Bro Internet Content Adaptation Protocol (ICAP) Analyzer

A Novel Method for Monitoring HTTPS Traffic in Plain-Text

M. Fernandez | MITRE

Presentation for

BroCon '16

Austin, Texas | 13-15 Sep 2016
Problem Statement

- **Encrypted Web Traffic**
  - Transport layer encryption (HTTPS)
  - Status quo for search engines, web mail, etc.
  - Blind spot for typical network security & monitoring
  - Potential vector for external and internal cyber threats
  - Majority of web traffic

- **Countermeasure**
  - Best practice... SSL/TLS-interception security device
  - Or perhaps... Web proxy w/content inspection?
  - Bro ICAP Analyzer
Outline

- **ICAP**
  - Background
  - Basic Operation
  - Web Proxies, Content Inspection & ICAP
  - References

- **Bro ICAP Analyzer**
  - Analyzing ICAP
  - Creating the Bro Analyzer via BinPAC
  - Caveats & Limitation

- **Recommendations for Future Work**
Internet Content Adaptation Protocol

- Internet Engineering Task Force (IETF) Request for Comments (RFC) 3507
  - Simple object-based content vectoring for HTTP
  - Content modification of either HTTP request/response messages
  - Syntax similar to HTTP
  - TCP port 1344

- Common Implementations
  - Web proxy devices with content inspection service
    - Anti-Virus (AV) / Malware
    - Data Loss Prevention (DLP)
ICAP Operation

- **Request Modification (REQMOD)**
  - Modifies HTTP request messages
  - ICAP payload:
    - HTTP request header
    - HTTP request body [chunk-encoded]

- **Response Modification (RESPMOD)**
  - Modifies HTTP response messages
  - ICAP payload:
    - HTTP request header
    - HTTP response header
    - HTTP response body [chunk-encoded]
ICAP Request Modification

Illustration adapted from RFC 3507

Legend
- HTTP/HTTPS Traffic
- ICAP Traffic

1. Web Client
2. Web Proxy (aka, ICAP Client)
3. DLP Proxy (aka, ICAP Server)
4. Web Server
5. (connection)
6. (connection)
ICAP Request Modification

Illustration adapted from RFC 3507

Legend
- HTTP/HTTPS Traffic
- ICAP Traffic

1. HTTP req msg
2. Web Server
3. DLP Proxy (aka, ICAP Server)
4. Web Proxy (aka, ICAP Client)
5. Web Client
6. HTTP req msg
ICAP Request Modification

Web Server

Web Proxy (aka, ICAP Client)

ICAP Payload

http req hdr
http req bdy

DLP Proxy (aka, ICAP Server)

Legend

HTTP/HTTPS Traffic
ICAP Traffic

Illustration adapted from RFC 3507
ICAP Request Modification

Illustration adapted from RFC 3507

Legend
- HTTP/HTTPS Traffic
- ICAP Traffic

1. Web Client
2. Web Proxy (aka, ICAP Client)
3. Web Server
4. DLP Proxy (aka, ICAP Server)
5. Web Proxy sends ICAP Request to DLP Proxy
6. DLP Proxy returns ICAP Response to Web Proxy

ICAP Resp Msg
204 “No modifications needed”
ICAP Request Modification

Illustration adapted from RFC 3507

Web Server

Web Proxy (aka, ICAP Client)

DLP Proxy (aka, ICAP Server)

Web Client

ICAP Payload

http req hdr

http req bdy

Legend

HTTP/HTTPS Traffic

ICAP Traffic

© 2016 The MITRE Corporation. All rights reserved.
ICAP Request Modification

Illustration adapted from RFC 3507

Legend

- HTTP/HTTPS Traffic
- ICAP Traffic

1. Web Client
2. Web Proxy (aka, ICAP Client)
3. ICAP Payload
   - http req hdr
   - modified bdy
4. DLP Proxy (aka, ICAP Server)
5. Web Server
6. Web Client

