Protocol Parser Development to Inspect Malware Command and Control (C2) Traffic

Mark Fernandez
Kevin McMahon

BroCon 2017
Sep 12-14, 2017 I NCSA I Urbana, IL

MITRE

Outline

- **Background**
  - Spicy Example (TFTP)
- **Spicy Parsers for Malware C2**
  - Turkojan
  - Gh0st
- **Issues Encountered**
- **Recommendations for Future Work**

Background

- **Bro Protocol Parsing Development**
  - Manual – hand coded
  - BinPac [1] – simplified syntactic specification
  - BinPac++/Spicy [2] – simplified syntactic and semantic specification
- **Standalone Tool**
  - Easily integrated with Bro, Wireshark, etc.
- **Bro is becoming more widely used**
- **Existing Parser Tools…**
  - Based on generic high-level language implementations

Spicy Parser Example – TFTP [3]

```
module TFTP;
import Spicy;

export type Message = unit {
    op: uint<16>;

    switch ( self.op ) {
        1 -> rrq: Request;
        2 -> wrq: Request;
        3 -> data: Data;
        4 -> ack: Ack;
        };

    on %done {
        print self;
        }
};
```

```
type Request = unit {
    fname: bytes &until(b"\x00");
    mode: bytes &until(b"\x00");
};

type Data = unit {
    num: uint<16>;
    data: bytes &eod;
    };

type Ack = unit {
    num: uint<16>;
    };
```


TFTP example adapted from [3] BinPAC++ Demo at BroCon 2014

Spicy Parser Example – TFTP [3]

```
module TFTP;
import Spicy;

export type Message = unit {
    op: uint<16>;

    switch ( self.op ) {
        1 -> rrq: Request;
        2 -> wrq: Request;
        3 -> data: Data;
        4 -> ack: Ack;
        };

    on %done {
        print self;
        }
};
```

```
type Request = unit {
    fname: bytes &until(b"\x00");
    mode: bytes &until(b"\x00");
};

type Data = unit {
    num: uint<16>;
    data: bytes &eod;
    };

type Ack = unit {
    num: uint<16>;
    };
```


TFTP example adapted from [3] BinPAC++ Demo at BroCon 2014

Spicy Bro Interface Example – TFTP [3]

```
grammar tftp.spicy;

protocol analyzer TFTP over UDP:
    parse with TFTP::Message,
    port 69/udp;

on TFTP::Message::rrq ->
    event tftp_rrq($conn, self.rrq.fname, self.rrq.mode);

on TFTP::Message::data ->
    event tftp_data($conn, self.data.data);
```

TFTP example adapted from [3] BinPAC++ Demo at BroCon 2014

Bro Script Example – TFTP [3]

```
module TFTP;

event tftp_rrq(c: connection, frame: string, mode: string) {
    print "RRQ", c$id$resp_h, fname, mode;
    }

event tftp_data(c: connection, data: string) {
    print "DATA”, |data|;
    }
```

TFTP example adapted from [3] BinPAC++ Demo at BroCon 2014

Motivation

- Developing and integrating protocol parsers is time consuming
- Spicy significantly reduces development and modification timeframes
- Spicy specifications are more compact and succinct
- Improved shareability?

Objectives

- **Short Term**
  - Leverage Spicy for dissecting malware C2 protocols
    - Tool familiarization
    - Sample protocols of increasing complexity
    - Identify and workaround idiosyncrasies
    - Document findings to enable broader usage
    - Get involved in further development of the tool
- **Longer Term**
  - Develop a library of existing parsers and techniques
    - Classified and Unclassified
  - Develop performance metrics
    - Developer perspective
    - Run time / scalability

Turkojan C2 Protocol

- **Controller Commands**
  - MINFO | BAGLI | BELGE | BROWS | HAYDI | DLF | ULF | SHELL
    - Some include parameter(s) which are rarely well delimited
- **Implant Messages/Responses**
  - ams | trn | MINFO <parameters> | <size> + <response body>

Turkojan C2 Protocol

- **Controller Commands**
  - MINFO | BAGLI | BELGE | BROWS | HAYDI | DLF | ULF | SHELL
    - Some include parameter(s) which are rarely well delimited
- **Implant Messages/Responses**
  - ams | trn | MINFO <parameters> | <size> + <response body>

```
MINFO
    MINFO|alice|172.16.0.132|JON-PC|WinVista|ENU|
```

```
BROWSC:\Users\Jon\Documents
    46 00 00 00 m e t i 0d . 0d 0a . . 0d 0a M y 20 M u s i c ...
```

