

~~vulnerabilities die hard~~



kowsik@musecurity.com

<http://labs.musecurity.com>

i *see* dead protocols

A thick, orange brushstroke underline is positioned below the text, extending across most of the width of the text.

this talk ...



- ✓ is not about mu
- ✓ is not about [just] fuzzing
- ✓ does not contain pictures from matrix

... is about



- ✓ protocols
 - ✓ in the pedantic sense
- ✓ abstractions and patterns
- ✓ string theory and unification
- ✓ laziness, impatience and hubris

what's a protocol anyways?

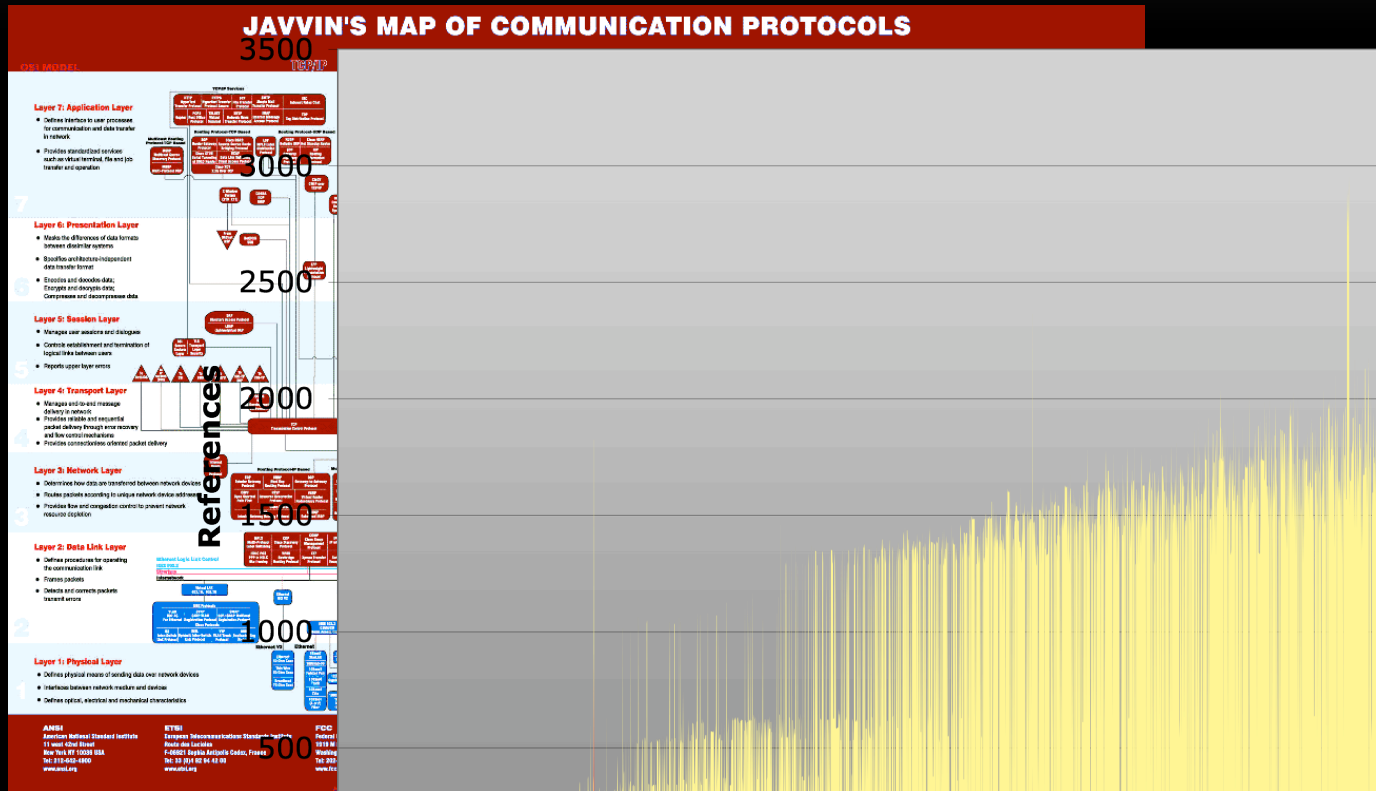
- ✓ rules governing the syntax, semantics, and synchronization of communication
 - ✓ wiki: Protocol (computing)
- ✓ protocol != network
- ✓ protocols represent the attack surface
 - ✓ you are who your interface is
- ✓ only way into the code
 - ✓ that really matters from the *outside*

