Distributed Denial of Service

and information flow, if time

John Mitchell

What are attackers after?

- Steal money
  - Break into e-commerce site, steal credit card numbers, calling card numbers, etc.
- Use computer resources
  - Store “data” on someone’s disk, share with friends
  - Use machine, bandwidth to play game, mount attack
- Damage systems
  - Replace web page of FBI, Stanford Admissions, etc. to embarrass them or as “practical joke”
- Denial of service
  - Keep legitimate user from using resource

How disaster strikes ...

To: nanog@merit.edu
Subject: Yahoo network outage
From: Declan McCullagh <declan@wired.com>
Date: Mon, 07 Feb 2000 16:22:41 -0500
Delivered-To: nanog-outgoing@merit.edu
Sender: owner-nanog@merit.edu

... I was wondering whether anyone has some insight into what happened with Yahoo. The main site (although not all properties) has been offline since 10:30 am pt Monday. It doesn't *appear* to be Global Crossing's problem, though I can't be sure. GC is mum on the phone.
-Declan

To: Declan McCullagh <declan@wired.com>
Subject: Re: Yahoo network outage
From: Richard Irving <rirving@onecall.net>
Date: Mon, 07 Feb 2000 16:34:44 -0500

To Quote my Noc:
I just got off the phone with Global Center NOC. Global Center Sunnyvale Router is down. Both Yahoo! and Global Center are working on the problem at this time. No ETA for repair

---

To: nanog@merit.edu
Subject: Re: Yahoo network outage
From: Kai Schlichting <kai@pac-rim.net>
Date: Mon, 07 Feb 2000 16:37:10 -0500
Delivered-To: nanog-outgoing@merit.edu

Yahoo seems to be down by itself, but GC (The former Exodus?) was majorly hosed for a couple of hours today, at least when seen from UUnet. This has cleared up since. The way it looked, they must have lost a larger circuit and traffic was falling back onto something smaller. I certainly heard about it from customers today.

To: <nanog@merit.edu>
Subject: Yahoo offline because of attack (was: Yahoo network outage)
From: Declan McCullagh <declan@wired.com>
Date: Mon, 07 Feb 2000 20:31:24 -0500

Yahoo told me on the phone that it's a malicious attack, and Global Center says the same thing. In Yahoo's words: "a coordinated distributed denial of service attack." We've got a brief story up at: http://www.wired.com/news/business/0,1367,34178,00.html The problem apparently originated with a router. But what kind of attack could have taken the network offline for that period of time and not affected other Global Center customers? I mean, there had to have been a gaping security hole somewhere: It looks like the routes got lost for (nearly) all of the Yahoo network, but no other non-Yahoo sites...
-Declan

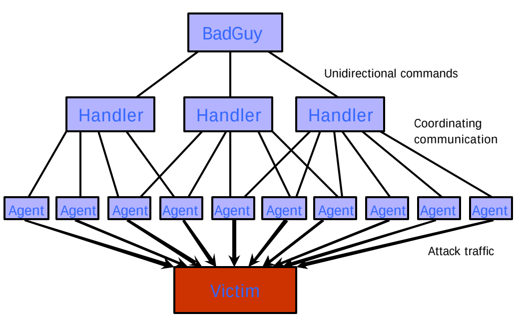
Routers Blamed for Yahoo Outage by Declan McCullagh and Joan

- Most of Yahoo unreachable for three hours on
- Attackers reportedly laid siege...
  - Denying millions of visitors access ...
- An engineer at another company ...
  - told Wired News outage due to misconfigured equipment
- Details remained sketchy...
- A Yahoo spokesperson called it a "coordinated distributed denial of service attack"

What happened?

- Coordinated effort from many sites
- Sites were compromised
  - According to Dittrich's DDoS analysis,
    - trinoo and tfn daemons found on of Solaris 2.x systems
    - systems compromised by exploitation of buffer overrun in the RPC services statd, cmsd and ttdbserverd
- Compromised machines used to mount attack

DDOS



Trin00

- Attacks through UDP flood
- Client to Handler to Agent to Victim
- Multi-master support
- Restarts agents periodically
- Warns of additional connects
- Passwords protect handlers and agents of Trin00 network, though sent in clear text

Tribal Flood Network (TFN)

- Client to Daemon to Victim
- TCP, SYN and UDP floods
- No passwords for client
- Client-Daemon communication only in ICMP
- Needs root access
- Fixed payload size
- Does not authenticate incoming ICMP

Stacheldraht

- Combines Trin00 and TFN features
- Communication is symmetric key encrypted
- Able to upgrade agents on demand
- Client to Handler to Agent to Victim topology, just like Trin00
- Authenticates communication

Traffic Characteristics

- Trinoo
  - Port 1524 tcp   Port 27665 tcp
  - Port 27444 udp   Port 31335 udp
- TFN
  - ICMP ECHO and ICMP ECHO REPLY packets.
- Stacheldraht
  - Port 16660 tcp   Port 65000 tcp
  - ICMP ECHO ICMP ECHO REPLY
- TFN2K
  - Ports supplied at run time or chosen randomly
  - Combination of UDP, ICMP and TCP packets.

