(D)DoS

Part 2: Specific Attacks and Defense Mechanisms

Network Security AS 2020

1 December 2020

Adrian Perrig
(some slides by M. Legner, S. Frei, T. Dübendorfer, H.-C. Hsiao)

ETH zürich

What are (distributed) denial-of-service attacks?

- Denial-of-service (DoS) attacks try to **make a service or network resource unavailable** to its intended/legitimate users.
- Typically achieved by **exhausting available resources** by sending an excessive amount of traffic/packets/requests.
- Distributed DoS (DDoS) attacks use many different sources simultaneously, often by creating and using so-called *botnets*.
- DDoS attacks are often used to extort companies: “Pay XX bitcoin and the attack will stop”

What are attack targets?

	Network links	Network devices / networking stack	Applications
Description	Volumetric attack	Protocol attack	Application-layer attack
Unit of measurement	Bits per second (bps)	Packets per second (pps)	Requests per second (rps)
Used mechanisms / examples	• Reflection and amplification • Shrew attack	• Reflection • State exhaustion • SYN/ACK floods • Fragmentation	• Computational complexity • Hash collisions • Slowloris
Defenses	• Filtering, traffic scrubbing • Black-hole filtering	• Cookies • Rate-limiting	• Randomized/keyed hash functions

Specific Attack Examples

Volumetric Attack: Shrew Attack

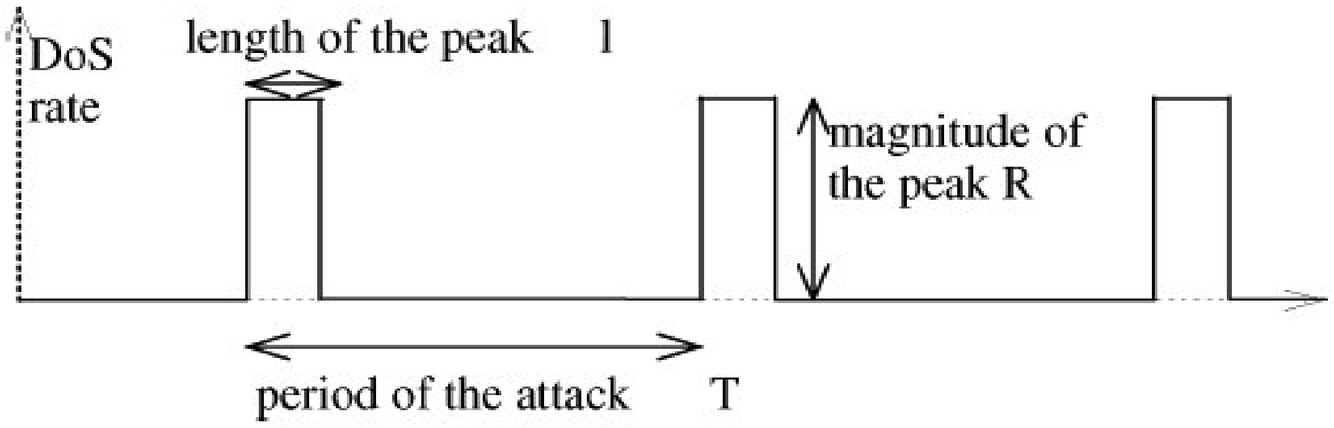
Elephant vs. Shrew

- 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

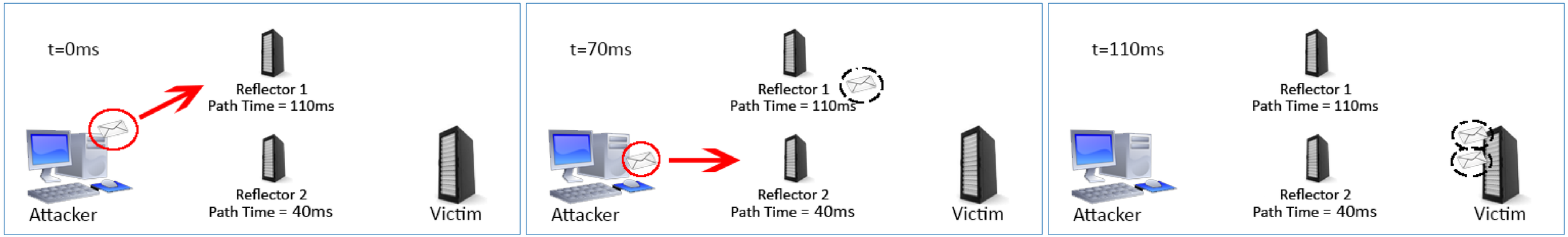
Exponential backoff timeout

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!