© 2016 The MITRE Corporation. All rights reserved.
ICAP Response Modification

Legend
- HTTP/HTTPS Traffic
- ICAP Traffic

1. Web Client
2. Web Server
3. Web Proxy (aka, ICAP Client)
4. AV Proxy (aka, ICAP Server)
5. Web Proxy
6. Web Client

Illustration adapted from RFC 3507
ICAP Response Modification

Illustration adapted from RFC 3507

Legend

- HTTP/HTTPS Traffic
- ICAP Traffic

1. HTTP req msg from Web Client
2. HTTP rsp msg to Web Client
3. HTTP req msg from Web Proxy
4. HTTP rsp msg to AV Proxy
5. HTTP req msg from AV Proxy
6. HTTP rsp msg to AV Proxy

Web Client

Web Proxy (aka, ICAP Client)

HTTP/HTTPS Traffic

AV Proxy (aka, ICAP Server)

ICAP Traffic
ICAP Response Modification

Legend

- HTTP/HTTPS Traffic
- ICAP Traffic
ICAP Response Modification

Illustration adapted from RFC 3507

Legend
- HTTP/HTTPS Traffic
- ICAP Traffic

1. Web Client
2. Web Proxy (aka, ICAP Client)
3. AV Proxy (aka, ICAP Server)
4. ICAP Resp Msg
   204 “No modifications needed”
5. Web Server
6. Web Client

© 2016 The MITRE Corporation. All rights reserved.
ICAP Response Modification

Illustration adapted from RFC 3507

Legend
- HTTP/HTTPS Traffic
- ICAP Traffic

1. Web Server
2. Web Proxy (aka, ICAP Client)
3. HTTP rsps msg
4. AV Proxy (aka, ICAP Server)
5. Web Client
6. HTTP/HTTPS Traffic

© 2016 The MITRE Corporation. All rights reserved.
ICAP Response Modification

Illustration adapted from RFC 3507

Legend
- HTTP/HTTPS Traffic
- ICAP Traffic

1. Web Client
2. Web Server
3. Web Proxy (aka, ICAP Client)
4. AV Proxy (aka, ICAP Server)
5. ICAP Payload
   - http req hdr
   - http rsp hdr
   - http rsp body
6. Web Client
ICAP Response Modification

Illustration adapted from RFC 3507

Legend
- HTTP/HTTPS Traffic
- ICAP Traffic

Web Server

Web Proxy (aka, ICAP Client)

ICAP Payload
- http req hdr
- http rsp hdr
- modified bdy

AV Proxy (aka, ICAP Server)

Web Client

1. Web Client requests content from Web Server.
2. Web Server sends initial HTTP response to Web Proxy.
4. AV Proxy checks the content for malicious content.
5. AV Proxy sends back modified HTTP response to Web Proxy.

© 2016 The MITRE Corporation. All rights reserved.
ICAP Response Modification

Illustration adapted from RFC 3507

Legend:
- **HTTP/HTTPS Traffic**
- **ICAP Traffic**

1. **Web Client**
2. **Web Server**
3. **Web Proxy (aka, ICAP Client)**
4. **AV Proxy (aka, ICAP Server)**

Steps:
1. **HTTP rsp msg** from **Web Client** to **Web Proxy**
2. **HTTP rsp msg** from **Web Proxy** to **Web Server**
3. **HTTP rsp msg** from **Web Server** to **Web Proxy**
4. **HTTP rsp msg** from **Web Proxy** to **AV Proxy**
5. **HTTP rsp msg** from **AV Proxy** to **Web Proxy**
6. **HTTP rsp msg** from **Web Proxy** to **Web Client**
The Bro ICAP Analyzer

Legend

HTTP/HTTPS Traffic

ICAP Traffic

Web Server

Web Proxy
(aka, ICAP Client)

AV / DLP Proxy
(aka, ICAP Server)

Web Client

Illustration adapted from RFC 3507
ICAP References

[1] Internet Content Adaptation Protocol (ICAP)
   - Request for Comments (RFC) 3507
   - J. Elson & A. Cerpa (2003 April)

[2] ICAP Extensions
   - IETF Draft
   - M. Stecher et al. (2003 April)

[3] ICAP Partial Content Extension
   - Draft (unofficial)
   - M. Stecher & A. Rousskov (2010 May)
ICAP References – cont.

