UNIVERSITY OF AMSTERDAM

Involuntary Browser-Based Torrenting

Supervisor:
Jan Freudenreich

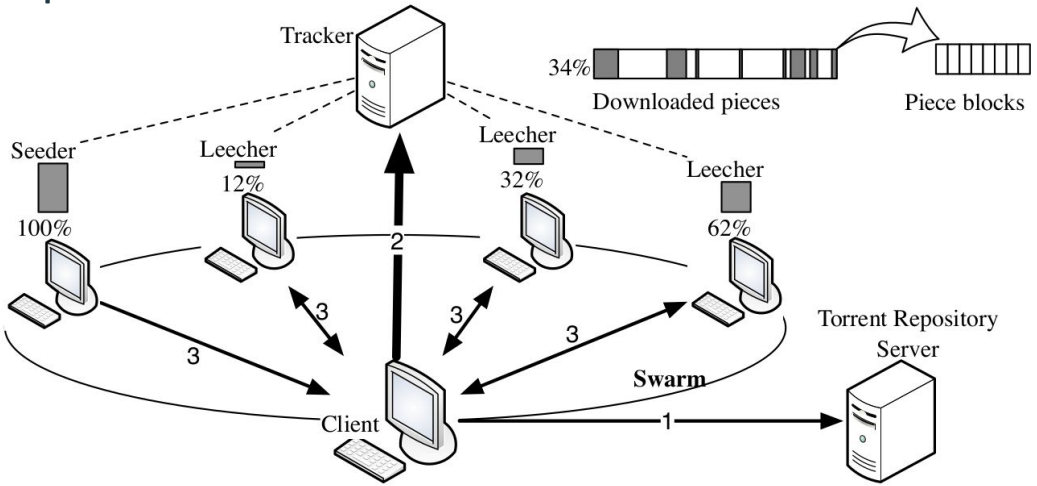
Course:
Research Project 2

Author:
Alexander Bode

BitTorrent

Protocol for distributing files using peer-to-peer connections.

- **BitTorrent Swarm**
  - Seeders
  - Leechers
- **Trackers**
  - Tracker Servers
  - Distributed Hash Tables
- **Repository Servers**
  - Torrents
  - Magnet URI's



Source: Enhanced BitTorrent Simulation using Omnet++, IEEE, 2020

Transport Protocol: TCP / uTP

Advantages of BitTorrent

- Every downloader is also an uploader
- Uses tit-for-tat principle for leeching
- No central point of failure
- Splits files into pieces
- Downloads rarest piece first
- Takes action with slow peers

Disadvantages of BitTorrent

- Torrent can’t complete if all seeds go offline and all leechers require a specific piece.
- IP address is exposed to the tracker and peers

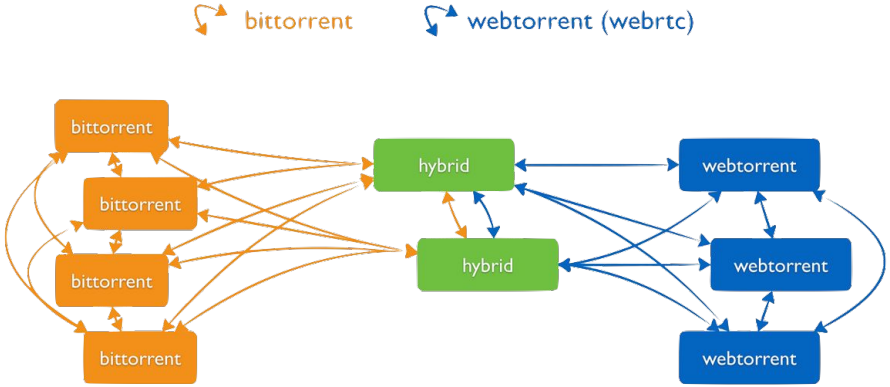
WebTorrent

First torrent client that works in a browser.

- Completely written in JavaScript
- WebRTC as transport protocol
- Custom tracker implementation, ICE
- Once peers connected, same as BitTorrent

Use Cases

- File sharing & streaming
- Peer-assisted delivery
- Hybrid clients as bridge to “normal” BitTorrent



Source: WebTorrent.io, 2020

Transport Protocol: WebRTC (on top TCP/UDP)

Research Questions

Main Research Question

Can WebTorrent be abused to have web page visitors involuntarily participate in peer-to-peer networks?