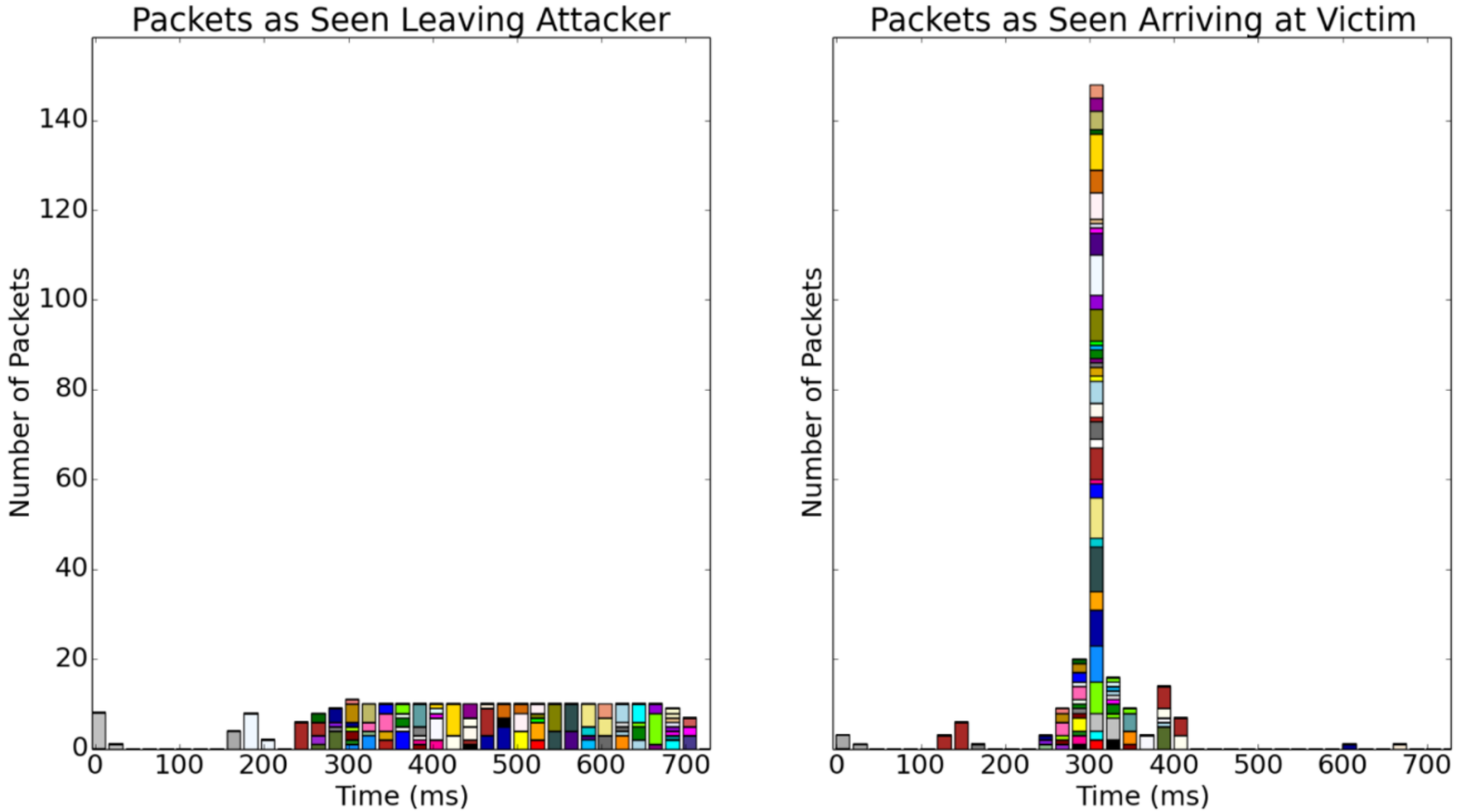
Temporal Lensing

- 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, V. Paxson. “Temporal Lensing and its Application in Pulsing Denial of Service Attacks,” IEEE S&P 2015.

Temporal Lensing

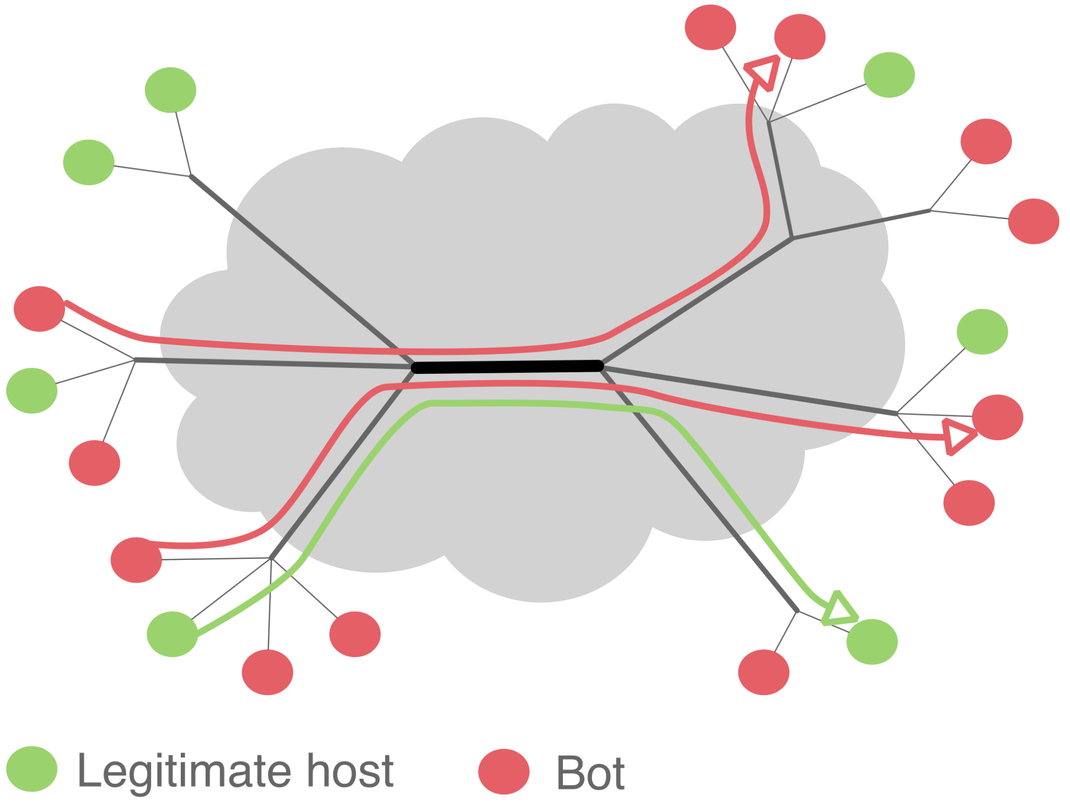


R. Rasti, M. Murthy, V. Paxson. “Temporal Lensing and its Application in Pulsing Denial of Service Attacks,” IEEE S&P 2015.