[4] ICAP Errata

- Measurement Factory (© 2014)
  - http://www.measurement-factory.com/std/icap
The Bro ICAP Analyzer

Objectives

- Monitor link between web proxy and AV/DLP proxy
  - IPs & Ports, Connection IDs

- Extract HTTPS messages from ICAP payload
  - Analyze ICAP headers
  - Generate icap.log

- Invoke the Bro HTTP analyzer
  - Invokes MIME analyzer, File analyzer, and...
  - Generates http.log, files.log, conn.log, and...
Analyzing ICAP Requests & Responses

- **ICAP Methods**
  - REQMOD
  - RESPMOD
  - OPTIONS
  - LOG *

- **ICAP Status Codes**
  - 1xx Informational
  - 2xx Success: 204 No modifications needed
  - 3xx Redirection
  - 4xx Client errors
  - 5xx Server errors

*LOG defined in [2] ICAP Extensions*
# Analyzing ICAP Headers

*Headers defined in [1] RFC 3507*

<table>
<thead>
<tr>
<th>ICAP Request Headers</th>
<th>ICAP Response Headers</th>
<th>ICAP Options Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow</td>
<td>Allow</td>
<td>Allow</td>
</tr>
<tr>
<td>Authorization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cache-Control</td>
<td>Cache-Control</td>
<td></td>
</tr>
<tr>
<td>Connected</td>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Date</td>
<td>Date</td>
</tr>
<tr>
<td>Encapsulated</td>
<td>Encapsulated</td>
<td>Encapsulated</td>
</tr>
<tr>
<td>Expires</td>
<td>Expires</td>
<td></td>
</tr>
<tr>
<td>From</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Host</td>
<td>ISTag</td>
<td>ISTag</td>
</tr>
<tr>
<td></td>
<td>Max-Connections</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Methods</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Opt-body-type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Options-TTL</td>
<td></td>
</tr>
<tr>
<td>Pragma</td>
<td>Pragma</td>
<td></td>
</tr>
<tr>
<td>Preview</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Referer</td>
<td>Server</td>
<td>Service</td>
</tr>
<tr>
<td></td>
<td>Service-ID</td>
<td></td>
</tr>
<tr>
<td>Trailer</td>
<td>Trailer</td>
<td>Transfer-Preview</td>
</tr>
<tr>
<td>Upgrade</td>
<td>Upgrade</td>
<td>Transfer-Ignore</td>
</tr>
<tr>
<td>User-Agent</td>
<td></td>
<td>Transfer-Complete</td>
</tr>
</tbody>
</table>
# Analyzing Extended Headers

<table>
<thead>
<tr>
<th>ICAP Request Extensions</th>
<th>ICAP Response Extensions</th>
<th>ICAP Options Extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Authenticated-User</td>
<td>X-ICAP-Profile</td>
<td>X-Include</td>
</tr>
<tr>
<td>X-Authenticated-Groups</td>
<td>X-Attribute</td>
<td>Attribute-List resp body</td>
</tr>
<tr>
<td>X-Client-IP</td>
<td>X-Attribute-Cacheability</td>
<td></td>
</tr>
<tr>
<td>X-Server-IP</td>
<td>X-Attribute-Prefix</td>
<td></td>
</tr>
<tr>
<td>X-Subscriber-ID</td>
<td>X-Infection-Found</td>
<td></td>
</tr>
<tr>
<td>X-LOG-[service-ID]</td>
<td>X-Violations-Found</td>
<td></td>
</tr>
<tr>
<td>New Method: LOG</td>
<td>X-Virus-ID</td>
<td></td>
</tr>
<tr>
<td>LOG-[service-ID]</td>
<td>X-Response-Info</td>
<td></td>
</tr>
</tbody>
</table>