Sub Questions

- Which WebTorrent specific features can be abused?
- In which ways could WebTorrent be useful to an adversary?
- What can be done to prevent involuntary browser-based torrenting?
- Can we determine if this is an already established and widely used tactic?

Importance

- WebTorrent is attracting interest
- No additional installations are required for its use
- Security implications are unknown

Research Goals

- Determine whether involuntary browser-based torrenting is possible
- Usefulness for a potential adversary
- Detection and prevention methods
- Determine if it is a widely used and established tactic

Current Research

- Security Architecture of WebRTC (IETF)
- WebRTC Data Channels (HTML5Rocks)
- OakStreaming (Koren & Klamma)

Shortcomings

- No public research focused on WebTorrent security

Methodology

Lab Setup

Virtual Machine 1:

- OS: Kali Linux
- Browsers: Mozilla Firefox 76.0 & 81.0
- Purpose: PoC Development, Web Server, Debugging & Traffic analysis

Virtual Machine 2:

- OS: Windows 10
- Browser: Google Chrome 84.0
- Purpose: WebTorrent Client Testing

Mobile Device:

- OS: Android 9
- Browser: Google Chrome 85.0
- Purpose: WebTorrent Client Testing

Involuntary File-Sharing with WebTorrent

- Determine the relevant API methods of WebTorrent
- Write custom WebTorrent clients
  - WebTorrent Uploader
  - WebTorrent Downloader
- Debug and test custom client across various different devices
- Write, debug and test proof-of-concept scripts
- Determine attack vectors and the usefulness

Detection and Prevention

- **Search for existing methods**
  - Related work
  - Blog posts
- **Source code**
  - Search for static values
  - Search for unique patterns
- **Web Developer Tools**
  - JavaScript console to analyse the Window interface
  - Javascript Debugger to analyse WebTorrent code execution
- **WebRTC Internals**
  - Trace API calls
  - View connection details
- **WireShark**
  - Inspect traffic
- **Mozilla MDN Web Docs**
  - Analyse relevant API’s
- **Proof-of-concepts**
  - Userscripts
  - Browser extensions

Searching in the Wild

- **PublicWWW - Source Code Search Engine**
  - Search for code unique to WebTorrent
  - Search using regular expressions
  - Using over a half billion indexed pages
  - Export results for later analysis

Source: PublicWWW, 2020

Subscription was kindly provided by the PublicWWW team!

Results

Involuntary File-Sharing with WebTorrent

Involuntary browser-based torrenting is possible!

Attack Vectors

- Malicious / Compromised Web Server e.g. XSS
- Compromised externally hosted JavaScript library
- Malicious browser extension

Usefulness

Usefulness for an adversary

- **Resource Hijacking**
  - File sharing
  - Peer assisted-delivery

- **Repudiation**
  - Let users unknowingly download files

Detection & Prevention

- **Browser**
  - Detect and block WebTorrent usage using the *Window* interface
  - Blacklist URL’s of common trackers and common names of the library
  - Filter all responses containing JavaScript files (may break some pages)
  - Disable WebRTC, JavaScript or WebSockets

Detection & Prevention

- **Network**
  - Block DNS queries to trackers, ICE servers, library hosting domains
  - Deny access to trackers, ICE servers, library hosting domains

- **Compromised Web Server**
  - Use Indicators of Compromise
  - Check integrity of included remote library using Subresource Integrity (SRI)

Searching in the Wild

- **PublicWWW - Results**
  - Searched for script includes, unique patterns, obfuscated unique patterns
  - 307 pages indexed containing “webtorrent.min.js”
  - Other queries did not result in much

Nonetheless, results still useful for testing detection proof-of-concepts.

Proof-of-Concepts

Custom Clients

- Involuntary Stealth Downloader
- Involuntary Stealth Seeder
- JavaScript payload to be used for external loading e.g. XSS

Custom Mozilla Firefox Extensions

- WebTorrent Blocker
- Background Seeder
- WebTorrent Filter

Other

- Greasemonkey WebTorrent Blocker script
- uBlock Origin Static filter list

PoCs available at GitHub

Repository
https://github.com/alexander-47u/Involuntary-WebTorrent-Test

Discussion

Discussion

- Involuntary browser-based torrenting is possible!
- The browser and WebTorrent library do not ask for permission
- The findings could assist examiners in developing counter-measures
- Proof-of-concept for detection and prevention is functional
- Not a widely used and established tactic

