# Inter-Protocol Communication

Wade Alcorn
wade@ngssoftware.com
August 2006


## Abstract

This paper explores the Inter-Protocol Communication attack vector. That is, the potential of two different protocols meaningfully communicating commands and data. This has been investigated through encapsulating the target protocol within a carrier protocol. The findings demonstrate that under certain conditions distinct protocols are interoperable.

# Contents

# Introduction

Research within the area of web browser security, in particularly Cross-site scripting Viruses and Browser Exploitation Frameworks, has become a catalyst for further exploration into the broader area of Inter-Protocol Communication. That is, an attack vector which potentially allows arbitrary protocols to meaningfully interact with each other.

Web browsers create an ideal environment to investigate the impact of communication across protocols. Web browsers are on the majority of machines within a network, giving them the privileged position of being in virtually all sections of the network. They also have the ability to make arbitrary requests and receive responses from web servers on the Internet. Each of the requests asks a web server to provide direction as to what actions the web browser should take. The process relinquishes control of the web browser environment to the web server where it has at its disposal a variety of scripting languages and APIs. Web server responses are not guaranteed to be free of malicious content, whether this is from an untrustworthy source or from interference en-route. Malicious content could have the capability to direct the web browser to perform Inter-Protocol Communication.

Inter-Protocol Communication involves encapsulating the target protocol within another carrier protocol to facilitate the communication of commands and data. For successful communication across protocols, two preconditions need to be met. The protocol needs to be sufficiently error tolerant, and a method needs to exist to encapsulate the target protocol within the carrier protocol.

Discussion in this paper will focus almost exclusively on using a web browser as a client for Inter-Protocol Communication. This does not suggest that other protocols' implementations won't be capable of Inter-Protocol Communication.

# Inter-Protocol Communication

Inter-Protocol Communication is the communication between two distinct protocols. It uses encapsulation to allow two different protocols to communicate meaningfully, even though they were implemented using different protocol specifications.

The two protocols in Inter-Protocol Communication are termed the carrier protocol and the target protocol. The target protocol is the protocol that the receiver can interpret. The carrier protocol is the protocol that is being sent and that encapsulates the target protocol.

A chief consideration for Inter-Protocol Communication is the separators in the target protocol. Different protocols implement different methods to separate sections of data. In some situations semicolons are employed and in others a carriage return may be used. For example, IMAP3 uses carriage returns to separate sections of data. It will usually be a requirement for Inter-Protocol Communication that the carrier protocol accurately encapsulate separator data.

## *Requirements*

There are two main requirements for successful Inter-Protocol Communication. These requirements will not be consistent across different implementations of the same protocol. The variation will occur due to different interpretations of the protocol specifications.

### Error Tolerance

The first requirement is that the implementation of the target protocol be sufficiently forgiving of errors. During the Inter-Protocol connection it is likely that a percentage of the communication will be invalid and cause errors. Some protocol implementations permit only a certain number of errors before dropping the connection. For example, Exim version 4.50 only allows 4 errors before disconnecting the client. If the carrier protocol causes more errors than this maximum before communicating any of the encapsulated target protocol the Inter-Protocol Communication attempt will fail.

### Encapsulation

The second requirement is that the target protocol can be encapsulated in the carrier protocol. The encapsulation is unlikely to be perfect and will cause errors. Whilst it is possible binary protocols will interweave with each other, it is more probable that text based protocols will be more Inter-Protocol compatible.

## *HTTP and IMAP3 Example*

Inter-Protocol Communication can be performed from a HTTP carrier protocol to an IMAP3 target protocol. It is possible to encapsulate IMAP3 within a web request so that valid commands can be passed to an IMAP3 server.

### TCP Connection

During the setup of Inter-Protocol Communication (initiated from the web browser) the HTTP request to an IMAP3 server will perform a normal layer 4 handshake. This will start with the browser's host's network stack sending a SYN. The target host will respond with a SYN ACK which will be in turn answered with an ACK. This completes the setting up of the connection. At this point, no errors have occurred and either end can start sending data. It is also important to note that the implementation of web browsers explicitly prohibit connecting to some ports.

### Error Tolerance

After the TCP connection is established, the web browser will commence sending HTTP protocol data. At this point it is possible that both protocols will register errors and gracefully drop the connection. For Inter-Protocol Communication to be successful it will also be required that the IMAP3 implementation will need to be forgiving of errors. This allows the lines of the HTTP (carrier protocol) request to be ignored if they do not conform exactly to the IMAP3 (target protocol) specification. A perfect protocol transformation without errors is highly unlikely.

During the communication the majority of the HTTP header will error as the IMAP3 target protocol is unlikely to parse it as valid input. If there was limited error tolerance in the IMAP3 implementation and the number of HTTP header errors were greater than this tolerance, the Inter-Protocol attempt may not be successful.

```
POST /localhost HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
```

HTTP request header snippet

```
POST /localhost HTTP/1.0
POST BAD Command unrecognized/login please: /LOCALHOST
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
Accept: BAD Command unrecognized/login please: IMAGE/GIF,
```

IMAP3 response snippet errors

## Encapsulation

For successful Inter-Protocol Communication the HTTP protocol must be able to encapsulate the IMAP3 protocol. That is, interpretable commands or data must reach the IMAP3 server. The IMAP protocol uses single line commands containing a space and is terminated by a carriage return. Any successful encapsulation must send commands to the IMAP3 server in this format. To achieve this, the web browser can employ both HTTP multipart post requests and Javascript.

