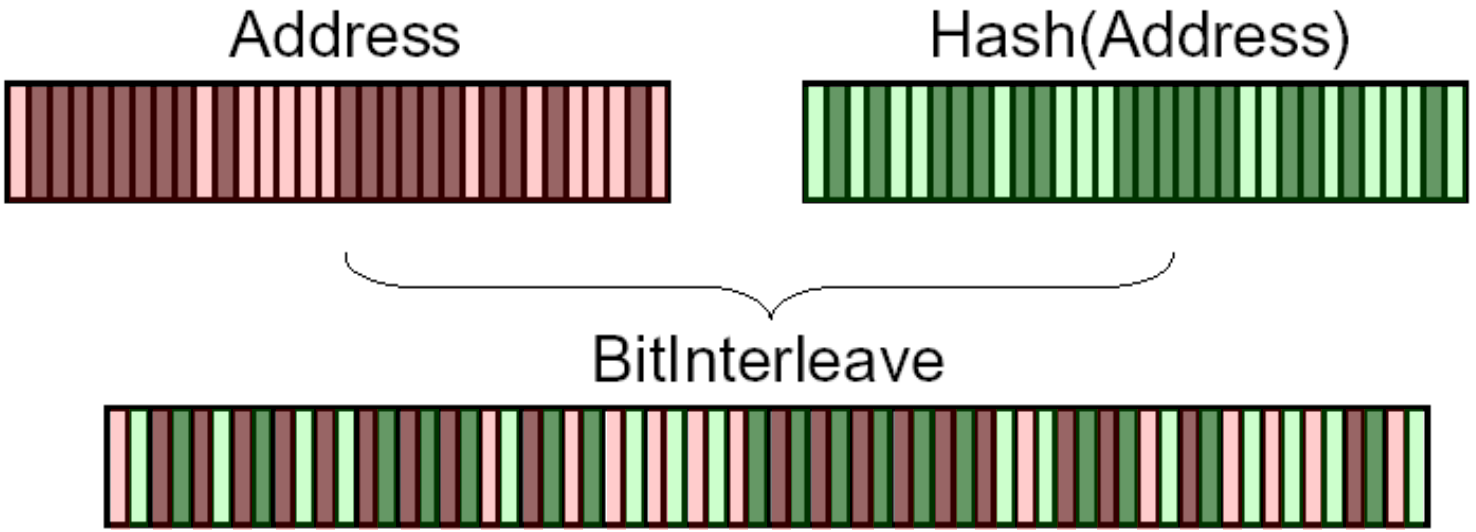


Edge Sampling Improvement III: Hash Check

Calculate the hash of the IP and interleave its bits with the IP itself to help reconstructing fragments

Problem?

Scalability: Computational overhead increases exponentially with the number of attacks because the victim has to check every possible combination



Threat Models and Assumptions

An attacker may generate any packet

Multiple attackers may conspire

Attackers may be aware they are being traced

Packets may be lost or reordered

Assumptions:

- Attackers send numerous packets
- The route between attacker and victim is fairly stable
- Routers are not widely compromised

Think about what would happen if any of the assumptions is violated