Turkojan Implant Parsing

```
export type Implant = unit   {
    irequests: list<iData>;
};

type iData = unit  {
    switch  {
        iC1: iStart;
        iC1F: ifStart;
        iC2: iMinfo;
        iC3: iHeartbeat;
        iC4: iCntData;
    };
};

type iStart = unit {
        iS: b"ams";
};
```

```
type iMinfo = unit {
        iM: b"MINFO|";
        iUser: bytes &until(b"\x7c");
        iIP: bytes &until(b"\x7c");
        iHost: bytes &until(b"\x7c");
        iOS: bytes &until(b"\x7c");
        iUnk: bytes &until(b"\x7c");
};

type iHeartbeat = unit {
        iHB: b"BAGLANTI?";
};

type iCntData = unit {
        iLen: uint32 &byteorder =
                        Spicy::ByteOrder::Little;
        iResp: bytes &length=self.iLen;
};
```

Turkojan Implant Parsing

```
export type Implant = unit   {
    irequests: list<iData>;
};

type iData = unit  {
    switch  {
        iC1: iStart;
        iC1F: ifStart;
        iC2: iMinfo;
        iC3: iHeartbeat;
        iC4: iCntData;
    };
};

type iStart = unit {
    iS: b"ams";
};
```

```
type iMinfo = unit {
        iM: b"MINFO|";
        iUser: bytes &until(b"\x7c");
        iIP: bytes &until(b"\x7c");
        iHost: bytes &until(b"\x7c");
        iOS: bytes &until(b"\x7c");
        iUnk: bytes &until(b"\x7c");
};

type iHeartbeat = unit {
    iHB: b"BAGLANTI?";
};

type iCntData = unit {
        iLen: uint32 &byteorder =
                    Spicy::ByteOrder::Little;
        iResp: bytes &length=self.iLen;
};
```

Turokan Controller Parsing

```
export type Controller = unit {
    cdata: list<cData>;
};

type cData = unit {
    a: bytes &chunked -> self.data;

    var data : sink;

    on %init {
        self.data.connect(new cCmd);
    }
};

export type cCmd = unit {
    cmd: bytes &chunked;
};
```

Turokan Controller Parsing

```
export type Controller = unit {
    cdata: list<cData>;
};

type cData = unit {
    a: bytes &chunked -> self.data;

    var data : sink;

    on %init {
        self.data.connect(new cCmd);
    }
};

export type cCmd = unit {
    cmd: bytes &chunked;
};
```

Turkojan Parser Output

```
01/01/1970 00:09:25: SHELL|372|dir
 Volume in drive C has no label.
 Volume Serial Number is 0075-80C6

 Directory of C:\Users\Jon\Downloads

07/17/2015  01:18 PM    <DIR>          .
07/17/2015  01:18 PM    <DIR>          ..
07/22/2015  08:38 AM    <DIR>          share
               0 File(s)              0 bytes
               3 Dir(s)  56,338,280,448 bytes free

C:\Users\Jon\Downloads>

01/01/1970 00:09:45: SHELL|netstat -an

01/01/1970 00:09:45: SHELL|2391|netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING…
```

Gh0st Malware

- **History**
  - Origins, circa 2001[4]
  - Popular “Off-The-Shelf” Remote Access Tool (RAT)
  - Used in Numerous High-Profile Cases
  - Still in Use Today [5]

- **Multi-Stage Malware**
  - Delivery → Exploitation → Dropper → Implant → Beacon to C2 Controller → Full C2
    - Command & Control (C2) Channel
    - Execution Channel

[4] McAfee whitepaper, Anatomy of Gh0st RAT, dated 2012
[5] FireEye article, EternalBlue Exploit Delivers Non-WannaCry Payloads, dated 02JUN2017

Gh0st Malware

- **History**
  - Origins, circa 2001[4]
  - Popular “Off-The-Shelf” Remote Access Tool (RAT)
  - Used in Numerous High-Profile Cases
  - Still in Use Today [5]

- **Multi-Stage Malware**
  - Delivery → Exploitation → Dropper → Implant → Beacon to C2 Controller → Full C2
    - Command & Control (C2) Channel
    - Execution Channel

[4] McAfee whitepaper, Anatomy of Gh0st RAT, dated 2012
[5] FireEye article, EternalBlue Exploit Delivers Non-WannaCry Payloads, dated 02JUN2017