Volumetric Attack: Coremelt and Crossfire

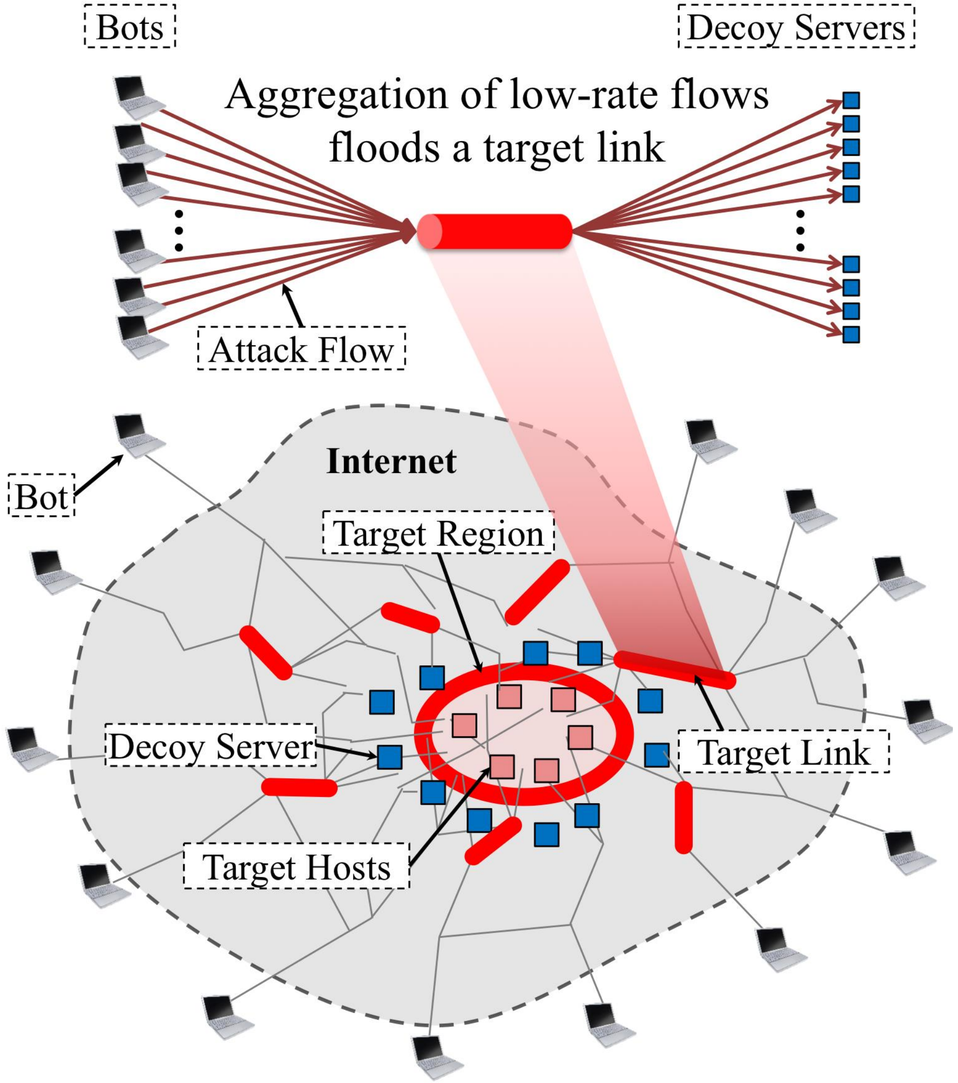
Coremelt Attack

- Adversary controls many bots distributed across the Internet
- Bots send traffic between each other, thus all traffic is desired by destination
  - Traffic is not sent to victim as in regular DDoS attacks
- Adversary can exhaust bandwidth on victim link
- Result: attack traffic exhausts bandwidth in per-flow fair sharing systems



Crossfire Attack [Kang, Lee, Gligor, IEEE S&P 2013]

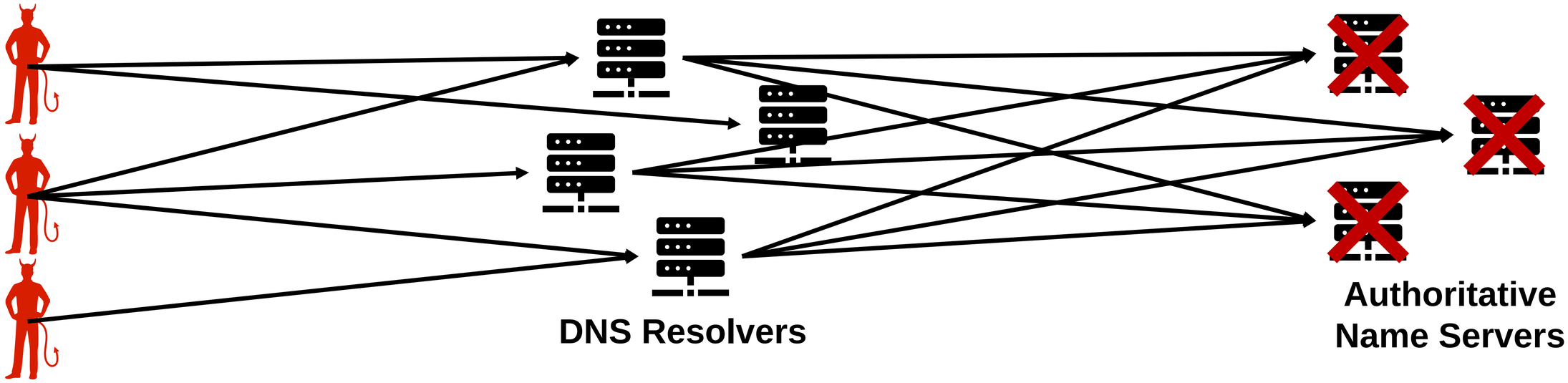
- Adversary controls distributed bot army
- Observation: due to route optimization, few links are actually used to connect a target region to rest of Internet
- Adversary can contact selected servers to overload target links
- Result: disconnect target region from remainder of Internet



Protocol Attack
DNS Flooding

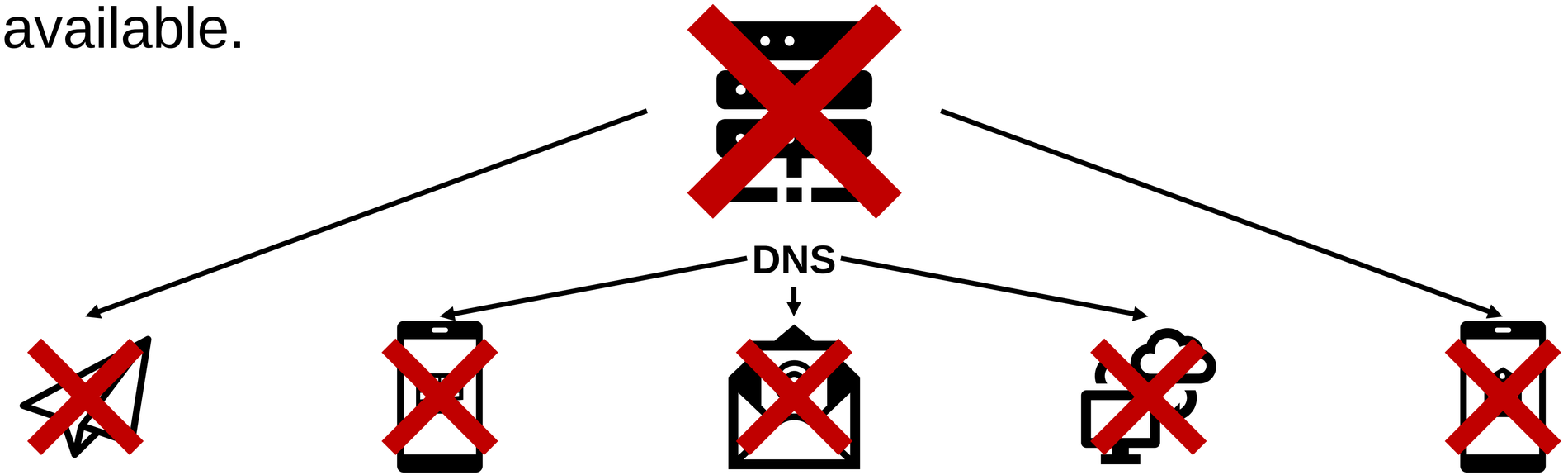
NXDOMAIN Attack

- Goal: overwhelm victim's authoritative name servers
- Idea: query many non-existent subdomains of victim domain
- Resolver queries all authoritative name servers in turn
- Can use multiple DNS resolvers
- Can be sent from distributed botnet and via many different DNS resolvers
- Result: name server can no longer reply to legitimate requests