Limitations

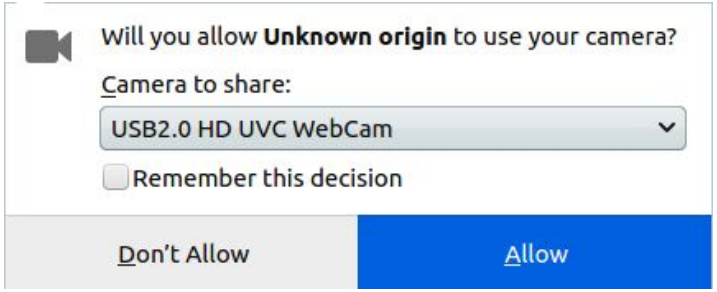
- Stealth Webtorrent downloads stop when page reloads/changes
- Browsers have limited cache for downloads
- WebTorrent Blocker extension depends on common names of objects

Limitations

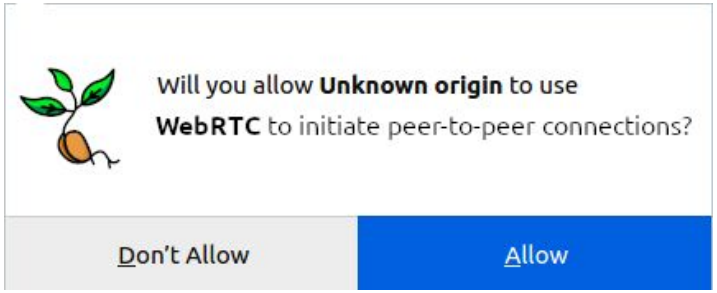
- Background Seeder extension requires initial seeder
- WebTorrent Filter slows down and sometimes breaks page

Recommendations

- **.getUserMedia()** prompts user for permission (camera, microphone)



- **No such method** or permission exists for WebRTC



Conclusion

Conclusion

Can WebTorrent be abused to have web page visitors involuntarily participate in peer-to-peer networks?

- Yes, although likely only useful for resource highjacking

Future Work

- Find more ways to use Involuntary WebTorrenting
- Investigate feasibility of different real-world attacks
- Methods for achieving persistence

Questions?

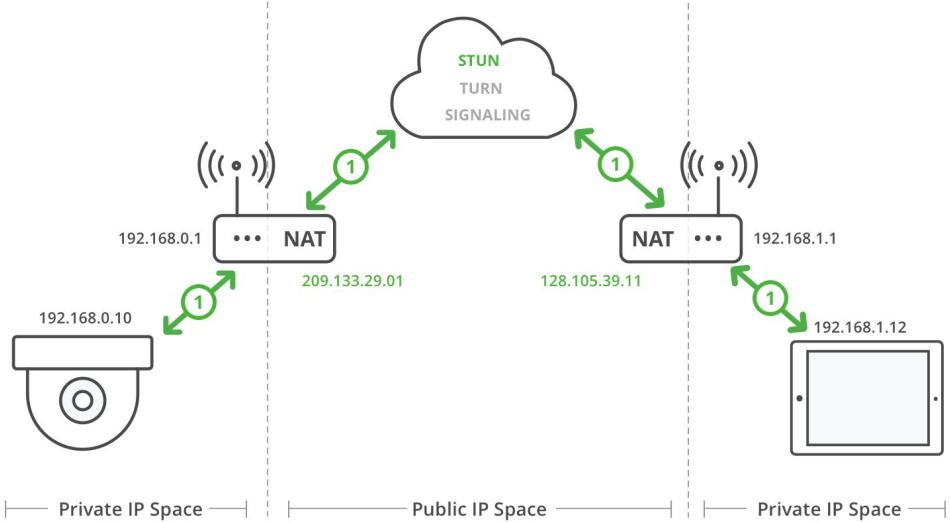
ICE Protocol

Technique used to find ways for peers to communicate as directly as possible.