*Headers defined in [2] ICAP Extensions*

<table>
<thead>
<tr>
<th>ICAP Request Extensions</th>
<th>ICAP Response Extensions</th>
<th>ICAP Options Extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-patience</td>
<td>use-original-body</td>
<td>'Allow' header allowed</td>
</tr>
</tbody>
</table>

*Headers defined in [3] ICAP Partial Content Extension*
Analyzing Packet Captures

**RESPMOD Request Packet**

ICAP_Request_Line
{   ICAP Method
    ICAP URI
    ICAP Version
}  
ICAP_Headers
{   Host
    X-Scan-Progress-Interval
    X-Client-IP
    X-Server-IP
    X-Authenticated-User
    Allow
    Encapsulated
}  
ICAP_Payload
{   HTTP Request Header
    HTTP Response Header
    HTTP Response Body
}

**RESPMOD Reply Packet**

ICAP_Response_Line
{   ICAP Version
    ICAP Status Code
    ICAP Reason
}  
ICAP_Headers
{   Date
    Service
    ITag
    Service-ID
    X-Scan-Progress
    X-Apparent-Data-Types
}  
ICAP_Payload, if applicable
{   HTTP Response Header
    HTTP Response Body
}
The Encapsulated Header

Per RFC 3507 [pg 17]

REQMOD request:  [req-hdr] req-body
REQMOD response:  {{[req-hdr] req-body} ||
                      {[rsp-hdr] rsp-body}}
RESPMOD request:  [req-hdr] [rsp-hdr] rsp-body
RESPMOD response:  [rsp-hdr] [rsp-body]
OPTIONS response:  opt-body || null-body

NOTE: only one (1) body can be encapsulated within ICAP payload.

Example:

RESPMOD request:
Encapsulated: req-hdr=0, rsp-hdr=440, rsp-body=990\x0d\x0a
Building the Bro ICAP Analyzer

Platform
- Linux CentOS 6.7 Server
- 8-core CPU
- Two 1-Gbps NIC

Bro
- Version 2.4.1
- Local Cluster
  - 1 Manager, 1 Proxy
  - 6 Workers [pin_cpus=2,3,4,5,6,7]
- PF_RING
Building the Bro ICAP Analyzer

BinPAC

- Version 0.44
  - https://www.bro.org/downloads/binpac-0.44.tar.gz
- BinPAC QuickStart Guide
  - https://github.com/grigorescu/binpac_quickstart/archive/master.zip
## Building the Bro ICAP Analyzer

### Source Files

<table>
<thead>
<tr>
<th>Source Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C++, BIF &amp; BinPAC Files</strong></td>
<td>src/analyzer/protocol/icap/</td>
</tr>
<tr>
<td>CMakeLists.txt</td>
<td>Indicates which compiler to use against which source-code files (C++ or BIF or BinPAC compiler).</td>
</tr>
<tr>
<td>ICAP.cc &amp; .h</td>
<td>Defines C++ class ICAP_Analyzer.</td>
</tr>
<tr>
<td>Plugin.cc</td>
<td>Defines C++ class Bro_ICAP::Plugin.</td>
</tr>
<tr>
<td>events.bif</td>
<td>Declares events generated by the ICAP analyzer.</td>
</tr>
<tr>
<td>icap.pac</td>
<td>Top-level BinPAC declarations.</td>
</tr>
<tr>
<td>icap-protocol.pac</td>
<td>Protocol-specific BinPAC declarations, defines data elements based on RFC 3507.</td>
</tr>
<tr>
<td>icap-analyzer.pac</td>
<td>Additional code launched after protocol-specific data elements are parsed, throws ICAP events.</td>
</tr>
<tr>
<td>icap-analyzer-http.pac</td>
<td>Additional code to assist processing and invoking the HTTP analyzer.</td>
</tr>
<tr>
<td>icap-analyzer-utils.pac</td>
<td>Additional code to perform useful functions.</td>
</tr>
<tr>
<td><strong>Scriptland Files</strong></td>
<td>scripts/base/protocols/icap/</td>
</tr>
<tr>
<td>main.bro</td>
<td>Bro script that handles and logs ICAP events.</td>
</tr>
<tr>
<td>dpd.sig</td>
<td>Bro dynamic protocol detection (DPD) script file is used to detect the ICAP protocol over a non-standard port.</td>
</tr>
<tr>
<td><strong>load</strong>.bro</td>
<td>Declares which ICAP-related scripts to load at Bro startup. By default, both 'main.bro' and 'dpd.sig' are loaded at startup.</td>
</tr>
</tbody>
</table>