taxonomy



- ✓ network-based (layers 2 through 7)
- ✓ command line interfaces
 - ✓ psql, argc/argv
- ✓ function calls
 - ✓ Java, IDispatch#invoke, ioctl
- ✓ file formats

kevin bacon



references

rfc's

six degrees of protocols



- ✓ SIP uses LDAP DN's
 - ✓ which use ASN
 - ✓ which are in X.509 certificates
 - ✓ which is used in TLS/SSL
 - ✓ which contains Name/Value pairs
 - ✓ that's used in iCal format
- ✓ DHCP has NetBIOS names
 - ✓ which is used in CIFS
 - ✓ which uses Kerberos
 - ✓ which uses ASN
 - ✓ which ...

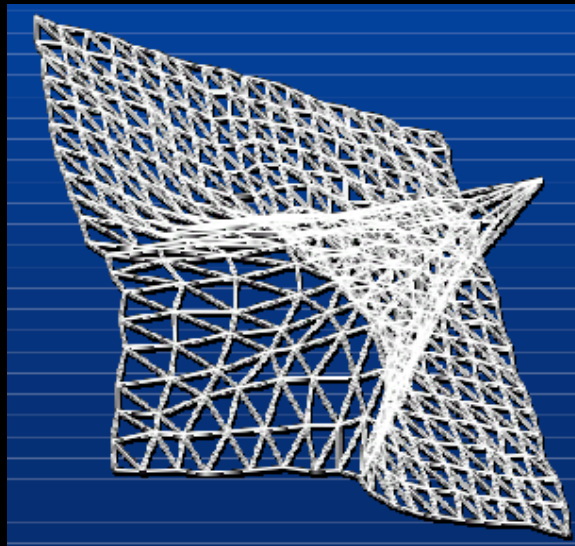
dom's and channel's



- ✓ state, structure, semantics and constraints
 - ✓ a semantic DOM
 - ✓ with associated vulnerability patterns
- ✓ io/delivery mechanism
 - ✓ sockets (raw, v4, v6, tcp, udp, ssl, sctp, ...)
 - ✓ interactive channels (telnet, ssh, console, ...)
 - ✓ bluetooth, wireless, usb, firewire
 - ✓ ioctl's
 - ✓ files