The encapsulation from the web browser is programmatically constructed by using a multipart form and submitting it to an IMAP3 server and port. The HTTP multipart post request provides a way to control the data at a low enough level to accomplish IMAP3 encapsulation.

The multipart post request delimits the content within the request into defined boundaries. Unlike other request types, the content is not URL encoded. The characters: carriage return, quotes, space, greater than and less than are represented as their ASCII value and not the escaped representation. This functionality allows for substantial control over this section and is ideal for the encapsulation of the IMAP3 target protocol. It provides a method to encapsulate the IMAP3 commands in the request.

After the TCP connection is established and the HTTP headers have produced errors the IMAP3 server will start to process the multipart post content. If the content has been constructed as IMAP3 commands (with carriage returns, etc) they will be executed on the server.

It is at this point that the target protocol is interpreted by the IMAP3 server as valid commands and Inter-Protocol Communication occurs. This protocol encapsulation demonstrates that it is possible for Inter-Protocol Communication from a web browser to an IMAP3 server.

```
----------------------------7d6195410340
Content-Disposition: form-data; name="ta"

```

```
a002 logout
```

HTTP request snippet sending the logout command

```
------------------------------7d6195410340
------------------------------7d6195410340 BAD Missing command
Content-Disposition: form-data; name="ta"
Content-Disposition: BAD Command unrecognized/login please: FORM-DATA;

* BAD Null command
a002 logout
* BYE debian IMAP4rev1 server terminating connection
a002 OK LOGOUT completed
```

IMAP3 response snippet interpret the logout command

## Bi-directional Inter-Protocol Communication

Inter-Protocol Communication can be both unidirectional and bi-directional. In Bi-directional Inter-Protocol Communication the initiator of the communication will be able to interpret the response.

The main difference is 759S R4ME79(M759SmR4VE88MF8SmR4V5VSeRVEMMFM5S R4ME79759SmF

## *HTTP and IMAP3 Example*

### Inter-Protocol Cross-site Scripting

Cross-site scripting is a vulnerability normally found in web applications by which an attacker finds a mechanism to inject a script into a victim's client. Inter-Protocol Cross-site scripting uses a non-HTTP protocol server to reflect the script into the web browser rather than a web application.

Web browsers will parse responses from arbitrary protocol servers provided they meet various pre-requisites. These pre-requisites vary between browser implementations and are not discussed in this paper.

The request is reflected off the target protocol server and becomes a mangled response containing errors, banners and other metadata. This response may contain verbose error messages which provide details of the errors occurring within protocol communication. It is these informative errors in which it may be possible to inject scripts to be executed by the web browser.

As discussed previously, a multipart post request provides sufficient control over the request to allow encapsulation of arbitrary IMAP3 commands. This, combined with the IMAP3 server providing verbose error messages, provides a mechanism for script injection.

A specifically crafted HTTP request to an IMAP3 server can contain Inter-Protocol Cross-site scripting that will be parsed and executed in a web browser. When the IMAP3 server receives an unknown command it responds by echoing the attempted command and stating that it was unrecognized. By attempting to execute the intentionally erroneous command "<script>alert('BAD COMMAND')</script>" it will be echoed back to the browser and parsed as a valid HTML script.

```
<script>alert('BAD COMMAND')</script> BAD Missing command
```

IMAP3 error containing a script tag

The following IMAP3alert function demonstrates Inter-Protocol Cross-site scripting. It will display an alert box within the browser showing content of the response.

```
var target_ip = '10.26.81.32';
var target_port = '220';

IMAP3alert(target_ip, target_port);

function IMAP3alert(ip, port) {

      // create the start of the form HTML
      var form_start = '<FORM name="multipart" ';
      form_start += 'id="multipart" action="http://';
      form_start += ip + ':' + port;
      form_start += '/dummy.html" ';
      form_start += 'type="hidden" ';
```

```
        form_start += 'enctype="multipart/form-data" ';
        form_start += 'method="post"> ';
        form_start += '<TEXTAREA NAME="commands" ROWS="0" COLS="0">';

        // create the end of the form HTML
        var form_end = '</TEXTAREA></FORM>';

        // create the commands
        cmd = "<scr"+"ipt>alert(document.body.innerHTML)</scr"+"ipt>\n";
        cmd += 'a002 logout' + "\n"; // IMAP3 logout command

        // create multipart form
        document.write(form_start);
        document.write(cmd);
        document.write(form_end);

        // send it
        document.multipart.submit();
}
```

IMAP3 alert function

## Inter-Protocol Fingerprinting

Fingerprinting is the method of determining details of (usually) remote systems through metadata. Operating system and protocol information can be attained in a variety of ways, including the simple extrapolation from a port number. In all methods there is a risk of false positives.