Possible firewall actions

- Only allow packets from known hosts
- Check for reverse path
  - Block packets from IP addr X at the firewall if there is no reverse connection going out to addr X
- Ingress/egress filtering
  - Packets in must have outside source destination
  - Packets out must have inside source destination
- Rate limiting
  - Limit rate of ICMP packets and/or SYN packets

All of these steps may interfere with legitimate traffic

Can you find source of attack?

- Hard to find BadGuy
  - Originator of attack compromised the handlers
  - Originator not active when DDOS attack occurs
- Can try to find agents
  - Source IP address in packets is not reliable
  - Need to examine traffic at many points, modify traffic, or modify routers

Methods for finding agents

- Manual methods using current IP routing
  - Link testing
  - Input debugging
  - Controlled flooding
  - Logging
- Changing router software
  - Instrument routers to store path
  - Provides automated IP traceback

Link Testing

- Start from victim and test upstream links
- Recursively repeat until source is located
- Assume attack remains active until trace complete

Input Debugging

- Victim recognize attack signature
- Install filter on upstream router
- Pros
  - May use software to help coordinate
- Cons
  - Require cooperation between ISPs
  - Considerable management overhead

Controlled Flooding

- Flooding link during attack
  - Add large bursts of traffic
  - Observe change in packet rate at victim
- Pros
  - Eventually works if attack continues
- Cons
  - Add denial of service to denial of service

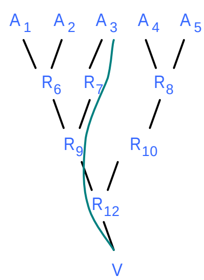
Logging

- Key routers log packets
- Use data mining to find path
- Pros
  - Post mortem – works after attack stops
- Cons
  - High resource demand

Modify routers to allow IP traceback

Traceback problem

- Goal
  - Given set of packets
  - Determine path
- Assumptions
  - Most routers remain uncompromised
  - Attacker sends many packets
  - Route from attacker to victim remains relatively stable

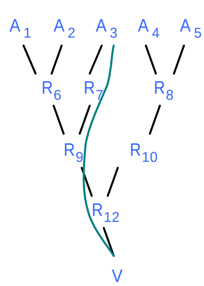


Simple method

- Record path
  - Each router adds IP address to packet
  - Victim reads path from packet
- Problem
  - Requires space in packet
    - Path can be long
    - No extra fields in current IP format
      - Changes to packet format are not practical

Better idea

- Many packets
  - DDoS involves many packets on same path
- Store one link in each packet
  - Each router probabilistically stores own address
  - Fixed space regardless of path length



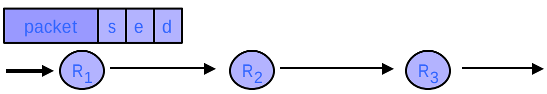
Edge Sampling

- Data fields
  - Edge: *start* and *end* IP addresses
  - Distance: number of hops since edge stored
- Marking procedure for router R

```
with probability p
        write R into start address
        write 0 into distance field
else
        if distance ==0 write R into end field
        increment distance field
```

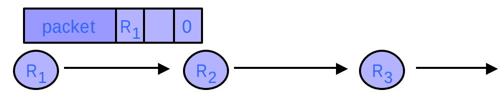
Edge Sampling: picture

- Packet received
  - R_1 receives packet from source or another router
  - Packet contains space for start, end, distance



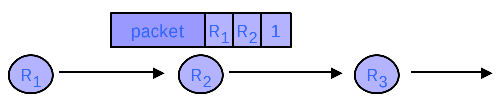
Edge Sampling: picture

- Begin writing edge
  - R_1 chooses to write start of edge
  - Sets distance to 0



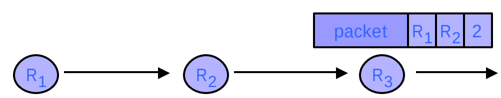
Edge Sampling

- Finish writing edge
  - R_2 chooses not to overwrite edge
  - Distance is 0
    - Write end of edge, increment distance to 1



Edge Sampling

- Increment distance
  - R_3 chooses not to overwrite edge
  - Distance >0
    - Increment distance to 2



Path reconstruction

- Extract identifiers from attack packets
- Build graph rooted at victim
  - Each (start,end,distance) tuple is an edge
  - Eliminate edges with inconsistent distance
  - Traverse edges from root to find attack paths
- # packets needed to reconstruct path

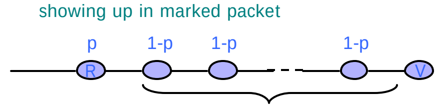
$$E(X) < \frac{\ln(d)}{p(1-p)^{d-1}}$$

where p is marking probability, d is length of path

Optimal p is 1/d … can vary probability by distance

Node Sampling?