fuzzing

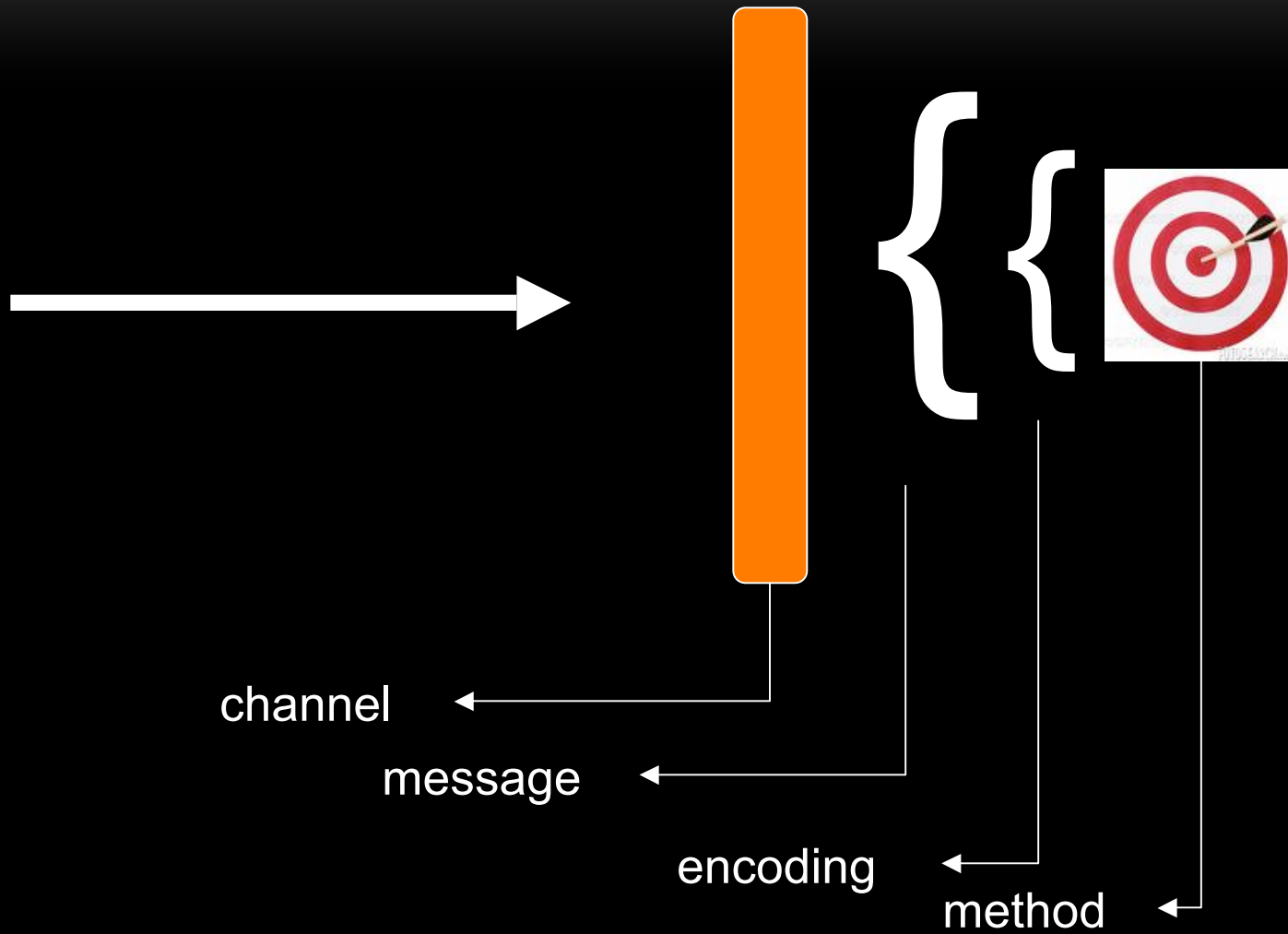
- ✓ is really about semantic data structures
 - ✓ free form deformation
 - ✓ dependency propagation
 - ✓ constraint violation



string is a string is a ...



peeling the onion



peeling the onion



serial	>	dnp::write	>	write_register()
udp	>	sip::invite	>	incoming_call()
telnet	>	set in "trust" ...	>	set_interface()
http	>	soap::xml	>	AddShoppingCart()
file	>	qt:moov	>	play_movie()

50 ways to encode your lover



- ✓ `def add_csw_speakers(emails)`
- ✓ `def add_csw_speakers (emails):`
- ✓ `int add_csw_speakers(const char **emails)`
- ✓ `public void add_csw_speakers(String[] emails)`

command line interface




```
cs> add speakers "foo@bar.com" "a@b.com"
```

xdr/rpc



```
07 e2 5d 7b 00 00 00 00 00 00 00 02 00 01 86 a0  ..]{}.....
00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00  .....
00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 0b  .....
66 6f 6f 40 62 61 72 2e 63 6f 6d dd 00 00 00 07  foo@bar.com.....
61 40 62 2e 63 6f 6d dd                               a@b.com.
```


asn.1 (ber)



```
30 16 04 0b 66 6f 6f 40 62 61 72 2e 63 6f 6d 04 0...foo@bar.com.  
07 61 40 62 2e 63 6f 6d .a@b.com
```

