it’s everyone’s favorite time of the year!

http://registrar.mit.edu/subjectevaluation
Colonial Pipeline hackers apologize, promise to ransom less controversial targets in future

‘We are apolitical, we do not participate in geopolitics’

By Mitchell Clark | May 10, 2021, 2:44pm EDT

how does an adversary — or anyone? — decide what systems are “controversial targets”? 
Lecture #24: Network-based attacks
preventing access — denying service — to online resources
this week, our adversaries are still on the network, but they have new goals

adversary’s goal: prevent legitimate access to an internet resource
this week, our adversaries are still on the network, but they have new goals

the primary method they’ll use to achieve this goal is a DDoS attack, made more effective with a botnet
this week, our adversaries are still on the network, but they have new goals

the primary method they’ll use to achieve this goal is a DDoS attack, made more effective with a botnet

‘Denial of service condition’ disrupted US energy company operations

Zack Whittaker @zackwhittaker / 4 days ago
**Policy:** Maintain **availability** of the service

**Threat Model:** Adversary controls a **botnet**, and is aiming to prevent access to a legitimate service via **DDoS attacks**

**Botnets:** Large collections of compromised machines controlled by an adversary

The Mirai paper calls these “C2 servers” instead of C&C servers.

Example command: **dos <IP>**

policy: maintain **availability** of the service

threat model: adversary controls a **botnet**, and is aiming to prevent access to a legitimate service via **DDoS attacks**

**network intrusion detection systems:**

attempt to detect network attacks so that users can then prevent them (detection is the first step to prevention)

botnets are sophisticated, so we can't rely on just blocking “bad” IP addresses

**signature-based** NIDS match traffic against known signatures

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;)

an example of a signature
network intrusion detection systems:

attempt to detect network attacks so that users can then prevent them (detection is the first step to prevention)

botnets are sophisticated, so we can't rely on just blocking “bad” IP addresses

signature-based NIDS match traffic against known signatures

anomaly-based NIDS match traffic against a model of “normal” traffic

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;)

an example of a signature

policy: maintain availability of the service

threat model: adversary controls a botnet, and is aiming to prevent access to a legitimate service via DDoS attacks
network intrusion detection systems:

- **signature-based** NIDS match traffic against known signatures
- **anomaly-based** NIDS match traffic against a model of “normal” traffic

**threat model:** adversary controls a **botnet**, and is aiming to prevent access to a legitimate service via **DDoS attacks**

**policy:** maintain **availability** of the service

**problem:** string might be split across packets

**for each packet:**

- search packet for “root”
**policy:** maintain **availability** of the service

**threat model:** adversary controls a **botnet**, and is aiming to prevent access to a legitimate service via **DDoS attacks**

---

**network intrusion detection systems:**

attempt to detect network attacks so that users can then prevent them (detection is the first step to prevention)

botnets are sophisticated, so we can't rely on just blocking “bad” IP addresses

**signature-based** NIDS match traffic against known signatures

**anomaly-based** NIDS match traffic against a model of “normal” traffic

---

**stream = []**

**for each packet:**

- add packet data to stream
- search stream for “root”

**problem:** packets might arrive out of order

---

threat model: adversary controls a botnet, and is aiming to prevent access to a legitimate service via DDoS attacks
network intrusion detection systems:

attempt to detect network attacks so that users can then prevent them (detection is the first step to prevention)

botnets are sophisticated, so we can’t rely on just blocking “bad” IP addresses

signature-based NIDS match traffic against known signatures

anomaly-based NIDS match traffic against a model of “normal” traffic

stream = []
for each packet:
  get sequence number
  add to stream in the correct order
  search stream for “root”

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

problem 2: it doesn’t even work

policy: maintain availability of the service

threat model: adversary controls a botnet, and is aiming to prevent access to a legitimate service via DDoS attacks
policy: maintain **availability** of the service

threat model: adversary controls a **botnet**, and is aiming to prevent access to a legitimate service via **DDoS attacks**

**Network intrusion detection systems:**

attempt to detect network attacks so that users can then prevent them (detection is the first step to prevention)

botnets are sophisticated, so we can’t rely on just blocking “bad” IP addresses

**Signature-based** NIDS match traffic against known signatures

**Anomaly-based** NIDS match traffic against a model of “normal” traffic

---

received by NIDS, not by receiver, because of TTL
policy: maintain availability of the service

threat model: adversary controls a botnet, and is aiming to prevent access to a legitimate service via DDoS attacks

network intrusion detection systems:

attempt to detect network attacks so that users can then prevent them (detection is the first step to prevention)

botnets are sophisticated, so we can’t rely on just blocking “bad” IP addresses

signature-based NIDS match traffic against known signatures

anomaly-based NIDS match traffic against a model of “normal” traffic
**Policy:** maintain **availability** of the service

**Threat Model:** adversary controls a **botnet**, and is aiming to prevent access to a legitimate service via **DDoS attacks**

**Additional Challenge:** some DDoS attacks mimic legitimate traffic, and/or attempt to exhaust resources on the server itself

This attack is similar to the HTTP flood attack in the Mirai paper
**policy:** maintain availability of the service

**threat model:** adversary controls a botnet, and is aiming to prevent access to a legitimate service via DDoS attacks

**additional challenge:** some DDoS attacks mimic legitimate traffic, and/or attempt to exhaust resources on the server itself
policy: maintain availability of the service
threat model: adversary controls a botnet, and is aiming to prevent access to a legitimate service via DDoS attacks

additional challenge: some DDoS attacks mimic legitimate traffic, and/or attempt to exhaust resources on the server itself

this attack is similar to the SYN flood attack in the Mirai paper
policy: maintain availability of the service

threat model: adversary controls a botnet, and is aiming to prevent access to a legitimate service via DDoS attacks

additional challenge:
some DDoS attacks mimic legitimate traffic, and/or attempt to exhaust resources on the server itself
**policy:** maintain **availability** of the service

**threat model:** adversary controls a **botnet**, and is aiming to prevent access to a legitimate service via **DDoS attacks**

**additional challenge:**

some DDoS attacks mimic legitimate traffic, and/or attempt to exhaust resources on the server itself

victim will quickly saturate its own links, in some sense DoSing itself
**policy:** maintain *availability* of the service

**threat model:** adversary controls a *botnet*, and is aiming to prevent access to a legitimate service via **DDoS attacks**

**additional challenge:**
some DDoS attacks mimic legitimate traffic, and/or attempt to exhaust resources on the server itself

---

![Diagram](diagram.png)

- **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`

*This is a DNS amplification attack; it is *not* the “DNS flood” attack mentioned in the Mirai paper.*
**DDoS attacks** prevent legitimate access to internet services. Secure channels won’t help us here, and **botnets** make DDoS attacks relatively easy to mount.

DDoS attacks are difficult to prevent because they are sophisticated and can mimic legitimate traffic; **network-intrusion detection systems** help, but they’re not perfect.

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