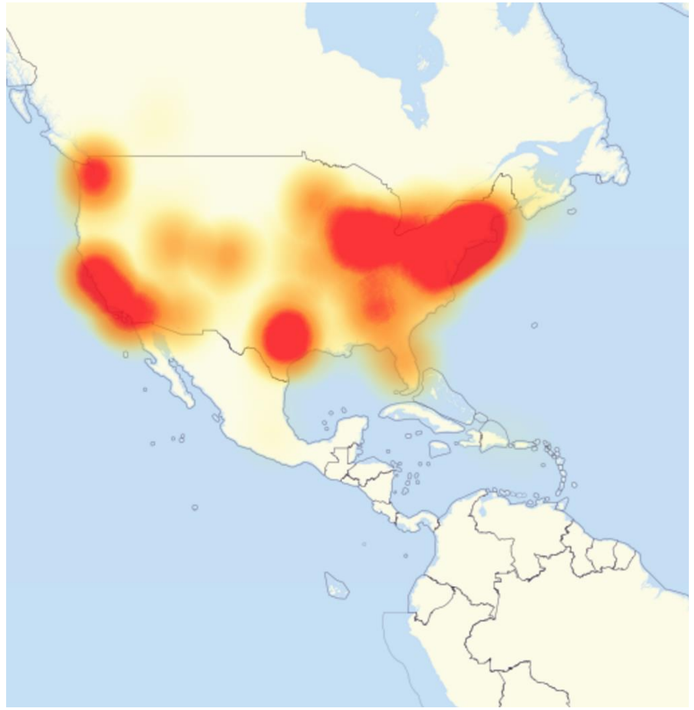
Why is this a problem?

- Most Internet-based services rely on DNS to map domain names to IP addresses.
- If a domain name cannot be resolved, the service does not work.
- → A DoS attack on the DNS system makes many additional systems unavailable.



Example: Attack on Dyn, 2016

- Mirai botnet attacked Dyn, a popular DNS provider
- Based on the NXDomain attack
- Caused widespread outages in the US
  - Three attacks of ~5h duration total
- Many popular websites were unavailable:
  - Including Twitter, Reddit, GitHub, Amazon, Netflix, Spotify



Protocol Attack:
Session State Exhaustion

Session State Exhaustion

Communication Channel

- In a two-way communication, a channel between peers is identified with a unique session number (session state)
- Each channel needs a unique session number
- The session number must be known at the server to match subsequent requests to the right session/channel

Attack

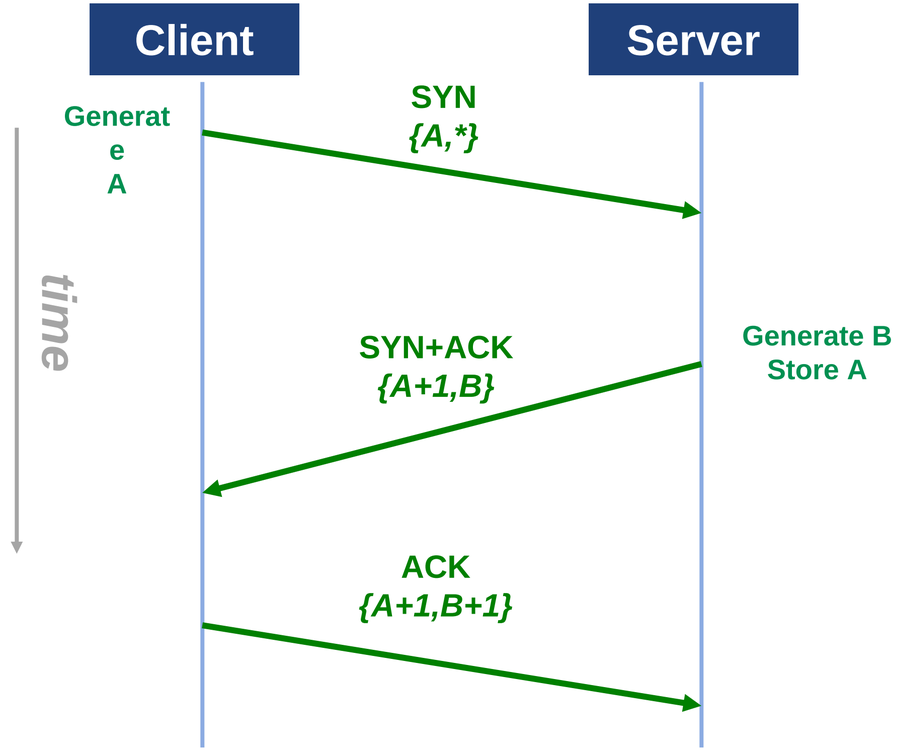
- Exhausting the session state table of the server

Result

- Server can no longer accept new connections
- Existing connections are dropped
- Maybe the server / service crashes

