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

MITRE

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

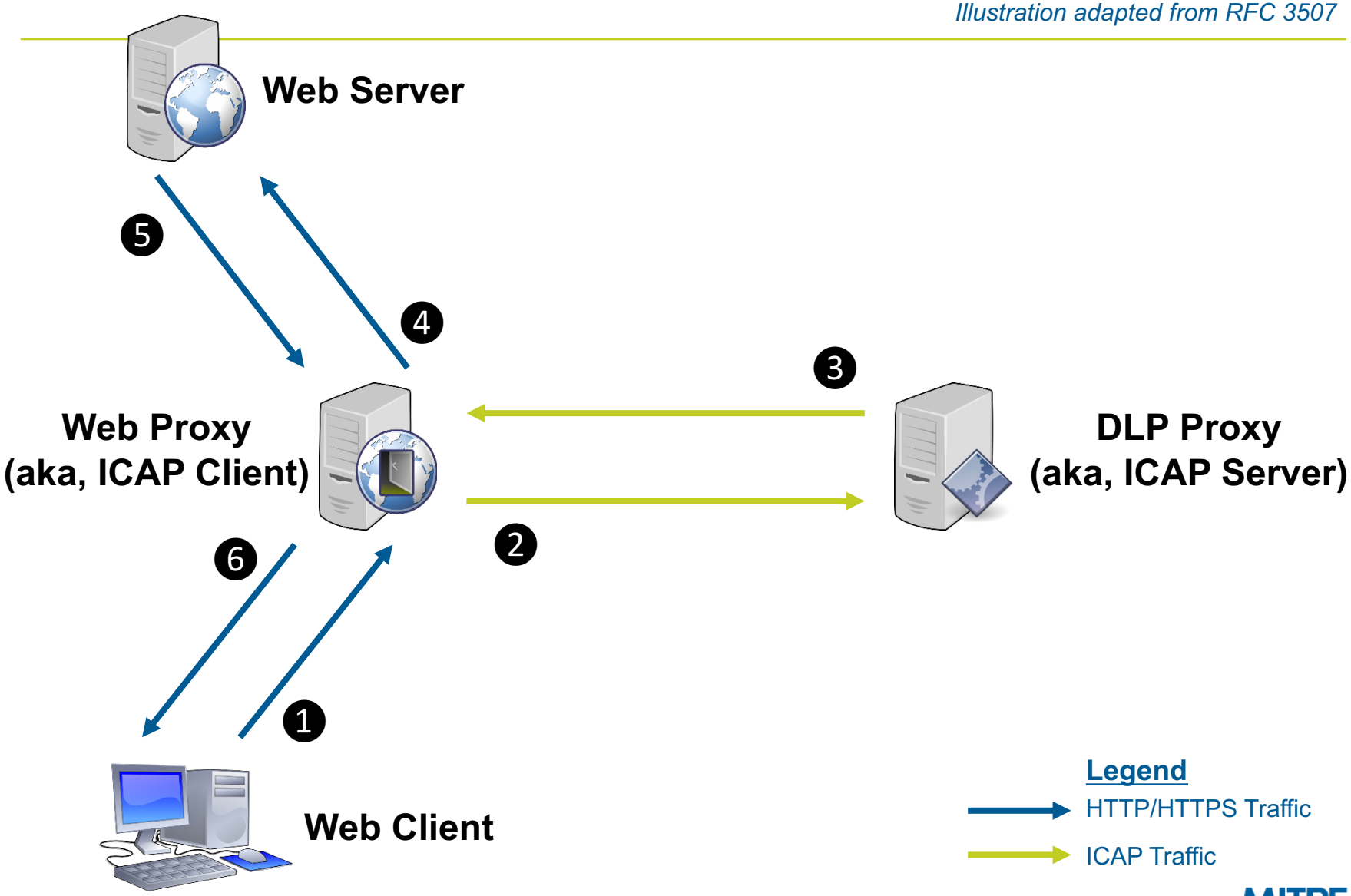
- **Internet Engineering Task Force (IETF) Request for Comments (RFC) 3507**
  - Simple object-based content vectoring for HTTP
  - Content modification of either HTTP request/response messages
  - Syntax similar to HTTP
  - TCP port 1344
- **Common Implementations**
  - Web proxy devices w/content inspection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