- Less data than edge sampling
  - Each router writes own address with probability p
- Infer order by number of packets
  - Router at distance d has probability $p(1-p)^d$ of showing up in marked packet

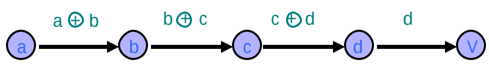


- Problems
  - Need many packets to infer path order
  - Does not work well if many paths

Reduce Space Requirement

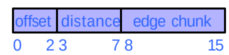
- XOR edge IP addresses
  - Store edge as start ⊕ end
  - Work backwards to get path:

    (start ⊕ end) ⊕ end = start
- Sample attack path

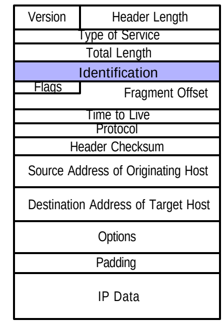


Details: where to store edge

- Identification field
  - Used for fragmentation
  - Fragmentation is rare
  - 16 bits
- Store edge in 16 bits?

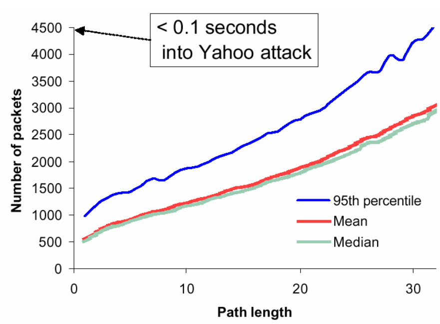


  - Break into chunks
  - Store start ⊕ end



[Savage et al]

Experimental convergence time



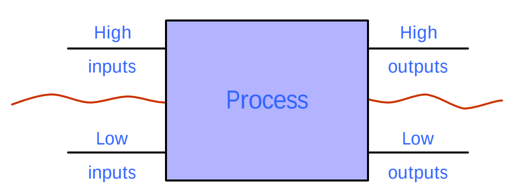
Summary of Edge Sampling

- Benefits
  - Practical algorithm for tracing anonymous attacks
  - Can reduce per-packet space overhead (at a cost)
  - Potential encoding into current IP packet header
- Weaknesses
  - Path validation/authentication
  - Robustness in highly distributed attacks
    - Both addressed nicely in [Song&Perrig00]
  - Compatibility issues (IPsec AH, IPv6)
  - Origin laundering (reflectors, tunnels, etc)

Limitations of Secure OS

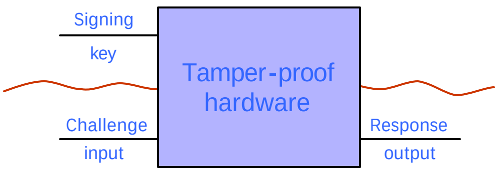
- Noninterference
  - Actions by high-level users (secret, top secret) should not be observable by low -level users (unclassified, ...)
  - Difficult to achieve and prove, not impossible
- Covert Channels
  - Can user of system deliberately communicate secret information to external collaborator?

Noninterference



Given program, can you determine information flow?

Example: Smart Card



Information flow example

◆ First guess
- Mark expressions as high or low
  - Some resemblance to Perl tainting
- Check assignment for high value in low location

◆ But

if $(x_{high} > 0)\ y_{low} = 0;$
else $y_{low} = 1;$

◆ Also
- This will never work unless programmer keeps high, low variables separate

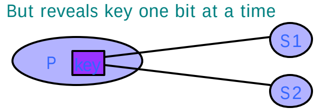
Covert Channels

◆ Butler Lampson
- Difficulty achieving confinement (paper on web)
- Communicate by using CPU, locking/unlocking file, sending/delaying msg, …

◆ Gustavus Simmons
- Cryptographic techniques make it impossible to detect presence of a covert channel

Example

◆ The Two-Server Trojan Horse: [McLean]
- Device P can chose from two Key Servers
- P is expected to choose randomly, to balance load
- But reveals key one bit at a time



◆ Observations
- Information flow easily detected by noninterference analysis of the algorithm
- More subtle if choice based on random seed known to external attacker