SYN Flood Attack

- TCP uses a three-way handshake to establish a connection



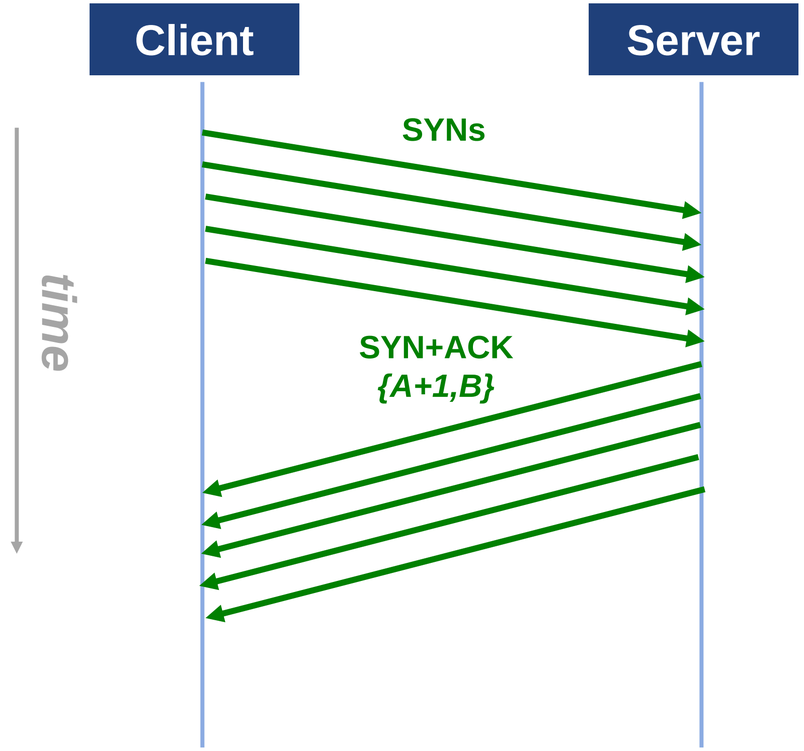
1. Client sends a SYN packet with a random sequence number *A*
2. Server replies with a SYN+ACK packet with acknowledge number *A+1* and a random sequence number *B*
3. Server keeps state *{A+1,B}*

4. Server waits for an ACK with sequence number *A+1* and acknowledge number *B+1*
5. Connection established

SYN Flood Attack

Attack

- Resource starvation of session table at server



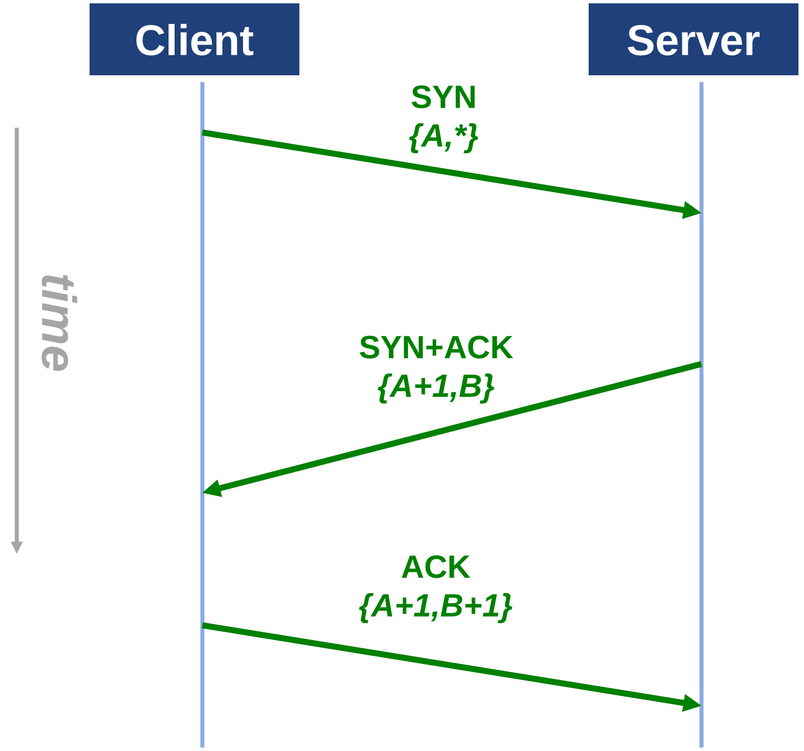
1. SYN flood with spoofed source addresses
2. Server tries to keep state, **eventually state table overflows**
3. Server unable to accept new legit connections

Source: https://en.wikipedia.org/wiki/SYN_flood

SYN Flood Attack

Mitigation - SYN Cookies

- Particular choice of initial TCP sequence number – no state table needed



1. Client sends a SYN packet with a random sequence number *A*
2. Server replies with a SYN+ACK packet with acknowledgement *A+1* and sequence number *B=F(time, IP address, port, ..)*

3. Server checks if expected/valid acknowledge number is returned by applying *F(..)*
4. Connection established

Generic Mitigations

Attack

- The attacker aims to exhaust session state of a protocol/application at server

Countermeasure

- Encode state in a unique but determined way that allows the server to validate the state in the reply
- No need to store session state at the server
- Ensure the encoding cannot be tampered (use crypto-hashes, unique data known to server only)

- Server generates **B = Hash(salt, A)** where **salt** is known to server only and changes over time
- Server only needs to store a few salt values to validate replies

IP Spoofing Defense

Objective: Prevent IP address spoofing, or identify attacker

Ingress Filtering	Gateway device (router, firewall, NAT) drops packets with an “invalid” source IP address field. ▪ Advantages: Eliminates source IP spoofing ▪ Disadvantages: Source-based solution, no deployment incentives, everybody has to deploy
iTrace	One in 20,000 packets “triggers” a router to send a special packet with route information ▪ Advantages: DDoS victim can reconstruct “attack” paths ▪ Disadvantages: extra packets waste bandwidth
Packet Marking	Routers mark 16-bit IP ID field with information that enables reconstruction of IP address ▪ Advantage: No extra overhead ▪ Disadvantage: Probabilistic marking often requires ~1000 packets

Application-Layer Attack: Algorithmic Complexity

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

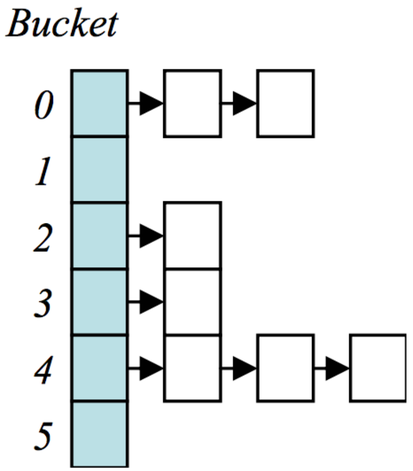


Figure 1: Normal operation of a hash table.

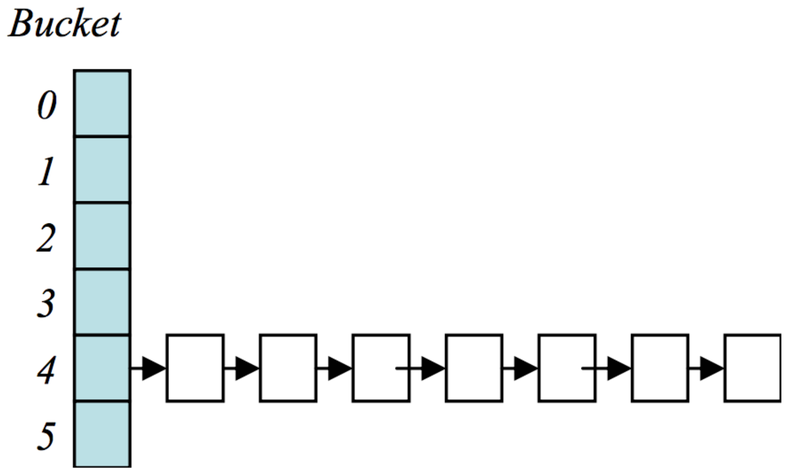


Figure 2: Worst-case hash table collisions.