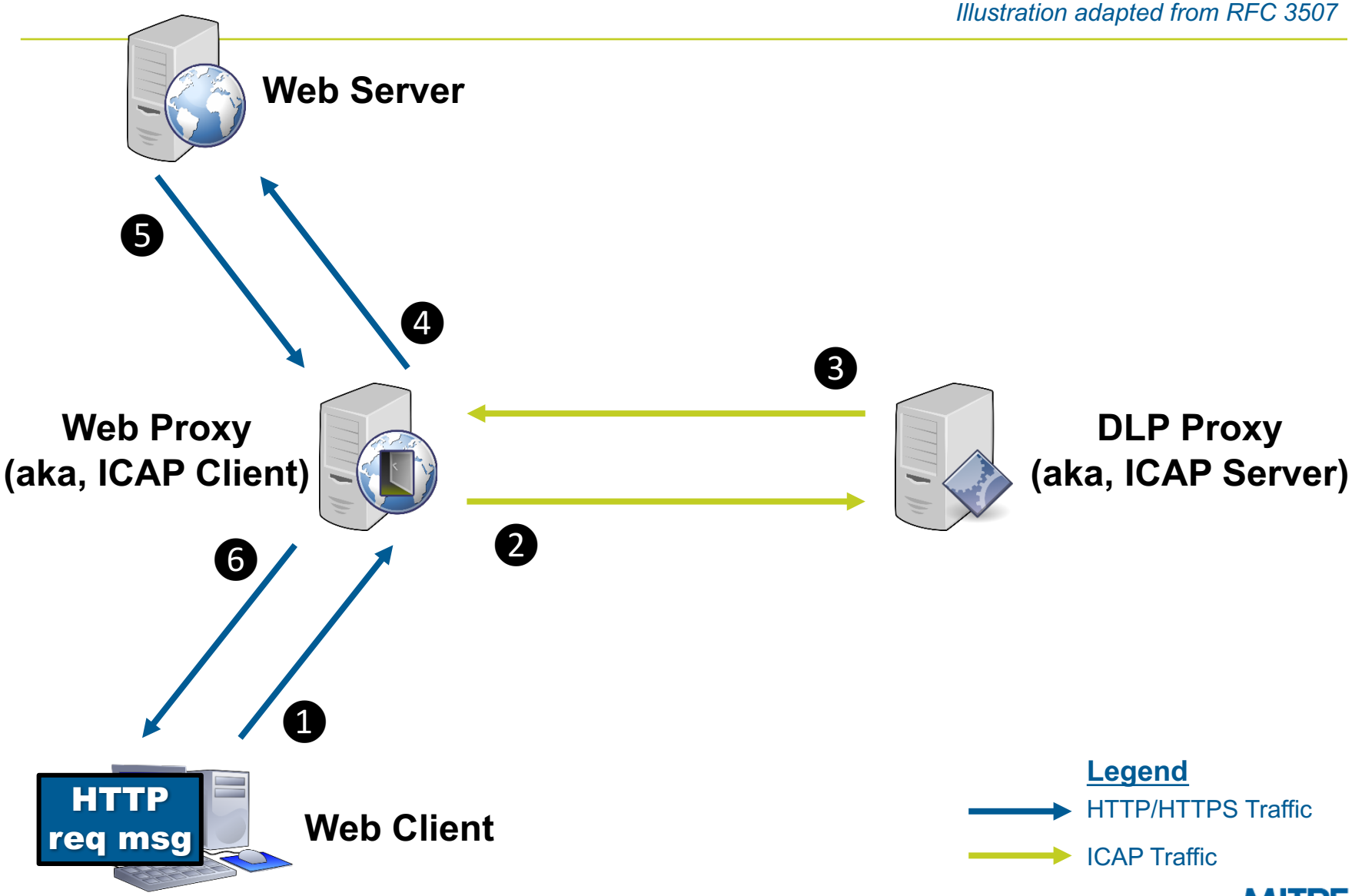
Illustration adapted from RFC 3507



MITRE

ICAP Request Modification

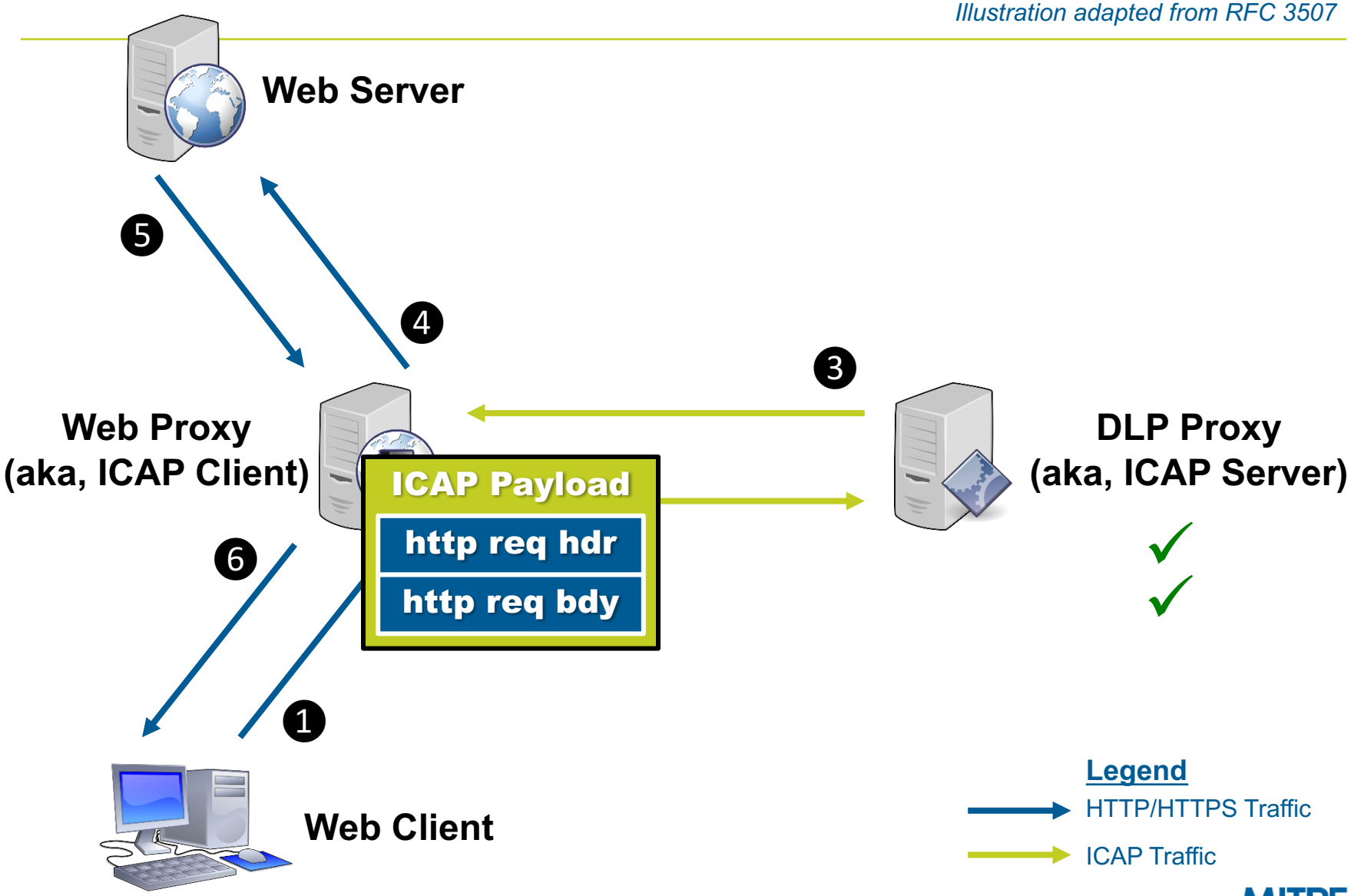
Illustration adapted from RFC 3507



MITRE

ICAP Request Modification

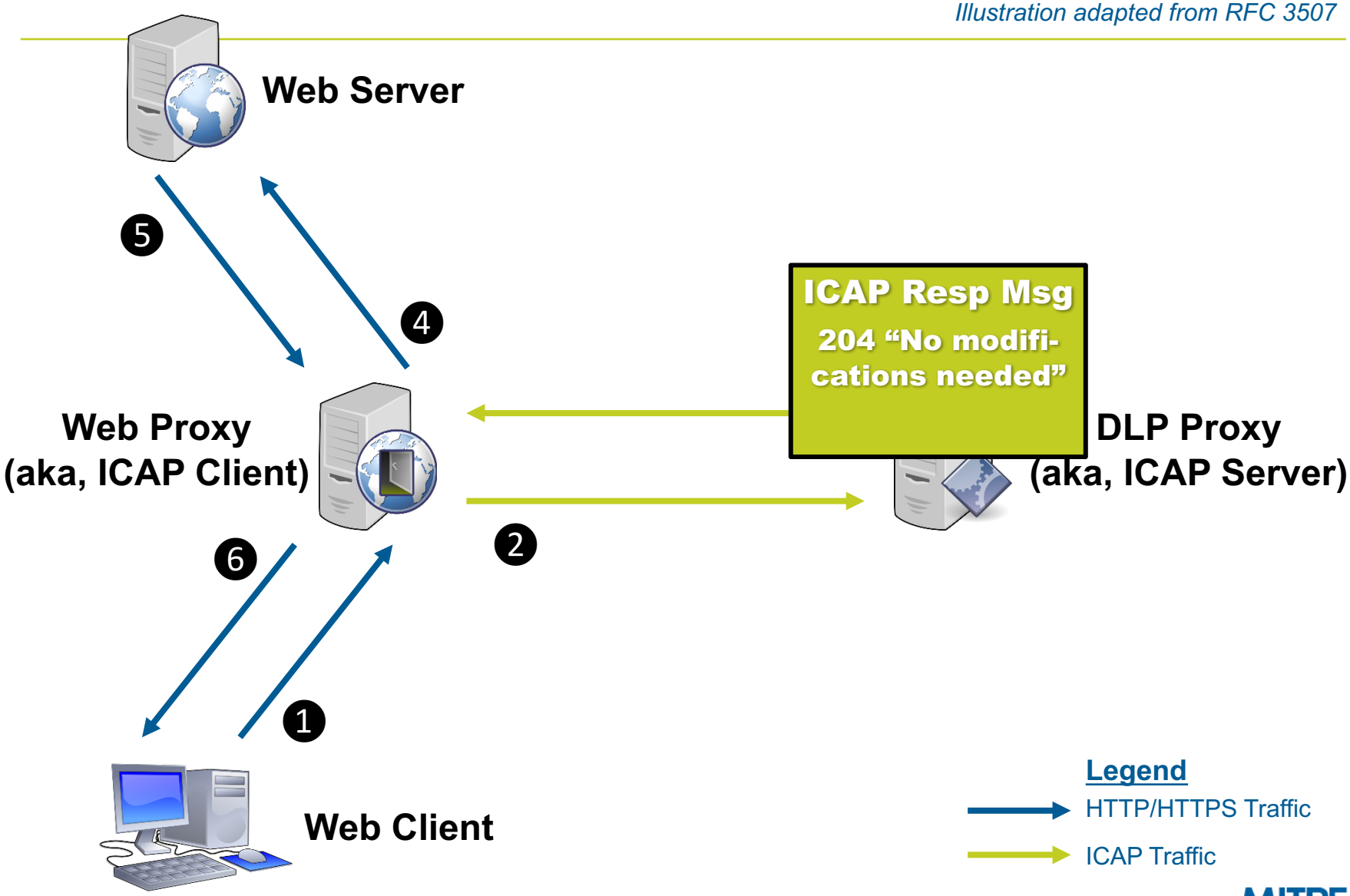
Illustration adapted from RFC 3507



MITRE

ICAP Request Modification

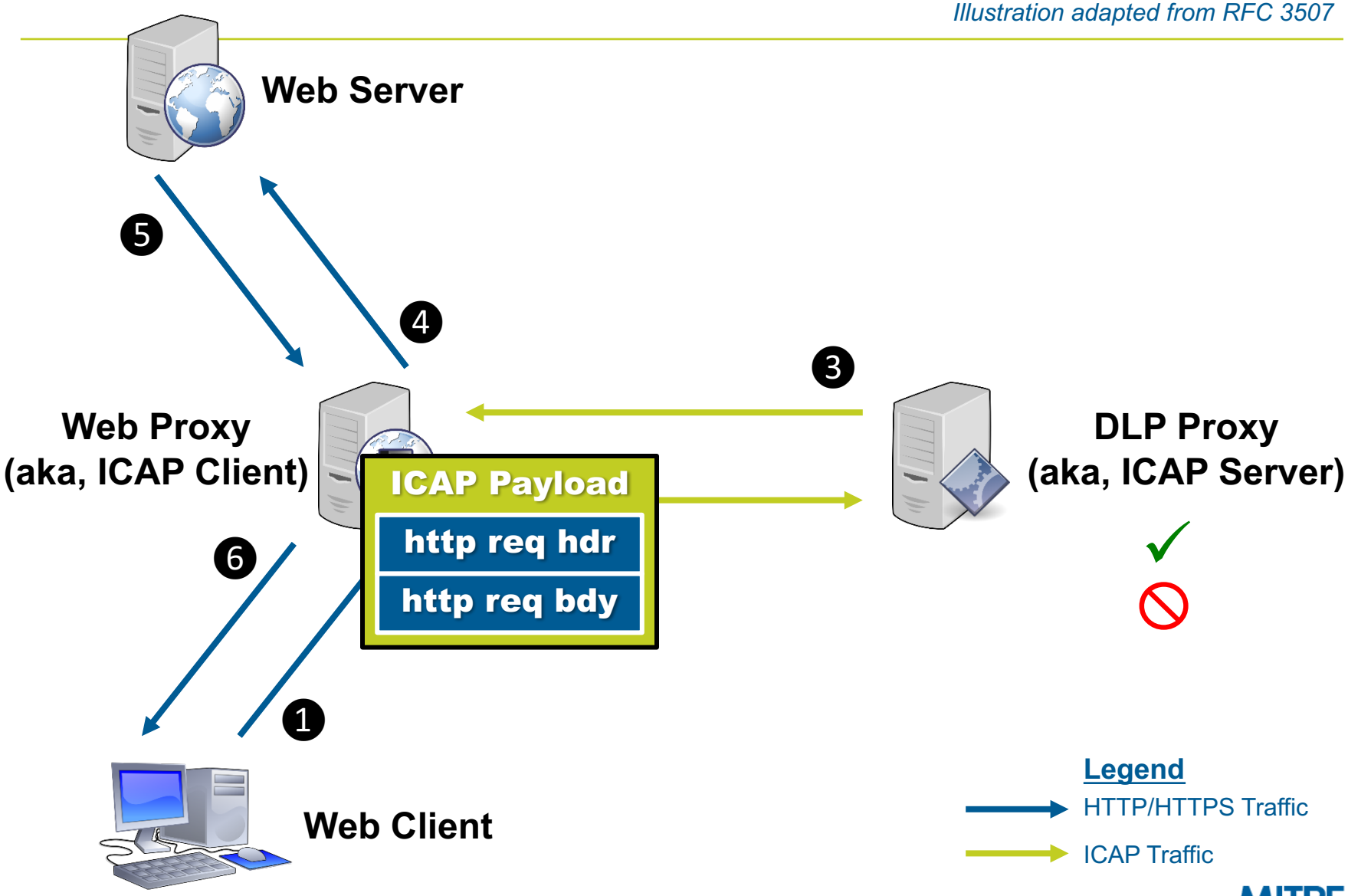
Illustration adapted from RFC 3507



MITRE

ICAP Request Modification

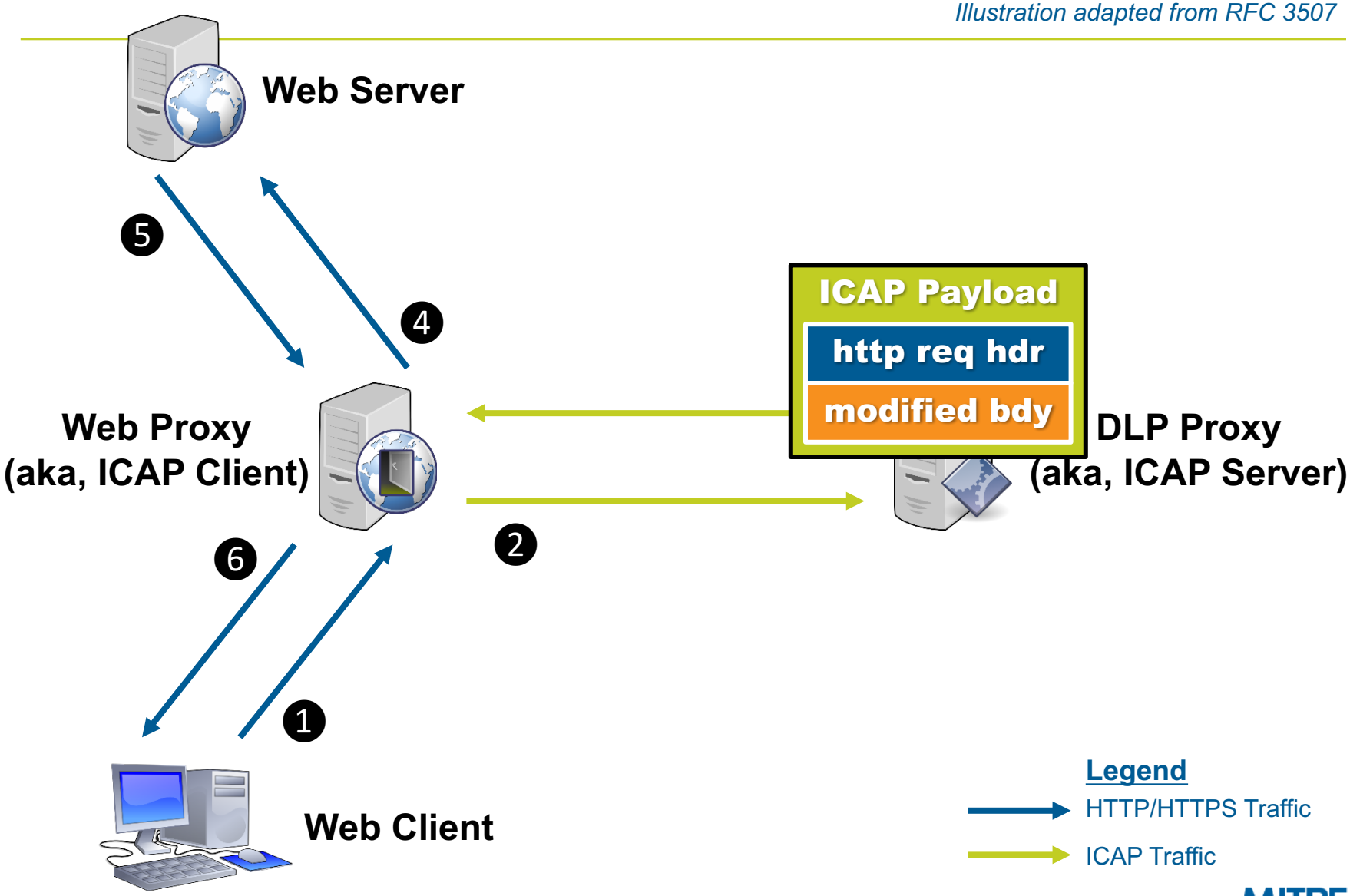
Illustration adapted from RFC 3507



MITRE

ICAP Request Modification

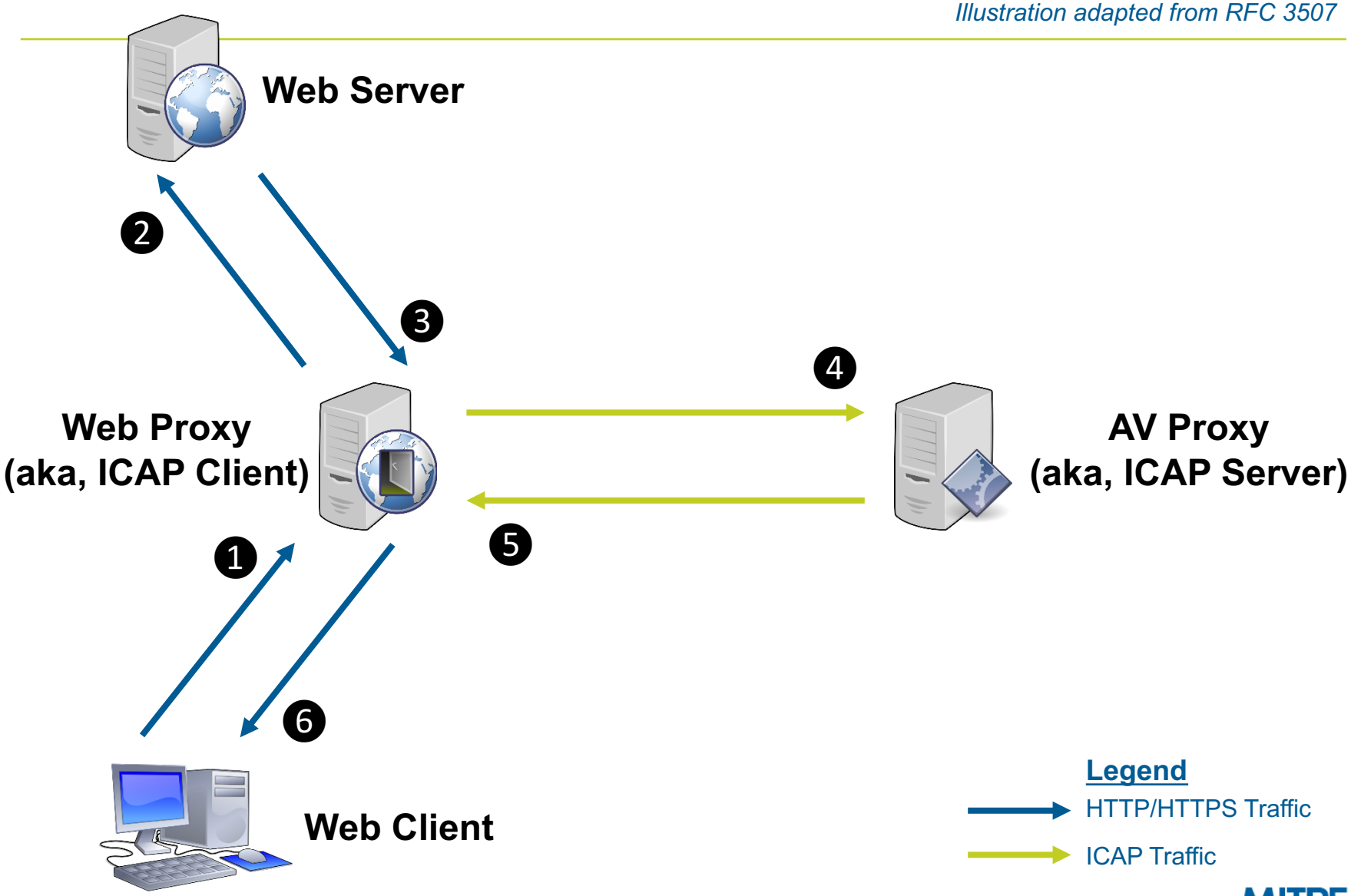
Illustration adapted from RFC 3507



MITRE

ICAP Response Modification

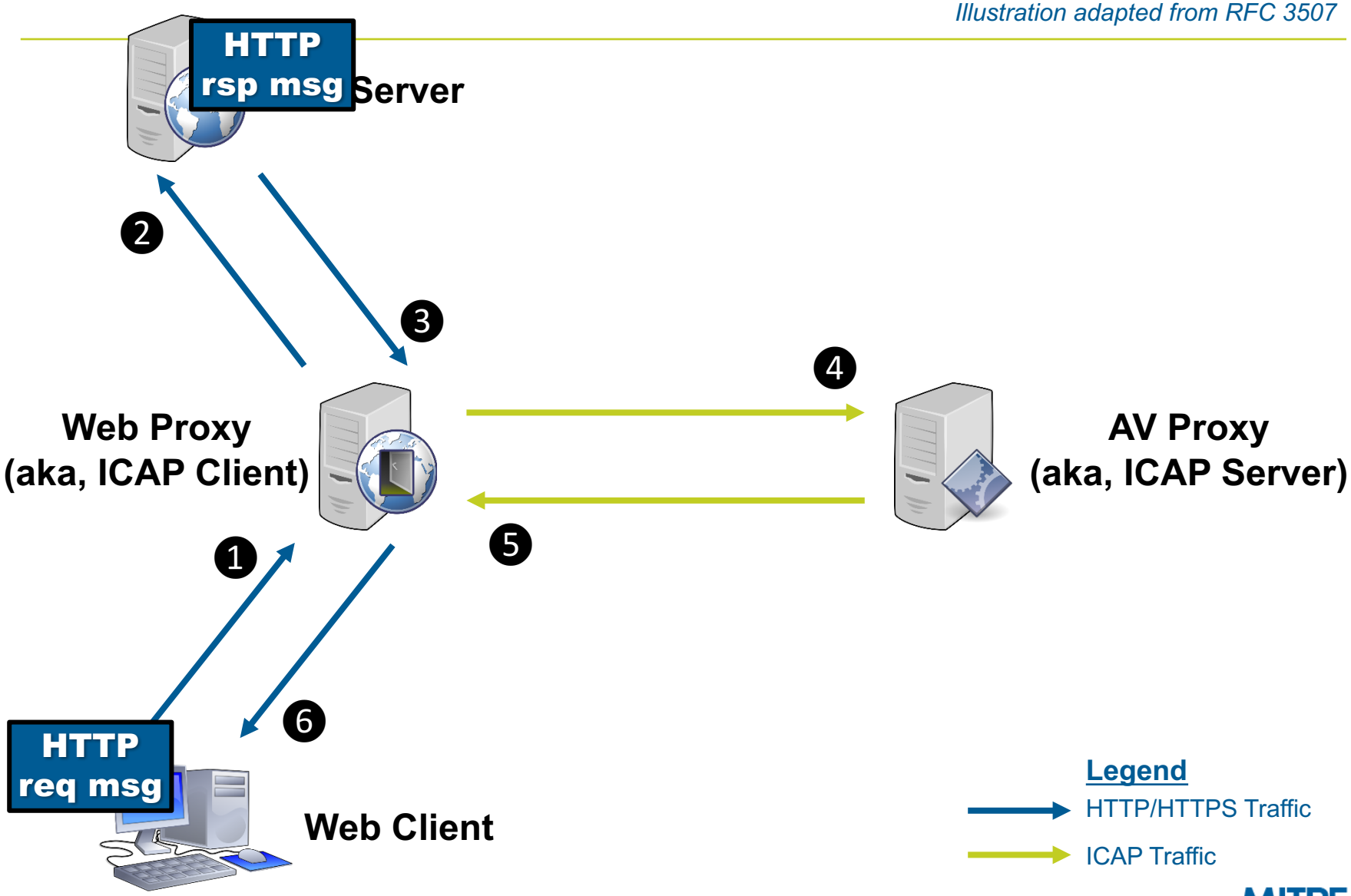
Illustration adapted from RFC 3507



MITRE

ICAP Response Modification

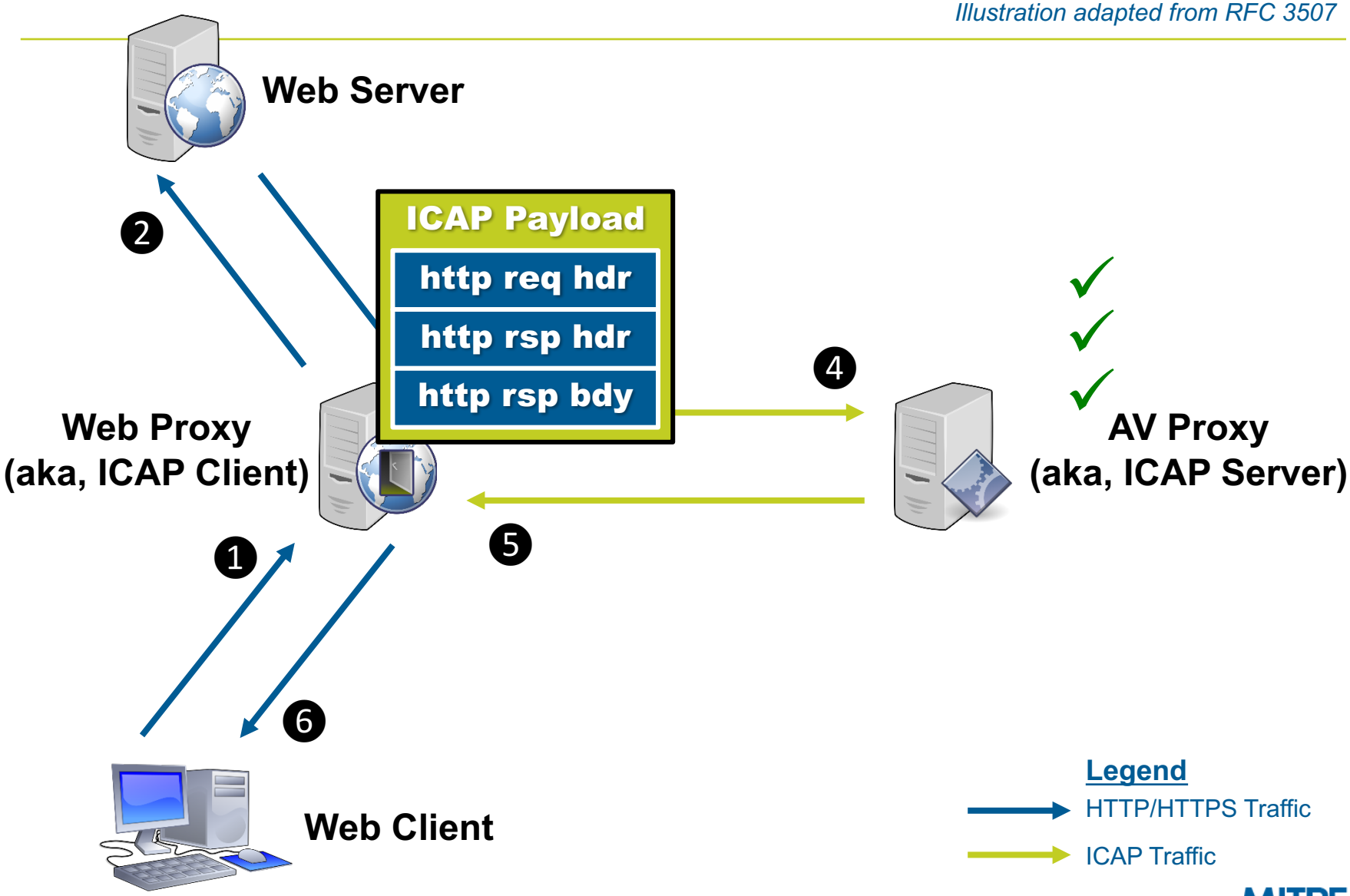
Illustration adapted from RFC 3507



MITRE

ICAP Response Modification

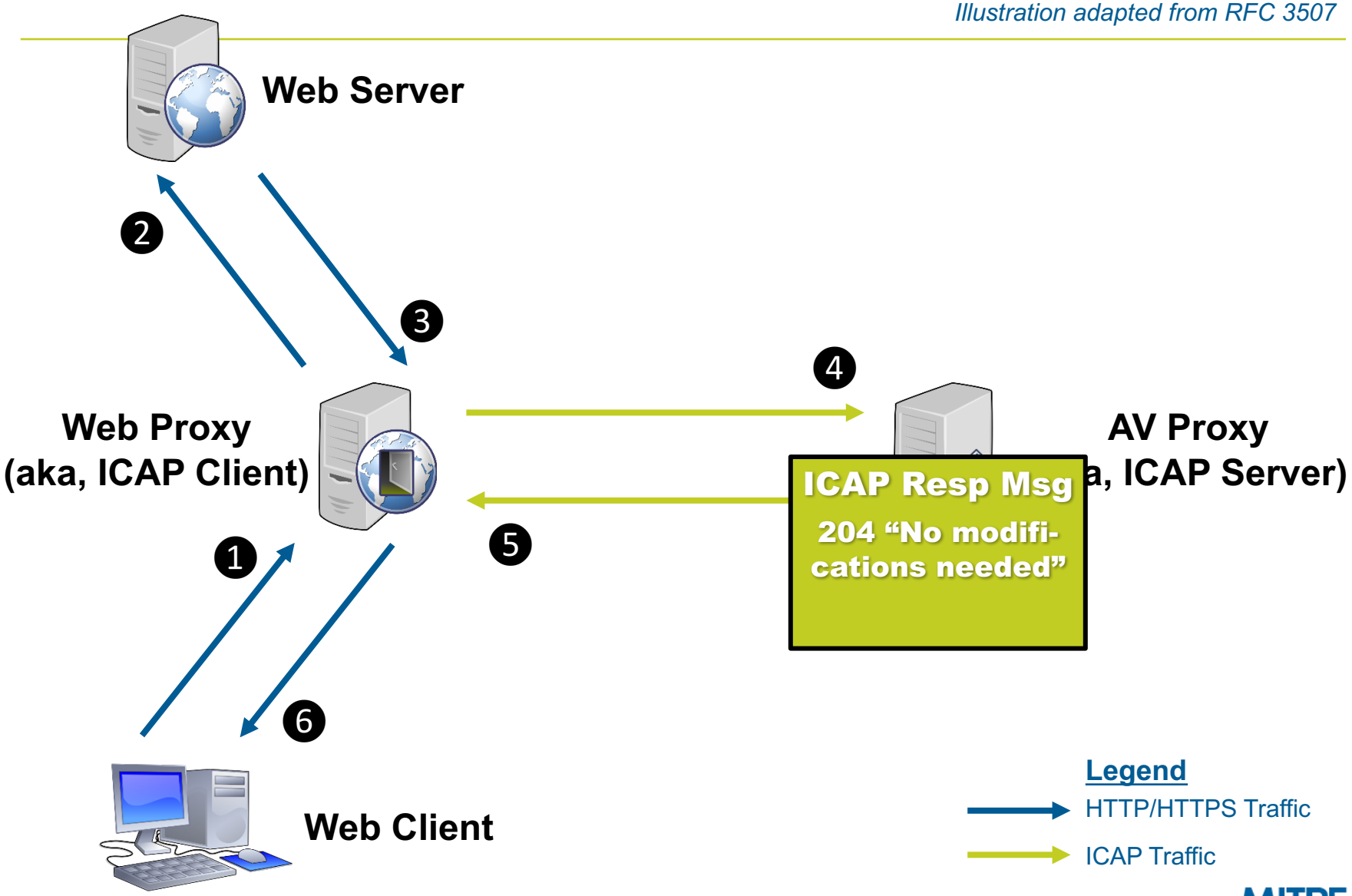
Illustration adapted from RFC 3507



MITRE

ICAP Response Modification

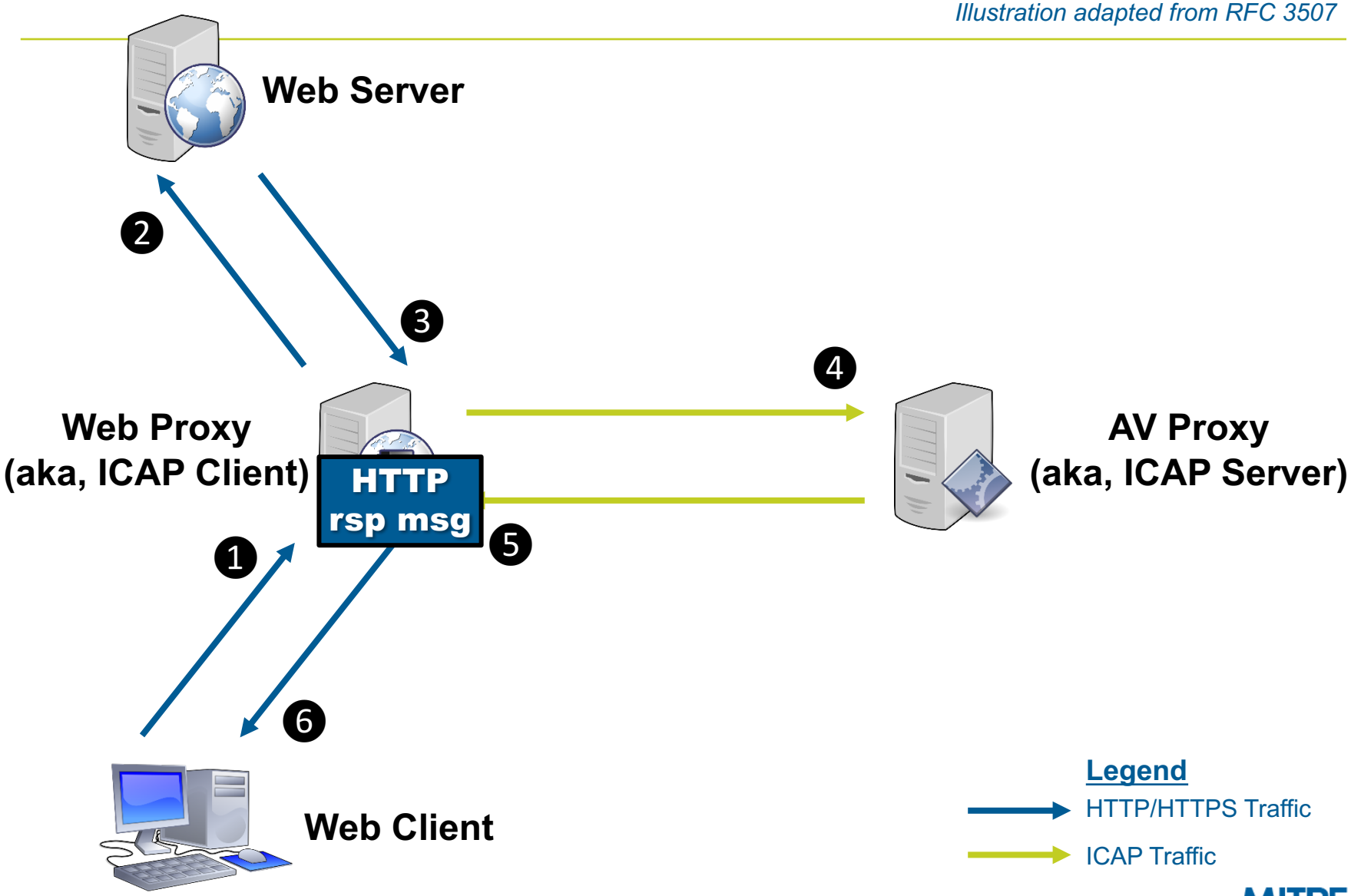
Illustration adapted from RFC 3507



