6.033 Spring 2018
Lecture #23

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

principal

(request identifies client on server)

attacker’s goal

observe or tamper with packets
This time

**principal** (identifies client on server)

**server**

**guard**

**resource**

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

**method: DDoS attacks**
congest the service enough to make it unavailable
**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

```
dos <IP>
```

![Diagram showing botnet communication]

compromised machines

(~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;)

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 \(\cdots\) 15 hops \(\cdots\) NIDS \(\cdots\) 5 hops \(\rightarrow\) receiver

\[
\begin{align*}
\text{seq}=1: & [ r ] \text{ or } [ n ] \\
[ r ] \text{ TTL}=23 \text{ seq}=1 \\
[ n ] \text{ TTL}=17 \text{ seq}=1
\end{align*}
\]

received by NIDS, not by receiver, because of TTL
At the attacker end, the sequences are:

- seq=1: [ r ] or [ n ]
- seq=2: [ o ] or [ i ]

At the receiver end, the sequences are:

- seq=1: [ r ]
- seq=2: [ o ]

The packets with TTL values are:

- [ r ] TTL=23  seq=1
- [ n ] TTL=17  seq=1
- [ o ] TTL=21  seq=2
- [ i ] TTL=15  seq=2

X marks the drops for TTL values.
Attacker \[ \rightarrow \] NIDS \[ \rightarrow \] Receiver

\begin{align*}
\text{seq}=1: & \quad [ \ r \ ] \text{ or } [ \ n \ ] & \quad \text{seq}=1: & \quad [ \ r \ ] \\
\text{seq}=2: & \quad [ \ o \ ] \text{ or } [ \ i \ ] & \quad \text{seq}=2: & \quad [ \ o \ ] \\
\text{seq}=3: & \quad [ \ o \ ] & \quad \text{seq}=3: & \quad [ \ o \ ] \\
\text{seq}=4: & \quad [ \ c \ ] \text{ or } [ \ t \ ] & \quad \text{seq}=4: & \quad [ \ t \ ]
\end{align*}
additional challenge:
some DDoS attacks mimic legitimate traffic
GET largeFile.zip

victim’s webserver
GET largeFile.zip

DO bigQuery

victim's webserver
GET largeFile.zip

DO bigQuery

victim’s webserver
GET largeFile.zip

DO bigQuery

victim’s webserver
TCP handshake

SYN

ACK

SYN-ACK

store state

connected!
SYN

SYN

store state
store state
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
seq=2
seq=3
seq=4
seq=5
seq=6
seq=7

ack=1
ack=2
ack=3
“Optimistic” 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

seq=2
seq=3
ack=2
ack=3

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

victim will quickly saturate its own links, in some sense DoSimg 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).