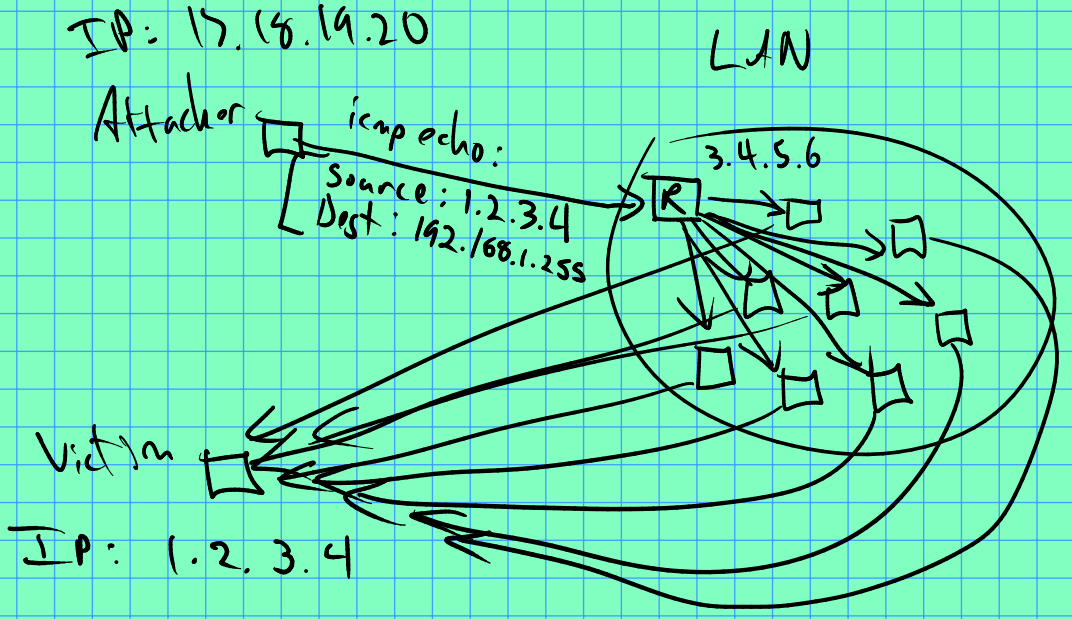
Network security part II : Denial of Service.

History (≈ 199x) "Smurf" attacks



Observations:

- Spoofed IP address
- Amplification
- Stateless/connectionless protocol (ICMP)

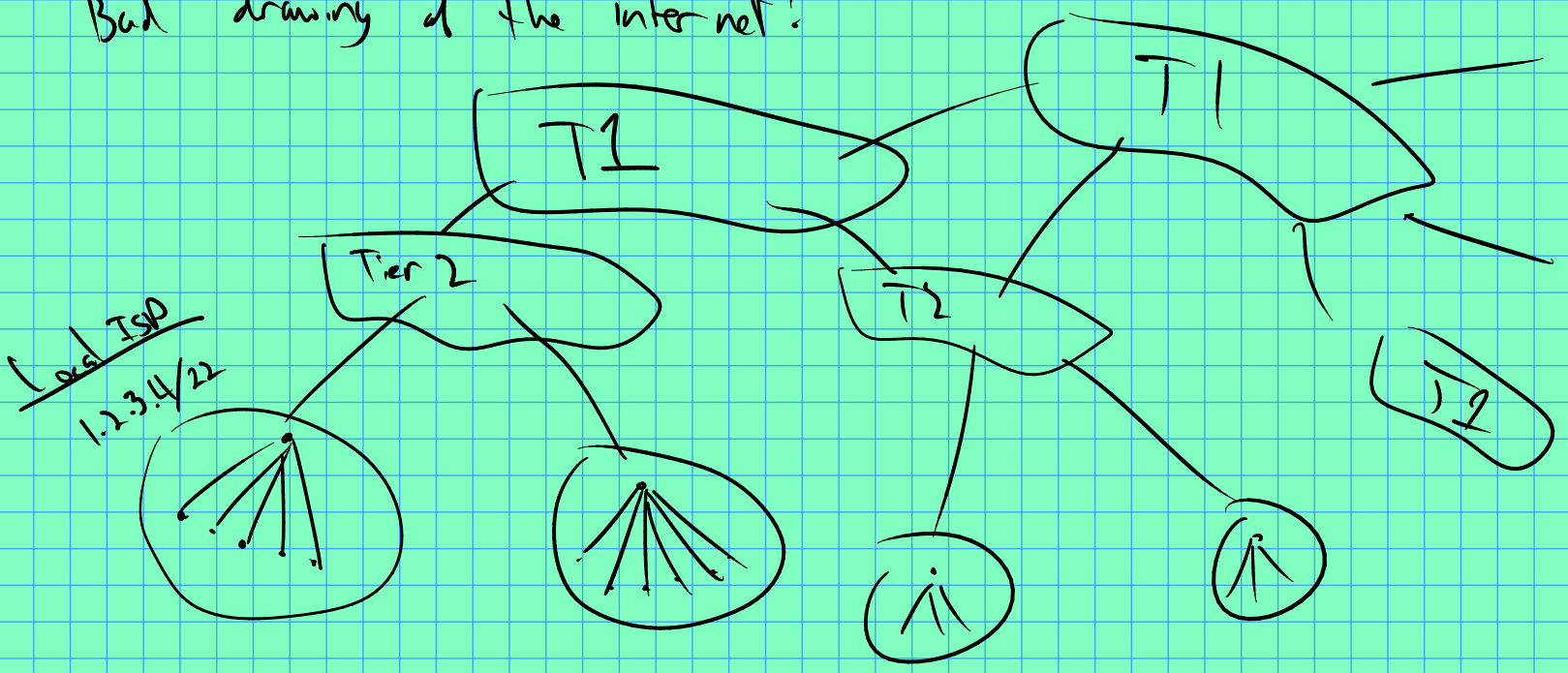
Other ancient attacks: SYN flooding

Start a TCP handshake, but spoof the source to be unreachable!