MITRE

ICAP Response Modification

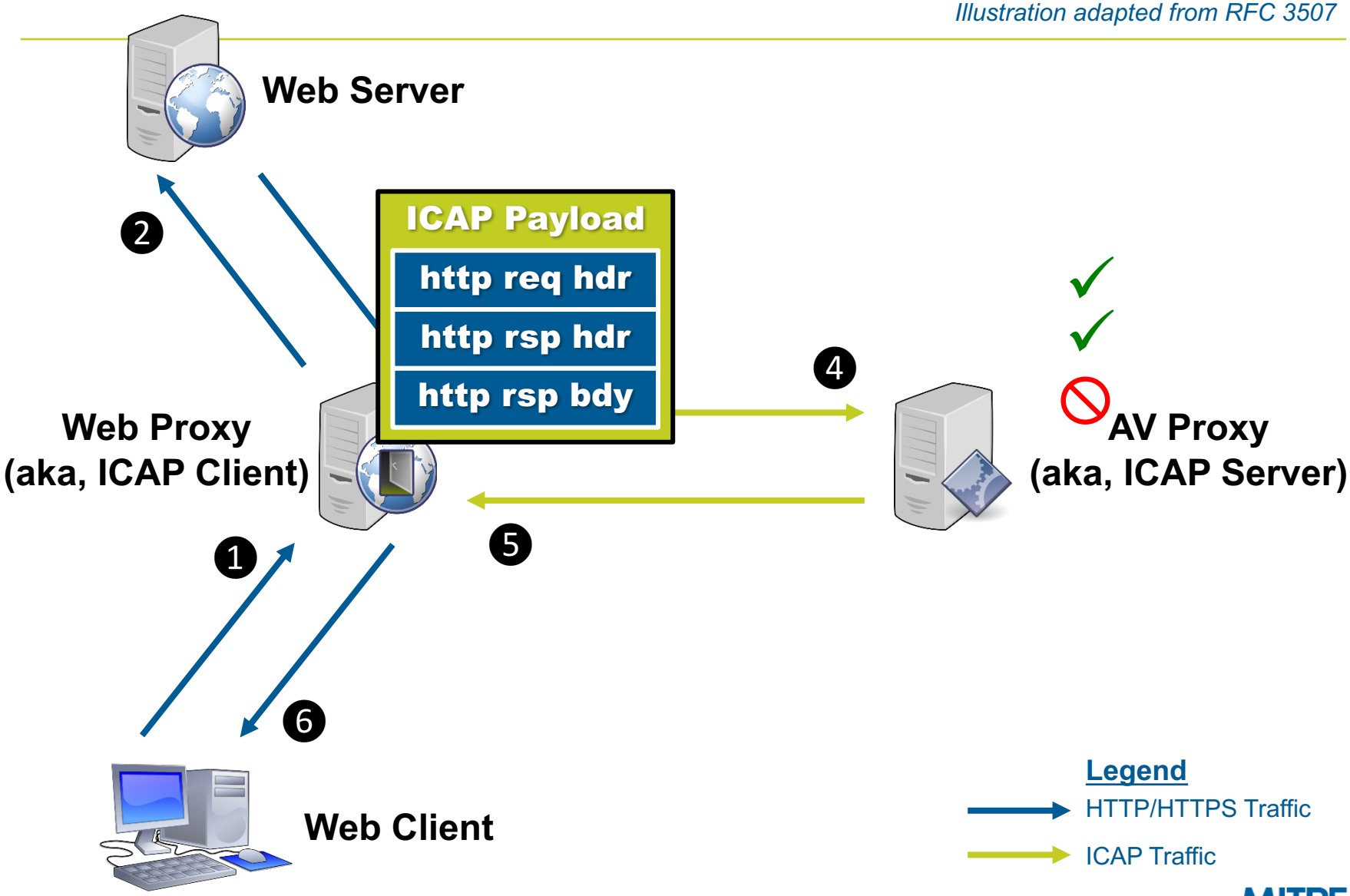
Illustration adapted from RFC 3507



MITRE

ICAP Response Modification

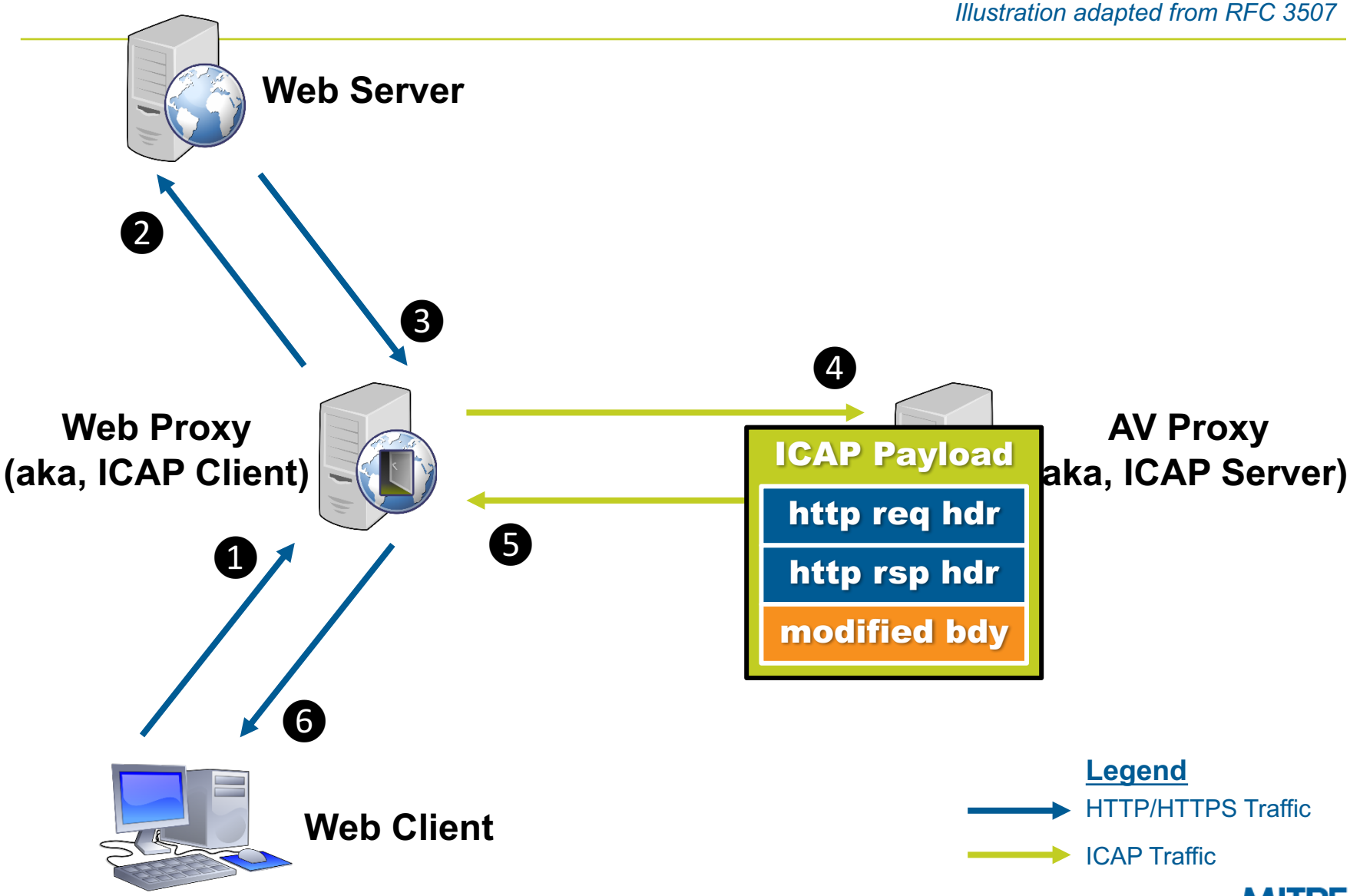
Illustration adapted from RFC 3507



MITRE

ICAP Response Modification

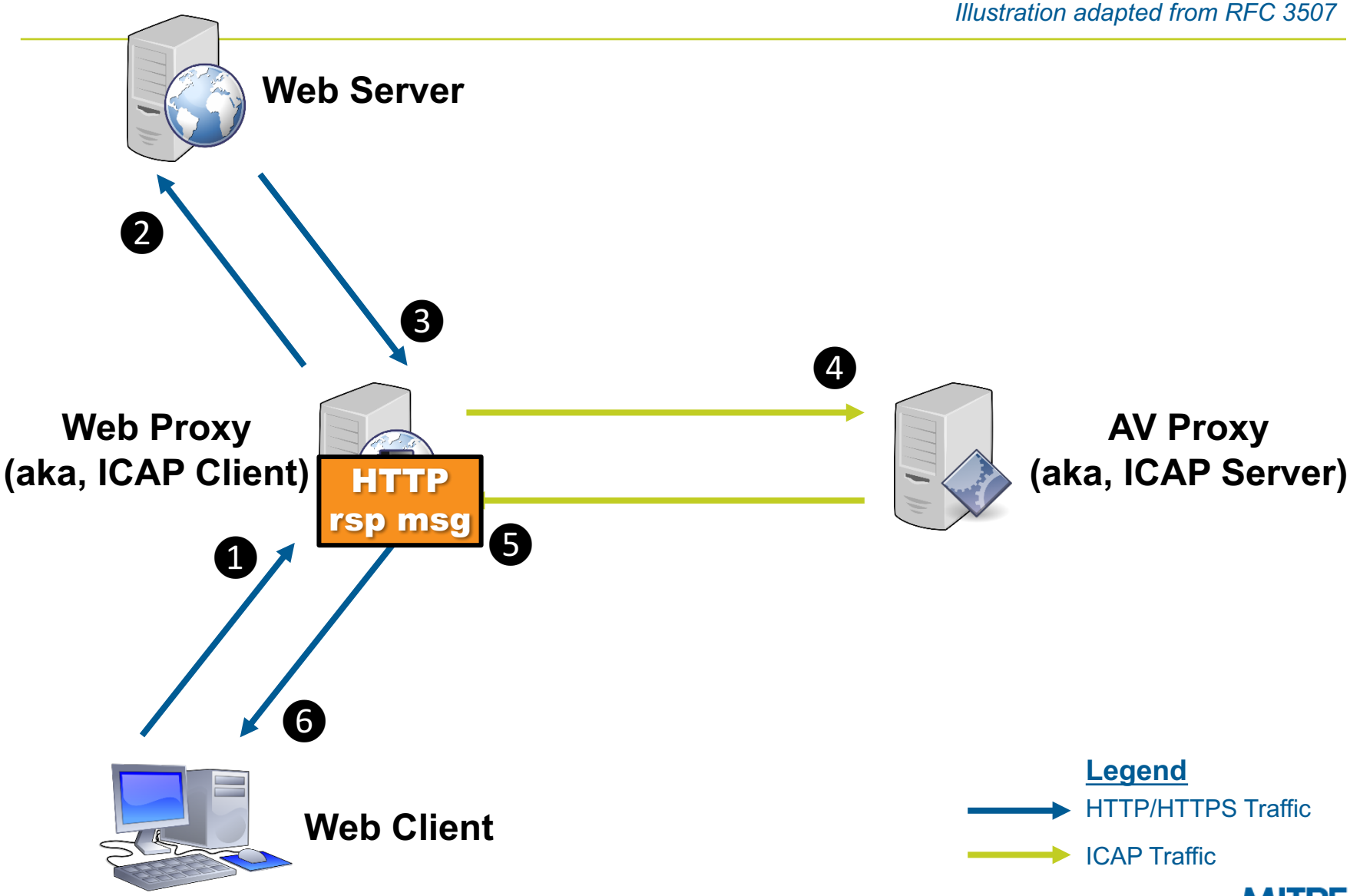
Illustration adapted from RFC 3507



MITRE

ICAP Response Modification

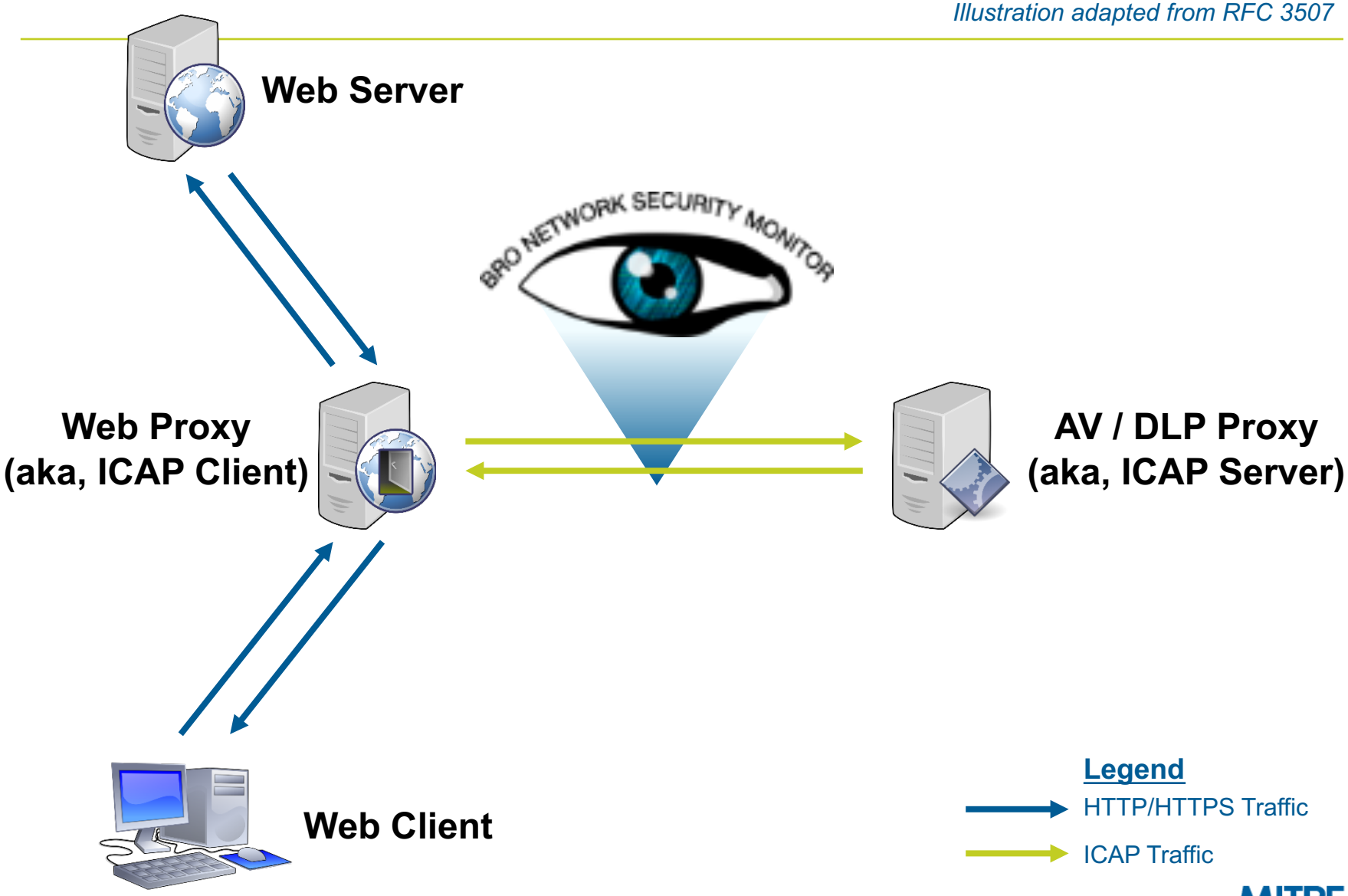
Illustration adapted from RFC 3507



MITRE

The Bro ICAP Analyzer

Illustration adapted from RFC 3507



MITRE

ICAP References

[1] Internet Content Adaptation Protocol (ICAP)

- Request for Comments (RFC) 3507
- J. Elson & A. Cerpa (2003 April)
  - https://tools.ietf.org/html/rfc3507

[2] ICAP Extensions

- IETF Draft
- M. Stecher et al. (2003 April)
  - https://tools.ietf.org/html/draft-stecher-icap-subid-00

[3] ICAP Partial Content Extension

- Draft (unofficial)
- M. Stecher & A. Rousskov (2010 May)
  - http://www.icap-forum.org/documents/specification/draft-icap-ext-partial-content-07.txt

ICAP References – *cont.*

[4] ICAP Errata