Gh0st Malware

Command OpCodes (50) :: Controller → Implant [4]		
00 Actived	11 Screen Reset	22. Audio
01 List Drive	12 Algorithm Reset	23. System
02 List Files	13 Screen Ctl Alt Del	24 Ps List
03 Down Files	14 Screen Control	25 Ws List
04 File Size	15 Screen Block Input	26. Dialupass
05 File Data	16 Screen Blank	27 Kill Process
06 Exception	17 Screen Capture Layer	28 Shell
07 Continue	18 Screen Get Clipboard	29 Session
08 Stop	19 Screen Set Clipboard	2A Remove
09 Delete File	1A Webcam	2B Down Exec
0A Delete Directory	1B Webcam Enable Compress	2C Update Server
0B Set Xfer Mode	1C Webcam Disable Compress	2D Clean Event
0C Create Directory	1D Webcam Resize	2E Open URL Hide
0D Rename File	1E Next	2F Open URL Show
0E Open File Show	1F Keyboard	30 Rename Remark
0F Open File Hide	20 Keyboard Offline	31 Replay Heartbeat
10 Screen Spy	21 Keyboard Clear	

Token OpCodes (29) :: Implant → Controller [4]	
64 Auth	75 Next Screen
65 Heartbeat	76 Clipboard Text
66 Login	77 Webcam Bitmap Info
67 Drive List	78 Webcam Dib
68 File List	79 Audio Start
69 File Size	7A Audio Data
6A File Data	7B Keyboard Start
6B Xfer Finish	7C Keyboard Data
6C Delete Finish	7D Ps List
6D Get Xfer Mode	7E Ws List
6E Get File Data	7F Dialupass
6F Create Dir Finish	80 Shell Start
70 Data Continue	
71 Rename Finish	
72 Exception	
73 Bitmap Info	
74 First Screen	

Gh0st Command OpCodes & Token OpCodes defined in [4] Anatomy of a Gh0st RAT

Gh0st Malware

Command OpCodes (50) :: Controller → Implant [4]		
00 Actived	11 Screen Reset	Audio
01 List Drive	12 Algorithm Reset	23. System
02 List Files	13 Screen Ctl Alt Del	24 Ps List
03 Down Files	14 Screen Control	25 Ws List
04 File Size	15 Screen Block Input	26. Dialupass
05 File Data	16 Screen Blank	27 Kill Process
06 Exception	Screen Capture Layer	Shell
07 Continue	18 Screen Get Clipboard	29 Session
08 Stop	19 Screen Set Clipboard	2A Remove
09 Delete File	Webcam	2B Down Exec
0A Delete Directory	1B Webcam Enable Compress	2C Update Server
0B Set Xfer Mode	1C Webcam Disable Compress	2D Clean Event
0C Create Directory	1D Webcam Resize	2E Open URL Hide
0D Rename File	1E Next	2F Open URL Show
0E Open File Shov	Keyboard	30 Rename Remark
0F Open File Hide	20 Keyboard Offline	31 Replay Heartbeat
Screen Spy	21 Keyboard Clear	

Token OpCodes (29) :: Implant → Controller [4]	
64 Auth	75 Next Screen
65 Heartbeat	Clipboard Text
66 Login	Webcam Bitmap Info
67 Drive List	78 Webcam Dib
68 File List	Audio Start
69 File Size	7A Audio Data
6A File Data	Keyboard Start
6B Xfer Finish	7C Keyboard Data
6C Delete Finish	7D Ps List
6D Get Xfer Mode	7E Ws List
6E Get File Data	7F Dialupass
6F Create Dir Finish	Shell Start
70 Data Continue	
71 Rename Finish	
72 Exception	
Bitmap Info	
74 First Screen	

Gh0st Command OpCodes & Token OpCodes defined in [4] Anatomy of a Gh0st RAT

Gh0st Command & Control (C2) Protocol

- **Original Gh0st Message Format**
  - Header
    - <name> Character string '`Gh0st`' [5-byte field]
    - <size1> Size of Message: <header> + <body> [4-byte field]
    - <size2> Size of <body> after de-obfuscation [4-byte field]
  - Body
    - <opcode> 79 different values [1-byte field]
    - <data> Optional / OpCode-dependent
  - Obfuscation
    - <header> Plain text
    - <body> ZLIB compressed