### Build Files

<table>
<thead>
<tr>
<th>Build Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BIF Files</strong></td>
<td>build/src/analyzer/protocol/icap/</td>
</tr>
<tr>
<td>events.bif.cc</td>
<td>Auto-generated by BIF compiler and moved into the Bro build tree.</td>
</tr>
<tr>
<td>events.bif.h</td>
<td></td>
</tr>
<tr>
<td>events.bif.init.cc</td>
<td></td>
</tr>
<tr>
<td>events.bif.register.cc</td>
<td></td>
</tr>
<tr>
<td><strong>BinPAC Files</strong></td>
<td>build/src/analyzer/protocol/icap/</td>
</tr>
<tr>
<td>icap_pac.cc</td>
<td>Auto-generated by BinPAC compiler and moved into the Bro build tree.</td>
</tr>
<tr>
<td>icap_pac.h</td>
<td></td>
</tr>
</tbody>
</table>
## Bro ICAP Events & Weird Log

### Bro ICAP Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>icap_request_line</td>
<td>Generated after REQUEST LINE is parsed</td>
</tr>
<tr>
<td>icap_response_line</td>
<td>Generated after RESPONSE LINE is parsed</td>
</tr>
<tr>
<td>icap_header</td>
<td>Generated after HEADER field is parsed</td>
</tr>
<tr>
<td>icap_options</td>
<td>Generated after OPTIONS BODY is parsed</td>
</tr>
<tr>
<td>icap_body_weird</td>
<td>Generated if unexpected BODY format encountered</td>
</tr>
<tr>
<td>icap_chunk_weird</td>
<td>Generated if sum of chunks not equal to HTTP ‘content-length’</td>
</tr>
<tr>
<td>icap_error</td>
<td>Generated for errors when decoding ICAP Requests &amp; Responses</td>
</tr>
<tr>
<td>icap_done</td>
<td>Generated after a complete ICAP transaction:</td>
</tr>
<tr>
<td></td>
<td>✓  ICAP Request followed by ICAP Response; and</td>
</tr>
<tr>
<td></td>
<td>✓  After invoking HTTP analyzer.</td>
</tr>
</tbody>
</table>

### Bro ICAP Weird

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unrecognized ICAP Methods</td>
<td>ICAP_WEIRD: unknown ICAP method &lt;string&gt;</td>
</tr>
<tr>
<td>Unrecognized ICAP Versions</td>
<td>ICAP_WEIRD: unknown ICAP version &lt;string&gt;</td>
</tr>
<tr>
<td>Unrecognized ICAP Status Codes</td>
<td>ICAP_WEIRD: unknown ICAP status code &lt;string&gt;</td>
</tr>
<tr>
<td>Unrecognized ICAP Header Names</td>
<td>ICAP_WEIRD: header: &lt;string_1&gt; : &lt;string_2&gt; :: method : &lt;string_3&gt; : is_orig : &lt;string_4&gt;</td>
</tr>
<tr>
<td>Unrecognized ICAP Body Format</td>
<td>ICAP_WEIRD: unknown ICAP body format &lt;string_1&gt; :: method : &lt;string_2&gt; : is_orig : &lt;string_3&gt;</td>
</tr>
</tbody>
</table>
BinPAC Files: icap.pac & icap-protocol.pac