- Used for NAT traversal
  - Session Traversal Utilities for NAT (STUN)
  - Traversal Using Relays around NAT (TURN)
    - Relay Extensions to STUN

ICE Protocol (P2P Behind NAT)

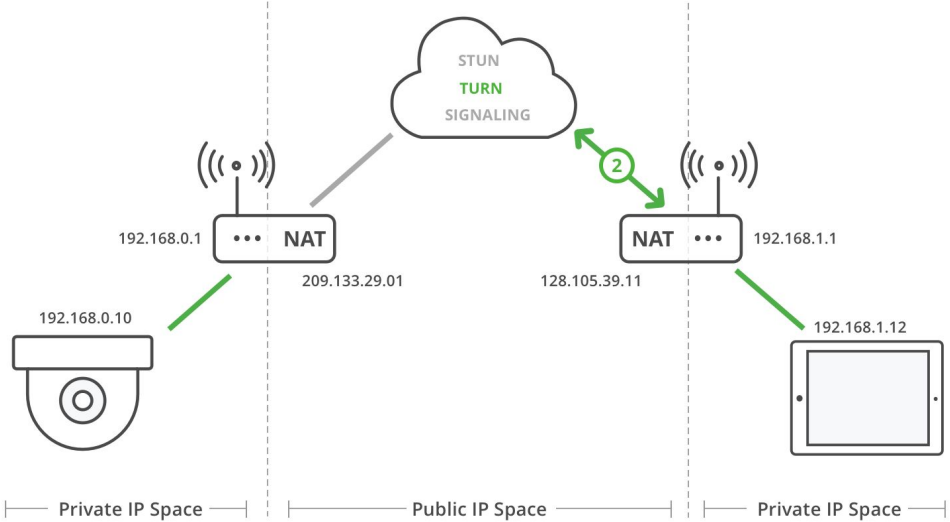
1. **STUN binding**



Source: AnyConnect, 2020

ICE Protocol (P2P Behind NAT)

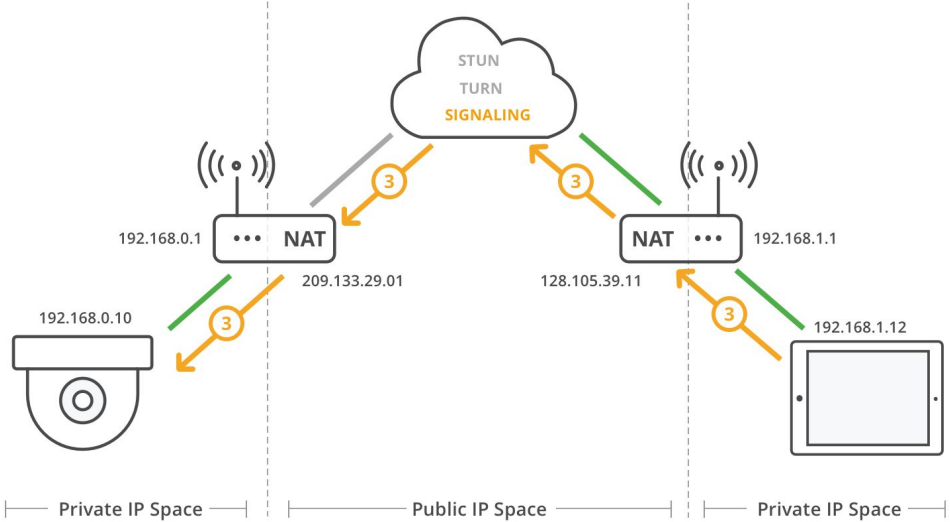
2. Caller TURN allocation



Source: AnyConnect, 2020

ICE Protocol (P2P Behind NAT)

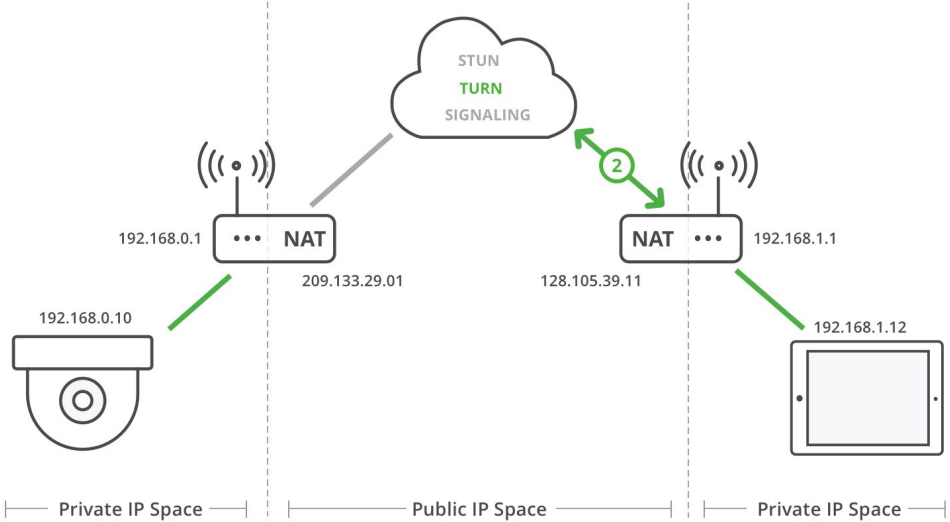
3. Caller sends invite



Source: AnyConnect, 2020

ICE Protocol (P2P Behind NAT)