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 DoS

- Hash table collision in Python

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

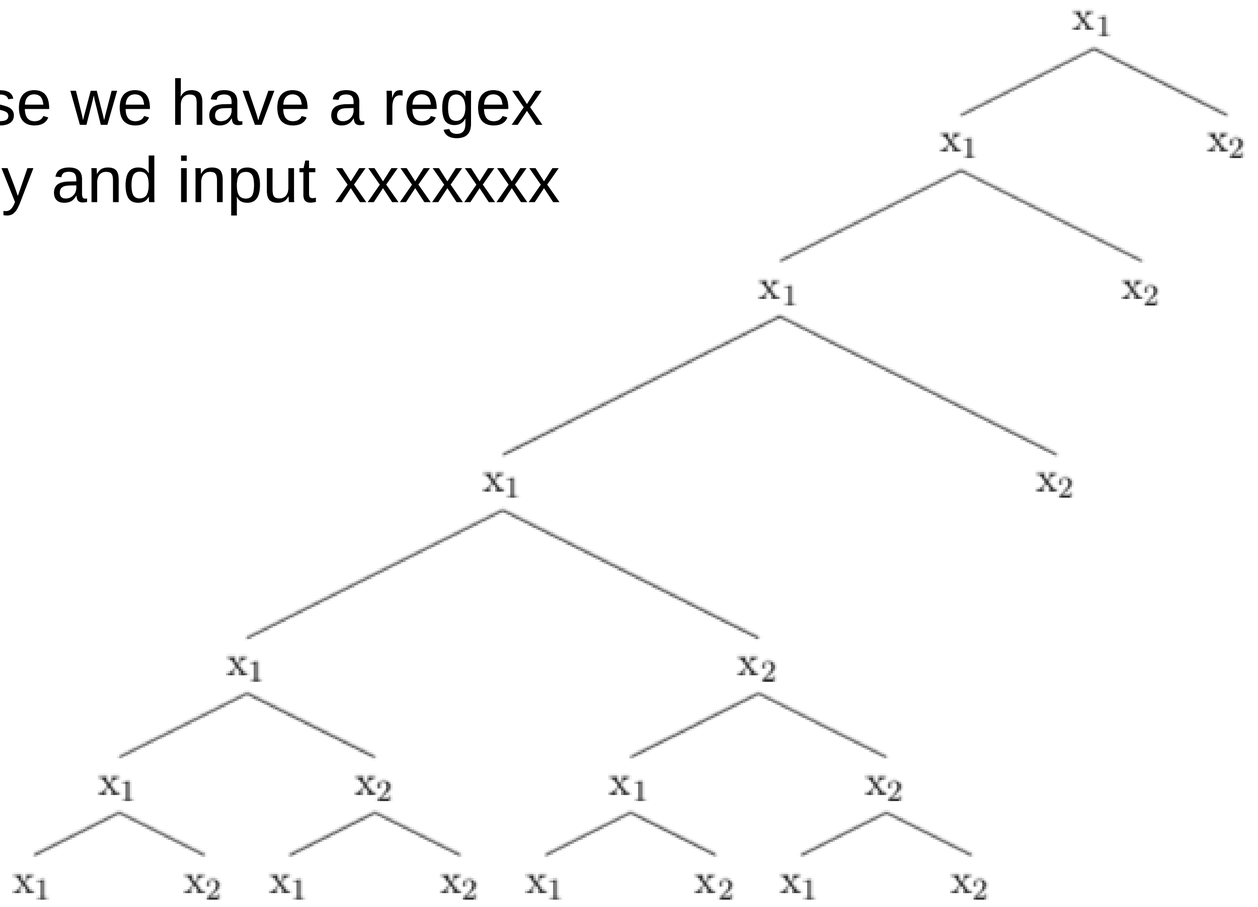
- Similar attacks work in PHP, ASP, Python, Ruby, C++, Java...

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

Application-Layer Attack
Slowloris

Slowloris Attack

Slowloris allows a single machine to take down another machine's web server with minimal bandwidth and side .

- Slowloris tries to keep many connections to the target web server open and hold them open as long as possible
- Slowloris opens connections to the target web server and sends partial requests
- Periodically, it will send subsequent HTTP headers, adding to – but never completing – the request
- Affected servers will keep these connections open, filling their maximum concurrent connection pool, eventually denying additional connection attempts from clients

Slowloris Attack – Mitigation

- Increasing the maximum number of clients the webserver will allow
- Limiting the number of connections from single IP address
- Impose restrictions on the minimum transfer speed per connection
- Restrict the length of time a client is allowed to stay connected

Other mitigating techniques involve setting up reverse proxies, firewalls, load balancers or content switches

Countermeasures

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

How to get 99.999% uptime (<5 min down/year)

Redundancy

- No single point of failure
- N+2 systems running under normal operation (service still available if two systems fail)
- More than two geographically diverse locations
- More than two independent Internet connections

Monitoring & Rapid detection

- Rapid detection and automatic failover
- Evict failed nodes/systems immediately