Gh0st Command & Control (C2) Protocol

- **Original Gh0st Message Format**
  - Header
    - <name> Character string 'Gh0st' [5-byte field]
    - <size1> Size of Message: <header> + <body> [4-byte field]
    - <size2> Size of <body> after de-obfuscation [4-byte field]
  - Body
    - <opcode> 79 different values [1-byte field]
    - <data> Optional / OpCode-dependent
  - Obfuscation
    - <header> Plain text
    - <body> ZLIB compressed

```
Bytes:  00 01 02 03 04 | 05 06 07 08 | 09 0A 0B 0C | 0D 0E 0F
Field:  <name>         | <size1>     | <size2>     | <body>...
```

Gh0st C2 Protocol Parsing w/Spicy

- **Original Gh0st Message Format**
  - Spicy File: gh0st.spicy

```
export type Message = unit {
    %byteorder = Spicy::ByteOrder::Little;
    name     : bytes &length=5;
    size1    : uint32;
    size2    : uint32;
    body     : bytes &length=self.body_len;

    var body_len : uint32;

    on self.size1 {
        self.body_len = self.size1 - 13;
    }
};
```

  - Event File: gh0st.evt

```
grammar gh0st.spicy;

protocol analyzer GH0ST over TCP:
    parse with GH0ST::Message;
#   ports {21/tcp, 80/tcp}; # Use DPD instead

on GH0ST::Message -> event gh0st_message (
    $conn,
    $is_orig,
    self.name,
    self.size1,
    self.size2,
    self.body_len,
    self.body
);
```

Gh0st C2 Protocol Parsing w/Spicy

- **Original Gh0st Message Format**
  - Spicy File: gh0st.spicy

```
export type Message = unit {
    %byteorder = Spicy::ByteOrder::Little;
    name    : bytes &length=5;
    size1   : uint32;
    size2   : uint32;
    body    : bytes &length=self.body_len;

    var body_len : uint32;

    on self.size1 {
        self.body_len = self.size1 - 13;
    }
};
```

  - Event File: gh0st.evt

```
grammar gh0st.spicy;

protocol analyzer GH0ST over TCP:
    parse with GH0ST::Message;
    #   ports {21/tcp, 80/tcp}; # Use DPD instead

on GH0ST::Message -> event gh0st_message (
        $conn,
        $is_orig,
        self.name,
        self.size1,
        self.size2,
        self.body_len,
        self.body
    );
```

Gh0st C2 Protocol Parsing w/Spicy

- **Original Gh0st Message Format**
  - Spicy File: gh0st.spicy

```
export type Message = unit {
    %byteorder = Spicy::ByteOrder::Little;
    name     : bytes &length=5;
    size1    : uint32;
    size2    : uint32;
    body     : bytes &length=self.body_len;

    var body_len : uint32;

    on self.size1 {
        self.body_len = self.size1 - 13;
    }
};
```

  - Event File: gh0st.evt

```
grammar gh0st.spicy;

protocol analyzer GH0ST over TCP:
    parse with GH0ST::Message;

on GH0ST::Message::size2 ->
event gh0st_header (
    $conn, $is_orig, self.name, self.size1,
    self.size2
);

on  GH0ST::Message::body ->
event gh0st_body (
    $conn, $is_orig, self.body_len, self.body
);
```

Gh0st C2 Protocol Parsing w/Spicy

- **Original Gh0st Message Format**
  - Spicy File: gh0st.spicy

```
export type Message = unit {
    %byteorder = Spicy::ByteOrder::Little;
    name    : bytes &length=5;
    size1   : uint32;
    size2   : uint32;
    body    : bytes &length=self.body_len;

    var body_len : uint32;

    on self.size1 {
        self.body_len = self.size1 - 13;
    }
};
```

  - Event File: gh0st.evt

```
grammar gh0st.spicy;

protocol analyzer GH0ST over TCP:
    parse with GH0ST::Message;

on GH0ST::Message::size2 ->
event gh0st_header (
    $conn, $is_orig, self.name, self.size1,
    self.size2
);

on  GH0ST::Message::body ->
event gh0st_body (
    $conn, $is_orig, self.body_len, self.body
);
```

Spicy Filter Usage

```
export type BodyUnzip = unit (msg:Message)
{
        body                    : bytes &length=msg.size1 -> self.data_sink;

        var data_sink           : sink;
        var body_plain          : BodyPlain;
        var body_len            : uint32;

        on %init {
                self.add_filter(Spicy::Filter::ZLIB);
        }

        on body {
                # Get size of 'body' after unzipping
                self.body_len = cast<uint32>(|self.body|);

                # Create Sink
                self.body_plain = new BodyPlain(msg);
                self.data_sink.connect(self.body_plain);
                self.data_sink.write(self.body);
        }
};
```