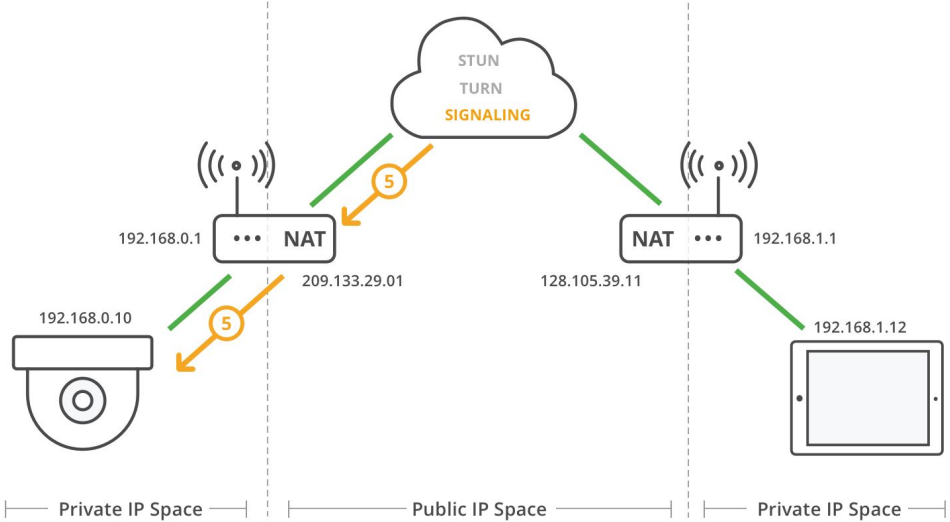
4. Callee TURN allocation



Source: AnyConnect, 2020

ICE Protocol (P2P Behind NAT)

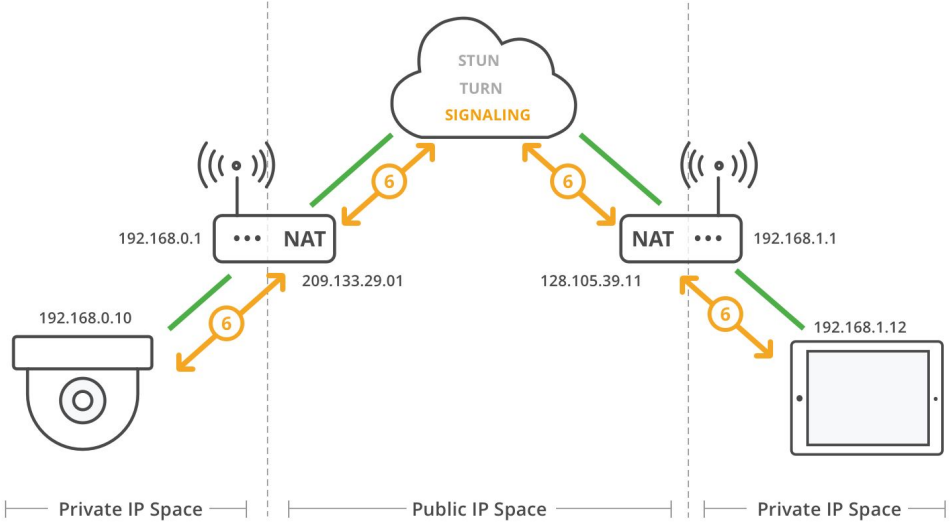
5. Callee answers OK



Source: AnyConnect, 2020

ICE Protocol (P2P Behind NAT)