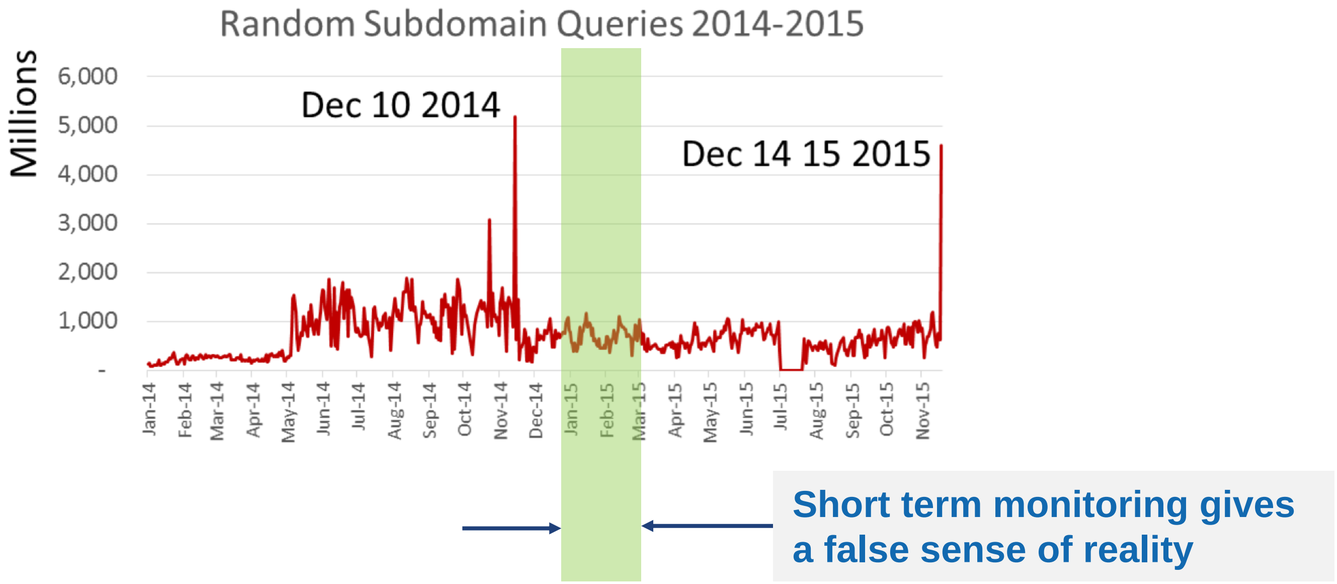
Failure resiliency

- System can tolerate various temporary component failures
- Graceful degradation upon failures

Long Term Monitoring

Perform long term monitoring to assess periodicity and peak periods/loads

- Considering the average load over a period is not sufficient
- Continuously gather data (bandwidth, packets, CPU, transaction)
- Look at the time series

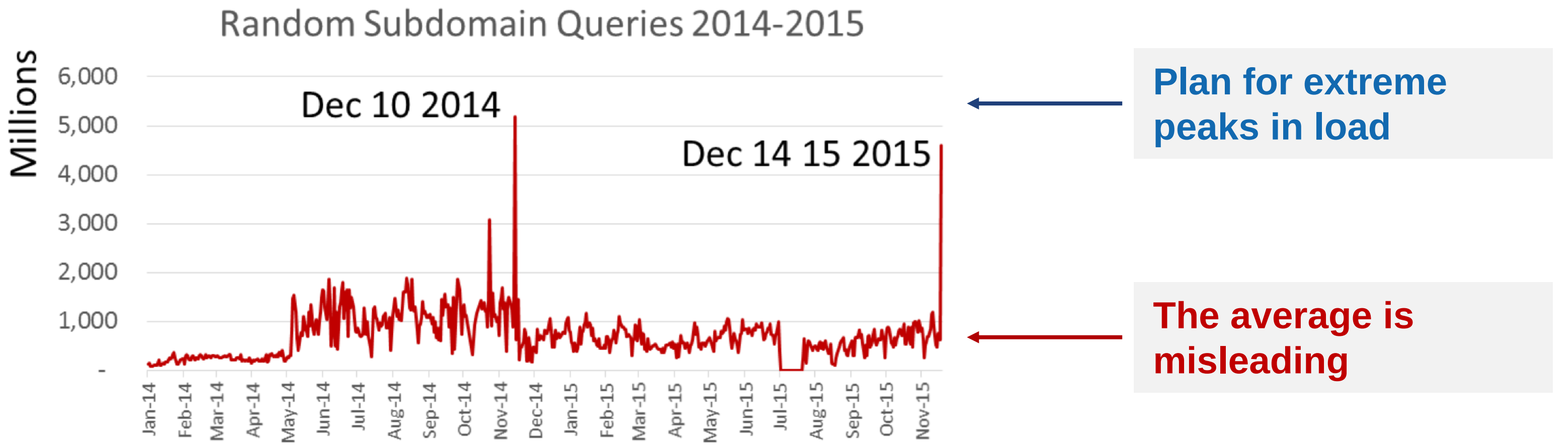


Source: http://nominum.com/huge-spike-in-ddos-activity-for-the-holidays/

Over Provisioning

Plan bandwidth and resources to cover the majority of extreme peak loads

- Consider peak loads under attack / denial of service
- Consider peak loads during high demand periods (Black Friday, promotions, sports events)