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

MITRE

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 *

** LOG defined in [2] ICAP Extensions*

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

Analyzing ICAP Headers

Headers defined in [1] RFC 3507

ICAP Request Headers	ICAP Response Headers	ICAP Options Headers
Allow		Allow
Authorization		
Cache-Control	Cache-Control	
Connected	Connected	
Date	Date	Date
Encapsulated	Encapsulated	Encapsulated
Expires	Expires	
From		
Host	ISTag	ISTag
		Max-Connections
		Methods
		Opt-body-type
		Options-TTL
Pragma	Pragma	
Preview		Preview
Referer	Server	Service
		Service-ID
Trailer	Trailer	Transfer-Preview
Upgrade	Upgrade	Transfer-Ignore
User-Agent		Transfer-Complete

Analyzing Extended Headers

ICAP Request Extensions	ICAP Response Extensions	ICAP Options Extensions
X-Authenticated-User	X-ICAP-Profile	X-Include
X-Authenticated-Groups	X-Attribute	Attribute-List resp body
X-Client-IP	X-Attribute-Cacheability	
X-Server-IP	X-Attribute-Prefix	
X-Subscriber-ID	X-Infection-Found	
X-LOG-[service-ID]	X-Violations-Found	
	X-Virus-ID	
New Method: LOG	X-Response-Info	
LOG-[service-ID]	X-Response-Desc	

Headers defined in [2] ICAP Extensions

ICAP Request Extensions	ICAP Response Extensions	ICAP Options Extensions
x-patience	use-original-body	'Allow' header allowed
'Allow: 206'		
'Allow: NNN', where NNN can be any token		

Headers defined in [3] ICAP Partial Content Extension

MITRE

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
        ISTag
        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
```


MITRE

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

Build Files	Description
❖ *BIF Files*	*build/src/analyzer/protocol/icap/*
events.bif.cc events.bif.h events.bif.init.cc events.bif.register.cc	Auto-generated by BIF compiler and moved into the Bro build tree.
❖ *BinPAC Files*	*build/src/analyzer/protocol/icap/*
icap_pac.cc Icap_pac.h	Auto-generated by BinPAC compiler and moved into the Bro build tree.

MITRE

Bro ICAP Events & Weird Log

Bro ICAP Events	Description
icap_request_line	Generated after REQUEST LINE is parsed
icap_response_line	Generated after RESPONSE LINE is parsed