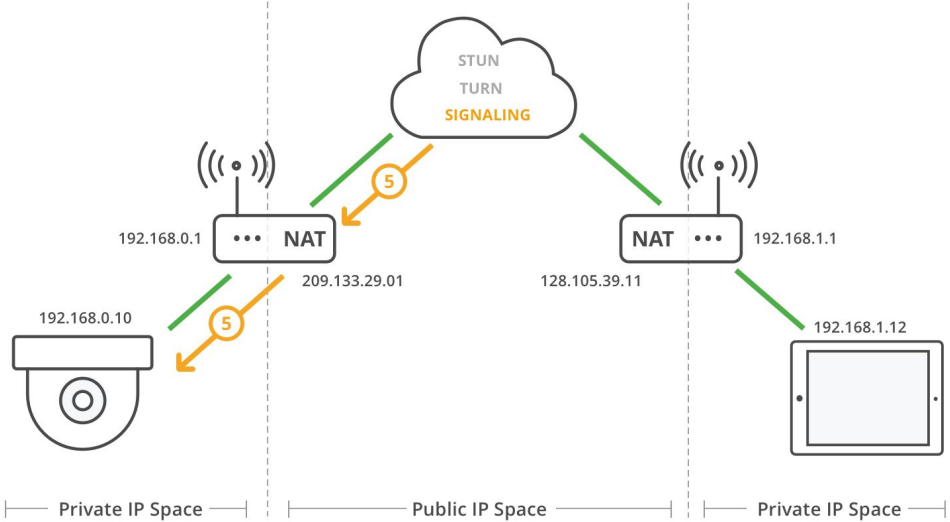
6. Exchange candidate IP addresses



Source: AnyConnect, 2020

ICE Protocol (P2P Behind NAT)

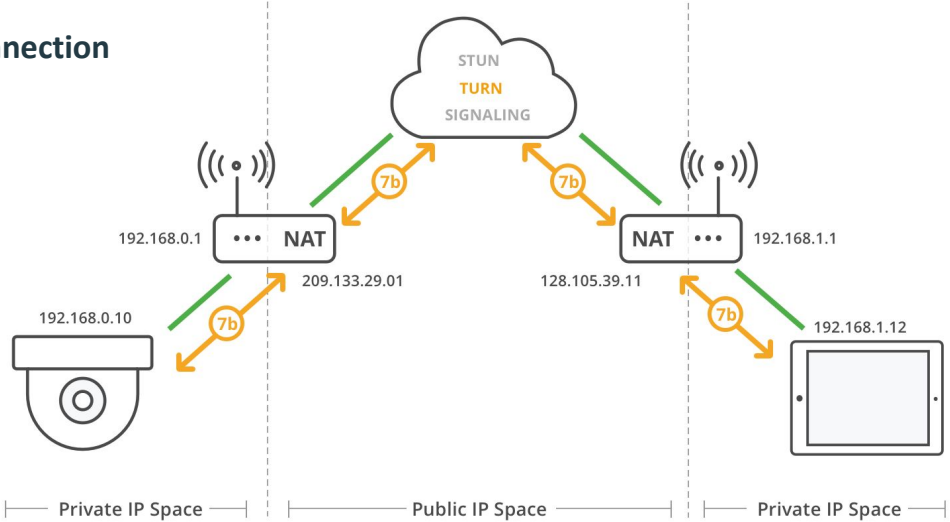
7. ICE check for P2P connection



Source: AnyConnect, 2020

ICE Protocol (P2P Behind NAT)

8. If P2P unsuccessful, make relay connection



Source: AnyConnect, 2020

BitTorrent DDoS Applicable?

Vulnerabilities that could be leveraged for DDoS were researched in 2015 in

- **Micro Transport Protocol (uTP):** No uses, WebRTC and then TCP or UDP
- **Distributed Hash Table (DHT)**: Not supported in the browser version of WebTorrent
- **Message Stream Encryption (MSE):** Not applicable
- **BitTorrent Sync (BTSync):** Not applicable

DDoS exploits do not apply to WebTorrent!

Research: *P2P File-Sharing in Hell: Exploiting BitTorrent Vulnerabilities to Launch Distributed Reflective DoS Attacks*

STUN Amplification Attack

Simple Traversal of UDP through NAT (STUN) amplification attack

1. STUN connectivity checks are directed to the target
2. Attacker proceeds by generating an offer with a large number of candidates
3. The peer endpoint, after receiving the offers, performs connectivity checks with all the candidates
4. Generate a significant volume of data flow with STUN connectivity checks

Can be mitigated by limiting the total number of candidates that are sent in an offer and response

Source: Microsoft Docs, 2020