6.033 Spring 2019

Lecture #23

• Combating network adversaries
  • DDoS attacks
  • Intrusion Detection
Last time

attacker’s goal
observe or tamper with packets
This time

- **principal**: (identifies client on server)

- **request**

- **server**
  - **guard**
  - **resource**

**attacker’s goal**
prevent legitimate access to an Internet resource

**method: DDoS attacks**
congest the service enough to make it unavailable
‘Denial of service condition’ disrupted US energy company operations

Zack Whittaker  @zackwhittaker  /  4 days ago

Comment
**botnets:** large collections of compromised machines controlled by an attacker. make DDoS attacks *much* easier to mount

**XSS:** if this script is executed on a victim’s machine, the attacker will get the victim’s cookie
**botnets:** large collections of compromised machines controlled by an attacker. make DDoS attacks *much* easier to mount

C&C server

example command: `dos <IP>`

compromised machines

(\(\sim 100,000\) of them)
network intrusion detection systems (NIDS):

attempt to detect network attacks so that users can then prevent them (detection is the first step to prevention)
alert tcp $EXTERNAL_NET any -> $HOME_NET 7597 (msg:"MALWARE-BACKDOOR QAZ Worm Client Login access"; flow:to_server,established; content:"qazwsx.hsq"; metadata:ruleset community; reference:mcafee,98775; classtype:misc-activity; sid:108; rev:11;)

6.033 | spring 2019 | lacurts@mit.edu
network intrusion detection systems (NIDS): attempt to detect network attacks so that users can then prevent them (detection is the first step to prevention)
for each packet:
    search packet for "root"

problem: string might be split across packets
stream = []
for each packet:
    add packet data to stream
search stream for “root”

problem: packets might arrive out of order
stream = []
for each packet:
    get sequence number
    add to stream in the correct order
    search stream for “root”

**problem:** this is more difficult than it looks on the slide, and requires keeping a lot of state

**problem 2:** it doesn’t even work
attacker \[ \text{15 hops} \rightarrow \text{NIDS} \rightarrow \text{5 hops} \rightarrow \text{receiver} \]

seq=1: [ r ] or [ n ]

[ r ] TTL=23 seq=1

[ n ] TTL=17 seq=1

received by NIDS, not by receiver, because of TTL
additional challenge:
some DDoS attacks mimic legitimate traffic
GET largeFile.zip → victim’s webserver
GET largeFile.zip

victim's webserver
GET largeFile.zip

DO bigQuery

victim's webserver
Victim's webserver

GET largeFile.zip

DO bigQuery
GET largeFile.zip

DO bigQuery

victim's webserver
TCP handshake

1. Client sends a SYN packet.
2. Server sends a SYN-ACK packet.
3. Client sends an ACK packet.
4. Server stores the state.

Connected!
SYN

store state

SYN-ACK
store state
Normal ACKs

seq=1

seq=2
seq=3

seq=4
seq=5
seq=6
seq=7

ack=1

ack=2
ack=3
“Optimistic” ACKs

seq=1

ack=1
“Optimistic” ACKs

seq=1
seq=2
seq=3

ack=1
ack=2
ack=3
“Optimistic” ACKs

seq=1 → ack=1

seq=2 → ack=2
seq=3 → ack=3

seq=4
seq=5
seq=6
seq=7
"Optimistic" ACKs

victim will quickly saturate its own links, in some sense DoSing itself
DNS request: src=1.2.3.4

DNS nameservers
(preferably DNSSEC-enabled)

DNS response: dst=1.2.3.4

DDoS traffic doesn’t even come from attacker-owned machines!

victim’s IP: 1.2.3.4
attackers can also mount attacks by controlling routers
• **DDoS** attacks prevent legitimate access to Internet services. Secure channels won’t help us here. **Botnets** make DDoS attacks very practical to mount.

• DDoS attacks are difficult to prevent because they are **difficult to detect**. Signature-matching and anomaly-detection help, but have their own challenges, and are sometimes evadable. Moreover, **DDoS traffic can mimic legitimate traffic**.

• Network attacks are particularly devastating when parts of the **network infrastructure** are attacked (e.g., DDoSing the DNS root zone, making fake BGP announcements).