soap/xml



```
<s:Envelope xmlns:s="http://schemas.xml.soap.org">
  <s:Body>
    <csw:AddSpeakers xmlns:csw="http://www.cansecwest.com">
      <csw:speaker>foo@bar.com</csw:speaker>
      <csw:speaker>a@b.com</csw:speaker>
    </csw:AddSpeakers>
  </s:Body>
</s:Envelope>
```

write once, 0-days everywhere

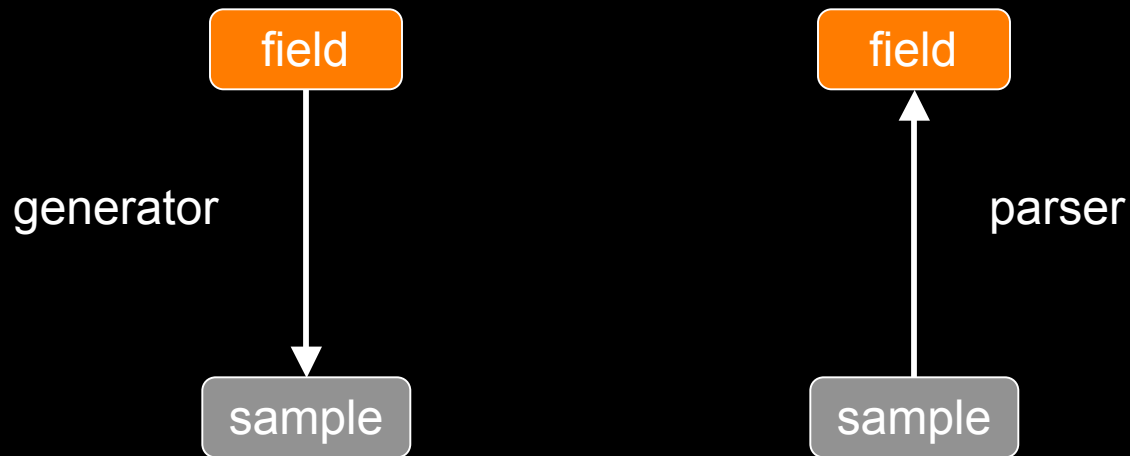


- ✓ problems at multiple levels
 - ✓ inside the method
 - ✓ with the encoding
 - ✓ with the message
 - ✓ with the protocol
 - ✓ with the channel
- ✓ YMMV with 0x41's
 - ✓ depends on which layer of the onion
 - ✓ validity of one layer is a prerequisite for the next

