Port Scanning and WebSockets

Tom Gallagher
NSA Information Assurance
tmgall4@empire.eclipse.ncsc.mil
Overview

- Background
- Existing mitigations
- Current weaknesses
- Proposed mitigation
Background

Base Assumption

- Attacker
- Internet
- Firewall
- Private Network

?
Background

Reasonable Assumption

Attacker

Firewall

Internet

127.0.0.1

Private Network
1. JavaScript can *make requests* to private IP addresses

2. JavaScript can *time responses* of those requests

3. An attacker can *infer from timing differences*
   - Short times indicate a responding port
   - Longer times indicate no response
A More Detailed Look

Proof of Concept Code (JavaScript)

```javascript
var ports = [80, 443, 445, 554, 3306, 3690, 1234];
for (var i = 0; i < ports.length; i++) {
  var s = new WebSocket("ws://192.168.1.1:" + ports[i]);
  s.start = performance.now();
  s.port = ports[i];
  s.onerror = function() { console.log("Port " + this.port + ": " + (performance.now() - this.start) + " ms"); }
  s.onopen = function() { console.log("Port " + this.port + ": " + (performance.now() - this.start) + " ms"); }
}
```

Timing Results

- **Port 80**: 632 ms
- **Port 443**: 3043 ms
- **Port 445**: 345 ms
- **Port 554**: 4254 ms
- **Port 3306**: 3697 ms
- **Port 3690**: 4113 ms
- **Port 1234**: 529 ms
Blocking: protects some well-known ports
Including: TCPMUX, Echo, Wake-on LAN, Systat, Daytime Protocol, Netstat, Chargen, FTP, SSH, Telnet, SMTP, TIME, WINS, WHOIS, DNS, Finger, Exchange, POP3, ONC RPC, Ident, Simple File Transfer, NNTP, NTP, MS RPC, NetBIOS, IMAP4, BGP, LDAP, Kerberos, rexec, rlogin, syslog, LPD/LPR, NFS

Throttling: reduces speed of port scanning
Mitigation Limitations

**Blocking**
- Browser inconsistency
- ~60 ports blocked
- Hard to be confident about additional blocking

**Throttling**
- Clustered address allocation
- Determined attackers
- WebSockets technique
WebSockets/WebWorkers

**Throttle**: *per process* limit on open connections

**WebWorkers**: child *processes*

**WebSockets**: arbitrary connections; *accessible from WebWorkers*

WebSocket x WebWorker = Faster Mapping
Proposed Mitigation

Focus on the WebSocket protocol

**Problem:** timing differences in handshake error

**Solution:** report errors at fixed latency
Does creating fixed latency *increase latency*?

Yes, but *only on error*

Enhanced security is worth the performance hit for an application already experiencing errors.
Mitigation Results

Proof of Concept Code (JavaScript)

```javascript
var ports = [80, 443, 445, 554, 3306, 3690, 1234];
for (var i = 0; i < ports.length; i++) {
    var s = new WebSocket("ws://192.168.1.1:" + ports[i]);
    s.start = performance.now();
    s.port = ports[i];
    s.onerror = function() { console.log("Port " + this.port + ": " + (performance.now() - this.start) + " ms"); };
    s.onopen = function() { console.log("Port " + this.port + ": " + (performance.now() - this.start) + " ms"); };
}
```

Current Mitigation Results

<table>
<thead>
<tr>
<th>Port</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>632</td>
</tr>
<tr>
<td>443</td>
<td>3043</td>
</tr>
<tr>
<td><strong>445</strong></td>
<td><strong>345</strong></td>
</tr>
<tr>
<td>554</td>
<td>4254</td>
</tr>
<tr>
<td>3306</td>
<td>3697</td>
</tr>
<tr>
<td>3690</td>
<td>4113</td>
</tr>
<tr>
<td>1234</td>
<td>529</td>
</tr>
</tbody>
</table>

Proposed Mitigation Results

<table>
<thead>
<tr>
<th>Port</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>6043</td>
</tr>
<tr>
<td>443</td>
<td>6021</td>
</tr>
<tr>
<td><strong>445</strong></td>
<td><strong>6125</strong></td>
</tr>
<tr>
<td>554</td>
<td>6104</td>
</tr>
<tr>
<td>3306</td>
<td>6028</td>
</tr>
<tr>
<td>3690</td>
<td>6110</td>
</tr>
<tr>
<td>1234</td>
<td>6031</td>
</tr>
</tbody>
</table>
What’s Next?

Internet-Draft: 
draft-gallagher-hybiWebSocketEnhancement-00

Browser adoption of updates

Consider this issue in future discussions of other protocols accessible from JavaScript