Gh0st C2 Protocol Parsing w/Spicy

- **Parsing Original Gh0st Message Format**
  - Easy & Straightforward
  - Only twelve (12) lines of Spicy code

- **Parsing Gh0st Variants**
  - Header <name>
    - Dozens of variants [6]
    - Character Strings > 5-bytes and Non-terminated
  - Body Obfuscation
    - Either ZLIB Compression
    - Or Plain Text [7]
  - Spicy Code Becomes Complex…

Variant Gh0st Names defined in [6] The Many Faces of Gh0st RAT

Gh0st C2 Protocol Parsing w/Spicy

- **Parsing Original Gh0st Message Format**
  - Easy & Straightforward
  - Only twelve (12) lines of Spicy code
- **Parsing Gh0st Variants**
  - Header <name>
    - Dozens of variants [6]
    - Character Strings > 5-bytes and Non-terminated
  - Body Obfuscation
    - Either ZLIB Compression
    - Or Plain Text [7]
  - Spicy Code Becomes Complex…

Variant Gh0st Names [6]			
00000	FWAPR	Lover	Tyjhu
7hero	FWKJG	LUCKK	URATU
ABCDE	Gh0st	LURK0	v2010
Adobe	Gi0st	lvxYT	VGTLS
ag0ft	GM110	LYRAT	W0LFKO
apach	GOLDt	Lyyyy	Wangz
Assas	GWRAT	MoZhe	wcker
attac	HEART	MYFYB	Wh0vt
B1X6Z	Heart	MyRat	whmhl
BEiLa	Hello	Naver	Winds
BEiJi	HTTPS	NIGHT	wings
Blues	https	NoNul	World
ByShe	httpx	Origi	X6M9K
cblst	HXWAN	OXXMM	X6RAT
chevr	IM007	PCRat	XDAPR
CHINA	ITore	QQ_124971919	xhjyk
cy122	kaGni	QWPOT	Xjjhj
DrAgOn	KOBBX	Snown	xqwf7
EXXMM	KrisR	Spidern	YANGZ
Eyesl	Level	SocKt	
FKJP3	light	Super	
FLYNN	LkxCq	Sw@rd	

Variant Gh0st Names defined in [6] The Many Faces of Gh0st RAT

Gh0st C2 Protocol Parsing w/Spicy

- **Using Bro's File Extraction Framework**
  - Gh0st OpCodes with Large Data Outputs
    - Audio Capture, Clipboard Capture, Screen Capture, Webcam Capture
    - File Transfer, Command Shell
  - Design Decision
    - Either Ignore the Data…
    - Or Use File Extraction Framework to Write to Disk…
- **Parsing in Spicy-land or Script-land**
  - Gh0st OpCode with Formatted Data
    - Login, List Drives, List Files, List Processes
  - Design Decision
    - Either Parse in Spicy-land…
    - Or Parse in Script-land (Bro script, Bash, Python, etc)…

Gh0st Summary

- **Spicy Parser**
  - Very Simple
  - Twelve Lines of Spicy Code
  - Becomes Complex with Gh0st Variants

Issues Encountered

- **Limited documentation**
  - As expected with an early beta-version of the tool
- **Public traffic samples were woefully incomplete**
  - Enlisted support of other MITRE staff to generate traffic
- **Somewhat steep learning curve involved in making changes to the tool itself**
- **Some Lack of Features, A few bugs, anomalies & peculiarities encountered**
  - Working on public release for decode_hex, rc4_{encrypt | decrypt}
- **Adversaries don't know how to develop protocols**
  - Which requires a variety of work-around strategies

Recommendations for Future Work

- Develop Parsers for More Malware
- Add File Analysis Framework for Gh0st Parser
- Contribute to HILTI*/Spicy Development Team to Fix & Document Issues
- Complete Primer Documentation

- What else? Suggestions?

* HILTI – High-level Intermediary Language for Traffic Inspection

Acknowledgments

- **Malware Packet Capture Files**
  - Frank Posluszny | MITRE Corp.
  - Nick Beede | MITRE Corp.