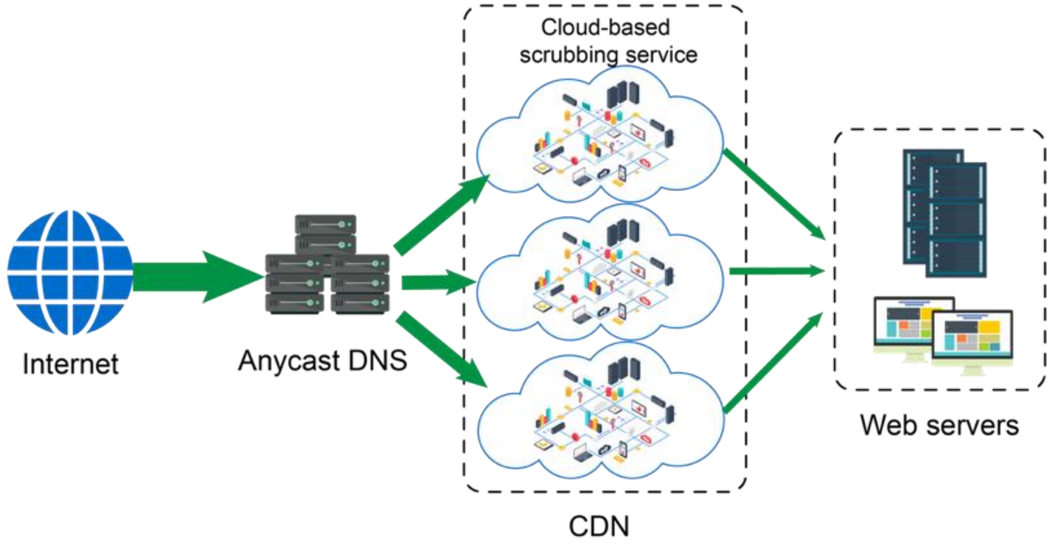


Source: http://nominum.com/huge-spike-in-ddos-activity-for-the-holidays/

In-network and Cloud-based DDoS Mitigation Services

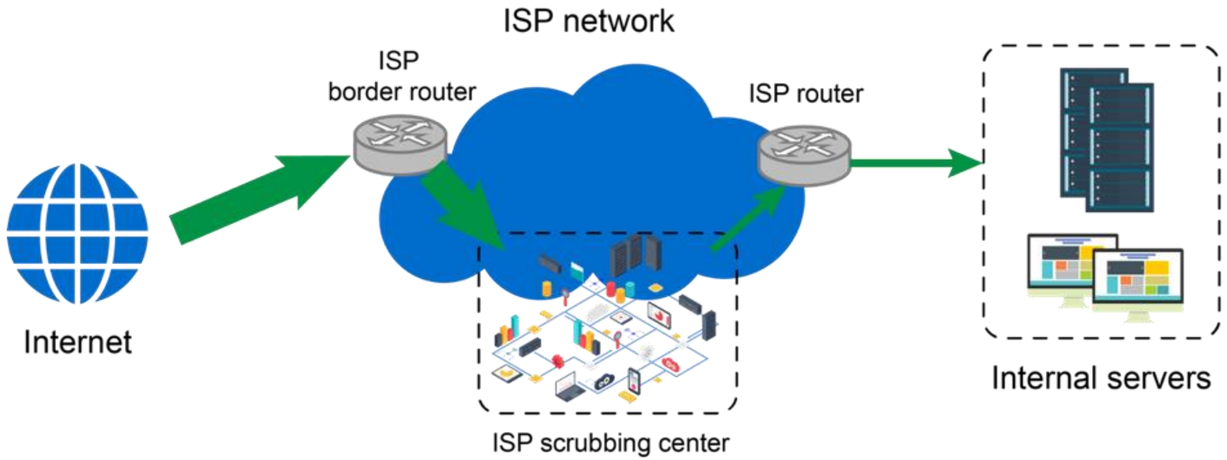
Cloud-based DDoS Mitigation Service

- 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

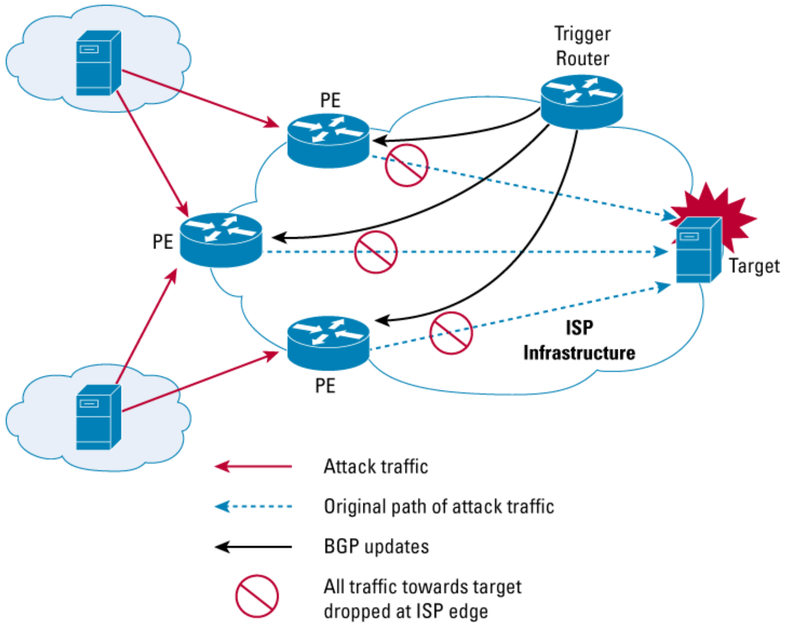


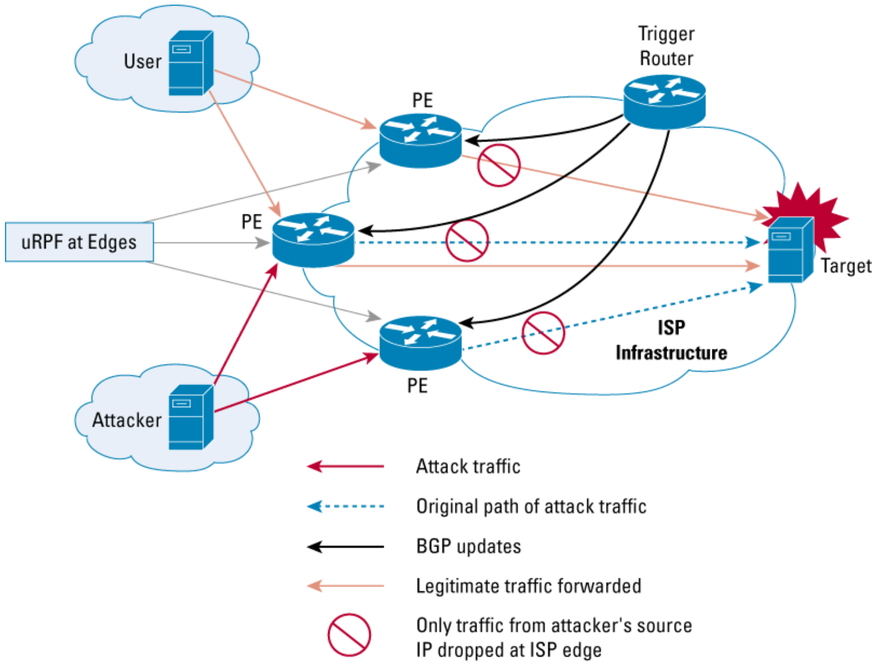
Discussion: Cloud- or ISP-based Filtering

- 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
- Very limited destination traffic control
- High cost for small-/medium-size organizations
- Requires continuous subscription for fetching attack signatures

Remotely Triggered Black Hole (RTBH) Filtering

- Install rules to drop traffic based on source or destination addresses in border routers of an AS → black hole
- Can be achieved through BPG updates





https://www.cisco.com/c/dam/en_us/about/security/intelligence/blackhole.pdf

Summary

State of DDoS

- Numerous attack possibilities: volumetric (network congestion), protocol-level, and application-level attacks
- Current defense mechanisms relatively weak
  - Filtering is very expensive, imperfect (false negatives), and causes collateral damage (false positives)
- Common approach: attack detection and reaction
- Some attacks require novel defense mechanisms (e.g., Coremelt and Crossfire)

Course Evaluation

- The course evaluation is open now (today, 11:30)
- Please follow the link in the email you received
- We always appreciate comments in text form as those help most with improving the course; please use the open questions to provide feedback on all elements of the course:
  - Organization (GitLab, Zoom, emails)
  - Lectures + slides
  - Exercise sheets + sessions
  - Projects
  - Guest lectures
  - Hacking-Lab