Inter-Protocol Cross-site scripting provides a new method to perform fingerprinting from within a web browser. After successful injection of Javascript within the unknown protocol, it is possible to access the “document.body.innerHTML" property. The innerHTML property provides (read and write) access to content body element and importantly in this instance it contains the response of the unknown protocol.

Next, the unknown protocol response can be transferred by using HTTP requests to an arbitrary server. A simple method to achieve this is to concatenate innerHTML property onto the end of a HTTP get request.

```
url = ‘http://www.egress-site.com/’ + escape(document.body.innerHTML);
```

Egress javascript concatenation command

Programmatically, finger printing the remote protocol will involve the content of the unknown protocol response being retrieved and then transferred to a remote server. From there any number of algorithms could be employed to digest the results and fingerprint the protocol.

## Brute Force Attacks

Brute forcing is a method that systematically attempts all authentication possibilities to gain access to a system. The major weakness is the amount of time the attack will take to complete. There are common methods to increase the likelihood of a successful attempt, for example, using high probability usernames and passwords.

Once a protocol is known to allow Inter-Protocol Cross-site scripting and has been successfully fingerprinted, there is a potential for a brute force attack. If the protocol requires authentication credentials these could be brute forced from within the web browser. Like traditional brute force applications, the target protocol negotiations and authentication attempts will custom to the protocol.

# Further Research

## *Inter-Protocol Exploitation*

One possible direction for further research in Inter-Protocol Communication is Inter-Protocol Exploitation. This is the process where by a vulnerability is exploited within the boundaries of a different carrier protocol. The protocol carrying the crafted exploit data will still abide to its protocol specification. Situations in which Inter-Protocol Exploitation is possible are likely to be limited, as it requires specific preconditions which are dictated by the carrier protocol implantation and target vulnerability.

## *Inter-Port Cross-site Scripting*

Web browsers vary in how restrictive DOM security is when accessing frames that retrieved data from different ports within the same domain. In some instances, a web browser may permit javascript to access the content of these frames even where Inter-Protocol Communication isn't attempted.

# Conclusion

In the past, it has been assumed that communication between different protocols is invalid and of no consequence. However this paper has demonstrated that when a particular set of preconditions are met, communication between two distinct protocols is possible and significant. These preconditions are that the protocol implementations are sufficiently error tolerant, and that a method exists to encapsulate the target protocol within the carrier protocol.

The resultant Inter-Protocol Communication has been demonstrated through examples of interaction between HTTP and IMAP3. This was achieved through a web browser encapsulating valid IMAP3 commands within a HTTP request. This in turn, provoked a response from the IMAP3 server, returning error messages that were deliberately crafted. These error messages were parsed by the web browser as valid HTML, resulting in the execution of scripts. This sequence of events proves that bi-directional and meaningful communication is possible.

Whilst some implications of Inter-Protocol Communication have been discussed, for example Inter-Protocol Cross-site scripting, this remains an area for further research. What is evident, however, at this early stage is that Inter-Protocol Communication is a attack vector worthy of significant consideration.

## About Next Generation Security Software (NGS)

NGS is the trusted supplier of specialist security software and hi-tech consulting services to large enterprise environments and governments throughout the world. Voted "best in the world" for vulnerability research and discovery in 2003, the company focuses its energies on advanced security solutions to combat today's threats. NGS maintains the largest penetration testing and security cleared CHECK team in EMEA. Founded in 2001, NGS is headquartered in Sutton, Surrey, with research offices in Scotland and Australia, and works with clients on a truly international level.

## About NGS Insight Security Research (NISR)

The NGS Insight Security Research team are actively researching and helping to security flaws in popular off-the-shelf products. As the world leaders in vulnerability discovery, NISR release more security advisories than any other commercial security research group in the world.

# References

1. The Cross-site Scripting Virus
   http://www.bindshell.net/papers/xssv

2. Browser Exploitation Framework
   http://www.bindshell.net/tools/beef

3. HTML Code Injection and Cross-site scripting
   http://www.technicalinfo.net/papers/CSS.html

4. XSS (Cross-site Scripting) Cheat Sheet
   http://ha.ckers.org/xss.html

5. CGISecurity's Cross-site Scripting FAQ
   http://www.cgisecurity.com/articles/xss-faq.shtml

6. Wikipedia Javascript
   http://en.wikipedia.org/wiki/Javascript

7. HTML 4.01 Specification
   http://www.w3.org/TR/html4/present/frames.html

8. Bugtraq Posting - 1998
   http://archive.cert.uni-stuttgart.de/archive/bugtraq/1998/10/msg00046.html

9. HTML Form Protocol Attack
   http://www.remote.org/jochen/sec/hfpa/index.html

10. Extended HTML Form Attack
    http://eyeonsecurity.org/papers/Extended%20HTML%20Form%20Attack.htm

11. Mozilla Port Blocking
    http://www.mozilla.org/projects/netlib/PortBanning.html

12. Interactive Mail Access Protocol - Version 3
    http://tools.ietf.org/html/rfc1203