**icap.pac**

```
enum ICAP_MSG_BODY_TYPES {
    BODY_TYPE_NONE,  # Message Body not present.
    BODY_TYPE_ACD,   # RESPMOD: (a) req-hdr, (c) rsp-hdr, (d) rsp-body
    BODY_TYPE_AC,    # RESPMOD: (a) req-hdr, (c) rsp-hdr, (f) null-body
    BODY_TYPE_CD,    # RESPMOD: (c) rsp-hdr, (d) rsp-body
    BODY_TYPE_D,     # RESPMOD: (d) rsp-body
    BODY_TYPE_AB,    # REQMOD: (a) req-hdr, (b) req-body
    BODY_TYPE_A,     # REQMOD: (a) req-hdr, (f) null-body
    BODY_TYPE_B,     # REQMOD: (b) req-body
    BODY_TYPE_OPTS,  # OPTIONS: (e) opt-body
    BODY_TYPE_WEIRD, # Unexpected body format
}
```

**icap-protocol.pac**

```
ICAP_Request {
    ICAP_Request_Line
    ICAP_Message
}

ICAP_Response {
    ICAP_Response_Line
    ICAP_Message
}

ICAP_Message {
    ICAP_Headers
    ICAP_Body
}

ICAP_Request_Line {
    ICAP_Method
    ICAP_URI
    ICAP_Version
}

ICAP_Response_Line {
    ICAP_Version
    ICAP_Status_Code
    ICAP_Reason
}

ICAP_Headers {
    Array of ICAP_Header
}
BinPAC Files: icap-protocol.pac

**icap-protocol.pac — cont.**

```plaintext
type ICAP_Message(is_orig : bool) = record
{
    headers     : ICAP_Headers(is_orig);
    body        : case $context.flow.get_icap_body_type_from_encap_hdr(headers, is_orig) of
    {
        BODY_TYPE_ACD    -> acd     : ICAP_Body_acd(is_orig);
        BODY_TYPE_AC     -> ac      : ICAP_Body_ac(is_orig);
        BODY_TYPE_CD     -> cd      : ICAP_Body_cd(is_orig);
        BODY_TYPE_D      -> d       : ICAP_Body_d(is_orig);
        BODY_TYPE_AB     -> ab      : ICAP_Body_ab(is_orig);
        BODY_TYPE_A      -> a       : ICAP_Body_a(is_orig);
        BODY_TYPE_B      -> b       : ICAP_Body_b(is_orig);
        BODY_TYPE_OPTS   -> opts    : ICAP_Body_options(is_orig);
        default         -> none    : empty;
    }
};

type ICAP_Body_acd(is_orig : bool) = record
{
    encap_req_hdr   : ICAP_Encapsulated_Http_Headers;
    encap_rsp_hdr   : ICAP_Encapsulated_Http_Headers;
    encap_rsp_bdy   : ICAP_Chunks(is_orig);
};
```

© 2016 The MITRE Corporation. All rights reserved.
## BinPAC Files: icap-analyzer.pac & -utils.pac

<table>
<thead>
<tr>
<th>icap-analyzer.pac</th>
<th>Event Generation:</th>
<th>HTTP Injection:</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc_icap_request_line()</td>
<td>icap_request_line</td>
<td>none</td>
</tr>
<tr>
<td>proc_icap_response_line()</td>
<td>icap_response_line</td>
<td>none</td>
</tr>
<tr>
<td>proc_icap_header()</td>
<td>icap_header</td>
<td>none</td>
</tr>
<tr>
<td>proc_icap_body_xxx()</td>
<td>none</td>
<td>proc_http_invoke_analyzer</td>
</tr>
<tr>
<td>proc_icap_options()</td>
<td>icap_options</td>
<td>none</td>
</tr>
<tr>
<td>proc_icap_pdu()</td>
<td>icap_done</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>icap-analyzer-utils.pac</th>
<th>Event Generation:</th>
<th>HTTP Injection:</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_icap_body_type_from_encap_hdr()</td>
<td>icap_body_weird</td>
<td>none</td>
</tr>
</tbody>
</table>
BinPAC Files: icap-analyzer-http.pac

**icap-analyzer-http.pac**

- **proc_http_invoke_analyzer()**
  - Top-level function called by `proc_icap_body_x`;
  - Calls `_submit_all_headers` and `_submit_body`

- **proc_http_invoke_analyzer_submit_all_headers()**
  - Calls `HTTP.cc :: HTTP_Analyzer::DeliverStream` to inject Headers into HTTP protocol analyzer
  - Must submit each header field one-by-one

- **proc_http_invoke_analyzer_submit_body()**
  - Calls `HTTP.cc :: HTTP_Analyzer::DeliverStream` to inject Body into HTTP protocol analyzer
  - Must check original Transfer Encoding

- **proc_http_reassemble_body()**
  - Used only for the purpose of creating an HTTP Body that is **NOT** chunk-encoded
  - Event Generation: `icap_chunk_weird`

**ICAP.h**