icap_header	Generated after HEADER field is parsed
icap_options	Generated after OPTIONS BODY is parsed
icap_body_weird	Generated if unexpected BODY format encountered
icap_chunk_weird	Generated if sum of chunks not equal to HTTP 'content-length'
icap_error	Generated for errors when decoding ICAP Requests & Responses
icap_done	Generated after a complete ICAP transaction: ❖ ICAP Request followed by ICAP Response; and ❖ After invoking HTTP analyzer.

Bro ICAP Weird	Description
Unrecognized ICAP Methods	ICAP_WEIRD: unknown ICAP method <string>
Unrecognized ICAP Versions	ICAP_WEIRD: unknown ICAP version <string>
Unrecognized ICAP Status Codes	ICAP_WEIRD: unknown ICAP status code <string>
Unrecognized ICAP Header Names	ICAP_WEIRD: header: <string_1> : <string_2> :: method : <string_3> : is_orig : <string_4>
Unrecognized ICAP Body Format	ICAP_WEIRD: unknown ICAP body format <string_1> :: method : <string_2> : is_orig : <string_3>

MITRE

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
}
```


MITRE

BinPAC Files: icap-protocol.pac

icap-protocol.pac – *cont.*

```
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);
};
```

BinPAC Files: icap-analyzer.pac & -utils.pac

icap-analyzer.pac		
proc_icap_request_line()	Event Generation: HTTP Injection:	icap_request_line none
proc_icap_response_line()	Event Generation: HTTP Injection:	icap_response_line none
proc_icap_header()	Event Generation: HTTP Injection:	icap_header none
proc_icap_body_xxx()	Event Generation: HTTP Injection:	none proc_http_invoke_analyzer
proc_icap_options()	Event Generation: HTTP Injection:	icap_options none
proc_icap_pdu()	Event Generation: HTTP Injection:	icap_done none

icap-analyzer-utils.pac		
get_icap_body_type_from_encap_hdr()	Event Generation: HTTP Injection:	icap_body_weird none

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

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

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


MITRE

BinPAC Files: Bugs & Challenges

Compiler Error in icap_pac.cc

icap.pac	Workaround
