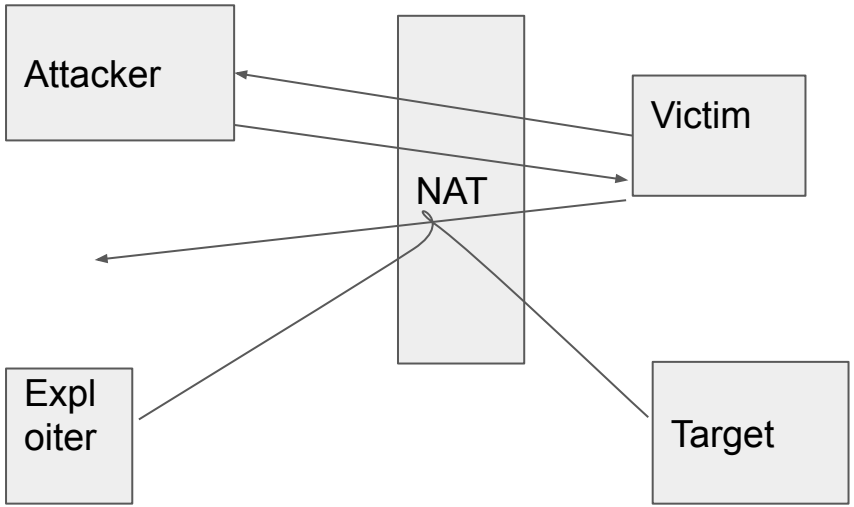
NAT Slipstream in 3 slides

Harald Alvestrand, Google

How NAT Slipstream works

- Attacker causes victim to load a page
- Victim sends something to the outer world
- The “something” gets broken into 2 packets by network layer
- The *second* packet is a valid request on an obscure protocol, understood by NAT
- This opens up a pinhole in the NAT
- Now “exploiter” can reach “target” - both of which are *unrelated* to attacker or victim.



Requirements to make this work

- Victim must generate a message larger than the MSS of the connection
  - With UDP: Larger than MTU.
  - With TCP: Can manipulate <mss> options to break “anywhere”
- Attacker must control the part of the message just after the break
- NAT box must interpret the port number + the message as a valid request

Defenses

- Don’t break up packets :-)
  - Unrealistic due to the TCP issue, and sometimes long UDP messages are valid.
  - Might want to detect strangely small TCP MSS as an IDS mechanism
- Don’t allow attacker control over packet content
  - Example: Restrict the length of a TURN USERNAME attribute
- Don’t allow connections to vulnerable port numbers
  - All of the most vulnerable ones seem to be in the 1-1023 port number range
  - Need to leave the “universal tunnel” ports 80 and 443 open
- Get rid of unused NAT functionality!
  - <heavy sigh>

Why it’s an RTCWEB matter

- RFC 8826 (Security considerations for WebRTC) doesn’t address attacks against middle boxes using RTCWEB protocols as the vector
- Being aware of the issue is a first line of defense
- Possible actions:
  - Do nothing
  - Write a short RFC describing this security consideration
  - Revise RFC 8826 to include this consideration
- Opinions?