```cpp
#include "analyzer/protocol/http/HTTP.h"

class ICAP_Analyzer ... {
  public:
    static analyzer::Analyzer* HttpAnalyzer(Connection* conn) {
      return new analyzer::http::HTTP_Analyzer(conn);
    }
}
```

© 2016 The MITRE Corporation. All rights reserved.
BinPAC Files: Bugs & Challenges

➢ Compiler Error in icap_pac.cc

<table>
<thead>
<tr>
<th>icap.pac</th>
<th>Workaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>let body_ : int=BODY_TYPE_NONE</td>
<td>✤ Added global variable ‘body_’ within icap.pac</td>
</tr>
<tr>
<td></td>
<td>✤ Needed to fix a C-compiler error in 'icap_pac.cc :: ICAP_Message::ParseBuffer'.</td>
</tr>
<tr>
<td></td>
<td>✤ BinPAC compiler created the variable 'body_' in icap_pac.cc file but never defined it in icap_pac.cc or icap_pac.h.</td>
</tr>
</tbody>
</table>
BinPAC Files: Bugs & Challenges

ParseBuffer() Fails to Parse ‘chunk_data’

**BIT-1500**: “BinPAC Call to FlowBuffer::NewFrame with frame_length -1”

```pascal
type TEST_Chunk = record {
    len_line : bytestring &oneline;
    chunk_data : bytestring &length = chunk_length;
    opt_crlf : case chunk_length of {
        0   -> none: empty;
        default   -> crlf: bytestring &oneline;
    };
} &let {
    chunk_length : int = bytestring_to_int(len_line, 16); # in hexadecimal
};

type TEST_Chunks = record {
    chunks : TEST_Chunk[] &until($element.chunk_length == 0);
};
```

**Resolution (V. Grigorescu):**

```pascal
TEST_Chunk { TEST_Chunk_Size { TEST_Chunk_Data { bytestring } }
} TEST_Chunk_Size { bytestring &length = TEST_Chunk_Size;
}```
Caveats & Limitations

➤ Operational Testing
  o Biased toward RESPMOD transactions
  o REQMOD not yet observed/tested
  o OPTIONS & Preview headers ignored

➤ Bro Connection 4-tuples & Identifiers
  o IP addresses derived from ICAP extended headers
  o TCP port is always 1344
  o Connection IDs overlap multiple unrelated user-sessions per ICAP session
Caveats & Limitations

- **REQMOD vs RESPMOD**
  - REQMOD yields HTTP request body
  - RESP MOD yields HTTP response body
  - Need both for full visibility

- **Transport Layer vs Application Layer Encryption**
Summary

- **Encrypted Web Traffic**
  - Blind spot for typical network security & monitoring
  - Majority of web traffic

- **Web proxy w/content inspection & SSL/TLS interception capabilities?**
  - If so... Bro ICAP Analyzer!
  - ICAP headers yield user ID and original IPs
  - ICAP payload yields decrypted copy of HTTPS messages
Future Work

- REQMOD Testing

- Revisit/optimize BinPAC code:
  - ICAP_Message & Encapsulated Headers
  - body__global variable

- Submit ICAP Analyzer to Bro Project Team
Questions?
Back-Up Slides