(More reading: Best Common Practice 38 (BCP38))

IP Spoofing: How? Why can't we stop it?

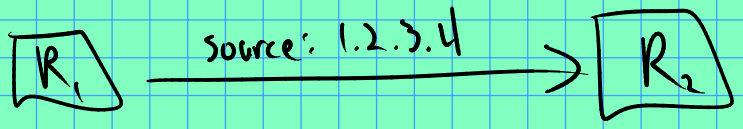
Bad drawing of the internet:



(Reading: look up "multihomed network")

filtering can be done @ local level ("last mile")
but difficult higher up (T1, T2)

Idea: look up reverse route and only accept if source is on it?



R_2 accepts/forwards if best route to 1.2.3.4 would be via R_1

Problem: in general, routing not symmetric... :(

More modern attacks: reflection + amplification.

Attack vector: DNS.

Idea: - spoof IP source addr.
- send query with large response ⊗
- profit.

Note: DNS uses UDP...

Note: likely need multiple DNS servers for a successful attack, and unfortunately there are many "open" DNS servers out there (≥10k?).

Spread attack across all available servers.

⊛ amplification factor: ≥ 42×

Other notable attack vectors: memcached

Used to be configured to speak UDP...

Small query could result in huge response

(15 byte query ⟶ 134 KB response !!)

(Reading: see attack on github ≈ 2018?)

NTP (network time protocol)

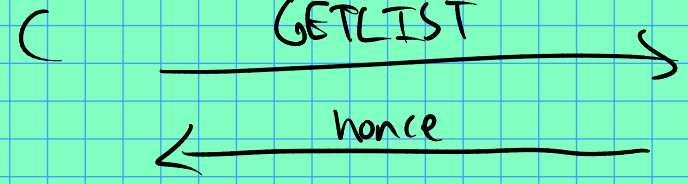
A few commands produced large responses on small queries, e.g. MON_GETLIST

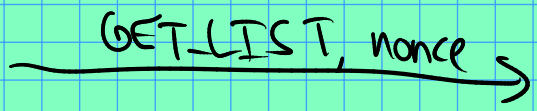
(would return list of up to 600 computers the server recently talked with)

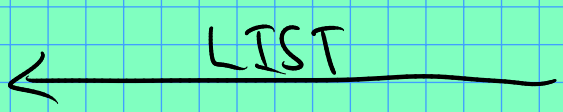
Mitigation?

Design goals: - keep MON_GETLIST (or similar)
- still use UDP
- Do not add per-client state

Before answering such queries, make sure source IP address is accessible to party asking the query. For this, generate a "nonce" for each asking client, and require that they know it before responding







send list only if nonce is valid!

(knowledge of nonce should preclude IP spoofing)

To check nonce, just recompute

$n' = H(IP(c) \| r \| \text{time range})$

return LIST if $n = n'$ & now ∈ time range

Exam notes:

#5 how to re-share a secret w/o reconstructing it?

Say P_i has $s_i = f(i)$ for $i = 1, \ldots, n$

and where $f = s + \sum_{j=1}^{t} c_j x^j$, $c_j \in_R \mathbb{Z}_p$

and $s \in \mathbb{Z}_p$.

Idea: each P_i produces shares of $\underline{0}$ and distributes to each other player

Key observation: $f, g \in \mathbb{Z}_p[X]$ then

$$(f+g)(i) = f(i) + g(i).$$

P_i: chose $c_k^i \in_R \mathbb{Z}_p$ for $k = 1, \ldots, t$.

Set $f_i(X) = \sum_{k=1}^{t} c_k^i X^k$.

P_i sends to P_j $f_i(j)$.

All players $1, \ldots n$ do the same.

Now each player i has $f_j(i)$ for $j = 1, \ldots, n$

Now compute new shares of s as

$$s_i' = s_i + \sum_{j=1}^{n} f_j(i).$$

s_i: original share, $= f(i)$

define $f' = f + \sum_{j=1}^{n} f_j \in \mathbb{Z}_p[X]$.

Then note that $f'(0) = f(0) + \sum_{j=1}^{n} f_j(0) = f(0) = s$.

($f_j(0) = 0$.)

Furthermore, as long as even

one player P_i created a random polynomial (of degree t)

then f' is distributed uniformly subject to $f'(0) = s$. (Same as if we ran Shamir "from scratch")

Further reading: "Proactive security" by R. Gennaro et. al
Also similar to distributed key generation for ElGamal (Pedersen ??)

#9 $y = x^2$ $\mathcal{P}$ knows x. y is public.
all values $\in \mathbb{Z}_n^*$, $n = pq$.

$\mathcal{P}$ $\xrightarrow{h = r^2}$ $\mathcal{V}$
$\mathcal{V}$ $\xleftarrow{b \in_R \{0,1\}}$ $\mathcal{P}$
$\mathcal{P}$ $\xrightarrow{z = r x^b}$ $\mathcal{V}$

How to simulate such transcripts knowing only y, n?
Easy if $b = 0$: $\mathcal{V}$ chooses own $r \in_R \mathbb{Z}_n^*$,
set $h = r^2$, $z = r$.
What about $b = 1$?
need $z^2 = hy$. So... $h = z^2 y^{-1}$.

More precisely: choose b first. If $b = 1$, chose $z \in_R \mathbb{Z}_n^*$ and set $h = z^2 y^{-1}$.
Else just choose $r \in_R \mathbb{Z}_n^*$, set $h = r^2$ and $z = r$.