  - Mila Parkour | Contagio Malware Dump

References

[1] **binpac: A yacc for Writing Application Protocol Parsers**
Vern Paxson, ICSI/LBNL, Ruoming Pang, Princeton; Robin Sommer, ICSI/LBNL; Larry Peterson, Princeton
http://www.icsi.berkeley.edu/pubs/networking/binpac/IMC06pdf

[2] **Spicy: A Unified Deep Packet Inspection Framework for Safely Dissecting All Your Data**
Robin Sommer, ICSI/LBNL; Johanna Amann, ICSI; Seth Hall, ICIS/LBNL
http://www.icir.org/robin/papers/acsac16-spicy.pdf

[3] **BinPAC++ Demo: A Next Generation Parser Generator**
Robin Sommer, ICSI/LBNL
Slides: https://www.bro.org/brocon2014/brocon2014_sommer_binpac.pdf | Video: https://www.youtube.com/watch?v=3sQ6thi_BR0

[4] **Know Your Digital Enemy: Anatomy of a Gh0st RAT**
Michael G. Spohn, McAfee Foundstone Professional Services
http://docplayer.net/176479-Know-your-digital-enemy.html

[5] **Threat Actors Leverage EternalBlue Exploit to Deliver non-WannaCry Payloads**
Ali Islam, et al, FireEye Threat Research, Advanced Malware
https://www.fireeye.com/blog/threat-research/2017/05/threat-actors-leverage-eternalblue-exploit-to-deliver-non-wannacry-payloads.html

[6] **The Many Faces of Gh0st RAT: Plotting the Connections Between Malware Attacks**
Snorre Fagerland, Norman ASA
http://download01.norman.no/documents/ThemanyfacesofGh0stRat.pdf

[7] **Gh0st Packet Capture**
MilaParkour, Contagio Malware Dump
http://contagiodump.blogspot.com/2013/04/collection-of-pcap-files-from-malware.html

[8] **Hunting and Decrypting Ghost Communications Using Memory Forensics**
Monnappa KA, NullCon Presentation
https://nullcon.net/website/archives/ppt/goa-15/hunting-and-decrypting-ghost-communications-using-memory-forensics-by-monnappa-k-a.pptx

Questions?

Background Slides

Gh0st Command & Control (C2) Protocol

C2 Channel *[Example]*

```
-----------------------------------------------------
Implant      →       C2 Server
-----------------------------------------------------
TCP Stream #1 on Port X
0x66 + data →        | token_login
            ←  0x00  | command_actived
            ←  0x10  | command_screen_spy









            ←  0x28  | command_shell




            ←  0x2A  | command_remove
-----------------------------------------------------
```

Execution Channel(s) *[Example]*

```
-----------------------------------------------------
Implant      →       C2 Server
-----------------------------------------------------



TCP Stream #2 on Port Y
0x73 + data →        | token_bitmap_info
            ←  0x1E  | command_next
0x74 + data →        | token_firstscreen
0x75 + data →        | token_nextscreen
...
0x75 + data →        | token_nextscreen

TCP Stream #3 on Port Z
0x80         →       | token_shell_start
             ←  0x1E | command_next
       data →        | <data>
...

-----------------------------------------------------
```

Gh0st C2 Protocol Parsing w/Spicy

- **Using Regular Expressions in Spicy**
  - Pattern to Check for String [5-16 bytes] + Digits [8 bytes] + ZLIB Header [0x78 0x9C]
    - /[a-zA-z0-9:]{5,16}..\x00\x00..\x00\x00\x78\x9c/[8]
  - Good Starting Point
    - Add '_' and '@' characters
    - Increase to three wildcards in <size1> and <size2> fields
    - Account for plain-text Gh0st body via Token OpCodes
  - Modified RegEx Pattern
    - /^[a-zA-Z0-9:_@]{5,16}…\x00…\x00(\x78\x9C|[\x64-\x80])/
- **Bug in Spicy RegEx Matching**
  - Spicy Method: 'bytes::match()'
  - Issue with Signed vs Unsigned Integers
  - Fails to Match Extended ASCII Characters (0x80 – 0xFF)

RegEx Pattern Adopted from [8] Gh0st Memory Forensics

Gh0st C2 Protocol Parsing w/Spicy

- **Using Regular Expressions in Bro's Dynamic Protocol Detection (DPD) Framework**
  - DPD is External to and Antecedent to Spicy
  - Pattern Matching via DPD Solves:
    - Issue of masquerading over well-known TCP Ports
    - Issue of detecting ZLIB compression or plain text
    - Issue of variable length <name> field
      - Though still an issue within Spicy
  - Solution:
    - One RegEx pattern to match Header + ZLIB
      - Parse with 'GH0ST::MessageUnzip'
    - One RegEx pattern to match Header + Token OpCodes
      - Parse with 'GH0ST::MessagePlain'