let body_ : int=BODY_TYPE_NONE	❖ Added global variable ‘body_’ within icap.pac ❖ Needed to fix a C-compiler error in 'icap_pac.cc :: ICAP_Message::ParseBuffer'. ❖ BinPAC compiler created the variable 'body_' in icap_pac.cc file but never defined it in icap_pac.cc or icap_pac.h.

BinPAC Files: Bugs & Challenges

ParseBuffer() Fails to Parse ‘chunk_data’

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

```
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):

```
TEST_Chunk {
    TEST_Chunk_Size
    TEST_Chunk_Data
}
```

```
TEST_Chunk_Size {
    bytestring
}
```

```
TEST_Chunk_Data {
    bytestring &length =
    TEST_Chunk_Size;
}
```

Caveats & Limitations

Operational Testing

- Biased toward RESPMOD transactions
- REQMOD not yet observed/tested
- OPTIONS & Preview headers ignored

Bro Connection 4-tuples & Identifiers

- IP addresses derived from ICAP extended headers
- TCP port is always 1344
- Connection IDs overlap multiple unrelated user-sessions per ICAP session

Caveats & Limitations

- **REQMOD vs RESPMOD**
  - REQMOD yields HTTP request body
  - RESPMOD 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?

MITRE

Back-Up Slides

MITRE