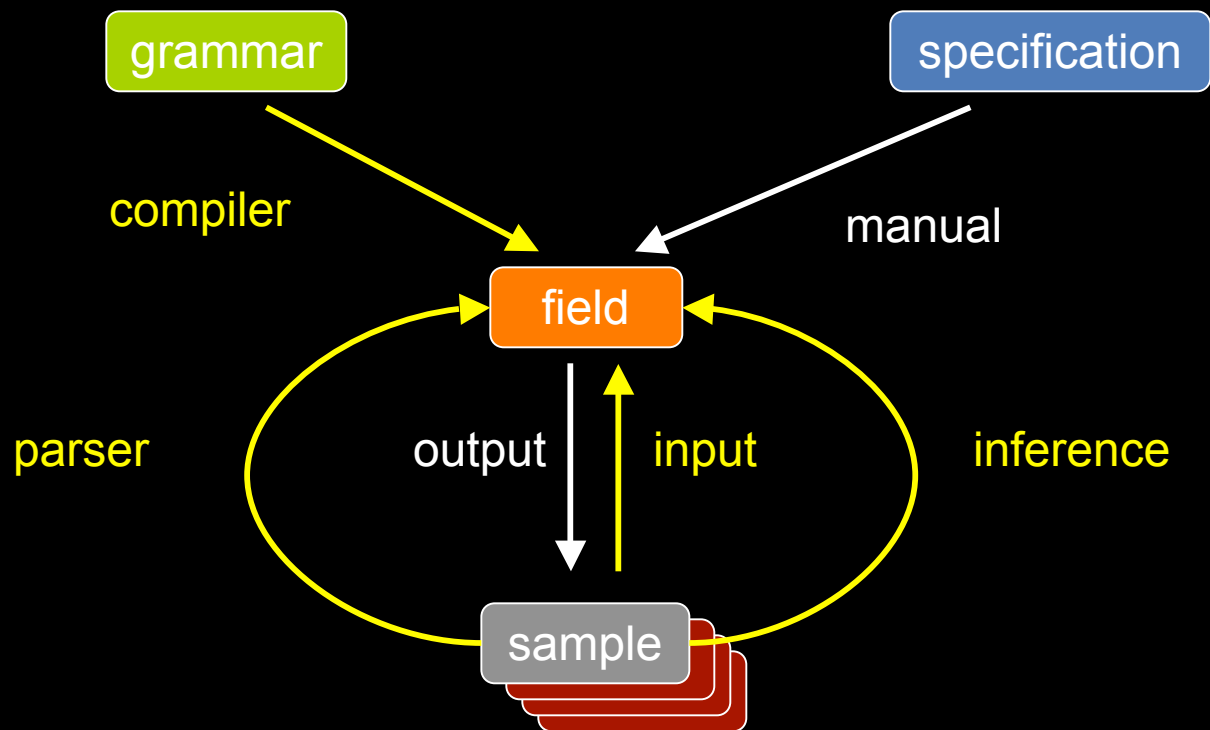
m-theory

A thick, horizontal orange brushstroke underline is positioned directly beneath the text 'm-theory'. The stroke is irregular and textured, resembling a hand-drawn line with a slight wavy pattern.

symmetry breaking



m-theory



definition: field



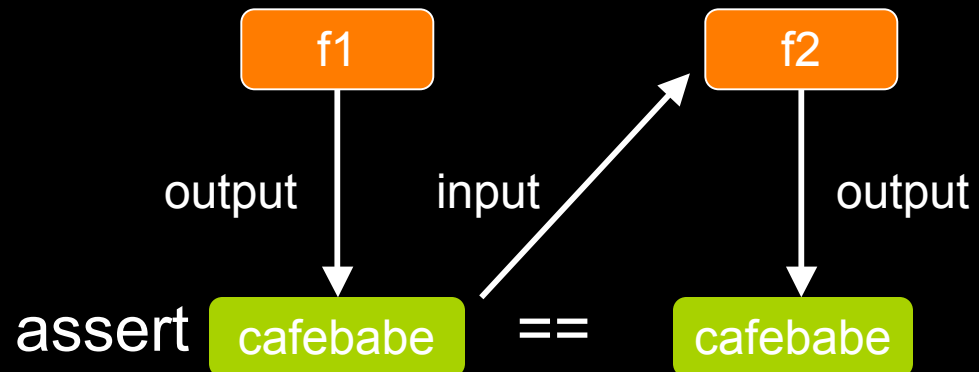
- ✓ core abstraction
 - ✓ exists outside of specific channels
- ✓ nested to arbitrary levels
- ✓ structural/semantic relationships
- ✓ primary *methods*
 - ✓ input
 - ✓ output
 - ✓ alternates
 - ✓ static and dynamic
 - ✓ semantics and context-aware
 - ✓ domain-specific
 - ✓ fuzzing is one kind of attack vector!

fields



- ✓ uint8, uint16, flags32, enum24, ...
- ✓ length, checksum, crc
- ✓ name-value, dsv, c-string, tlv
- ✓ http-header, http-content-length-header
- ✓ sip-request, sip-via-header
- ✓ qt-moov-atom, png-ihdr-chunk

super symmetry and equivalence



laws of fields



- ✓ fields shall be channel agnostic
- ✓ that which is sent has potential for alternates
 - ✓ requests in client mode
 - ✓ responses in server mode
- ✓ that which is received is canonicalized
 - ✓ etag's, challenge-handshake, cookies
 - ✓ via headers, route tables

input bounds

```
ascii.line {  
  encode.base64 {  
    ascii.dsv :delimiter => ':' {  
      string.basic :value => 'hello'  
      string.basic :value => 'world'  
    }  
  }  
}
```

generates and parses "aGVsbG86d29ybGQ=\r\n"

action at a distance



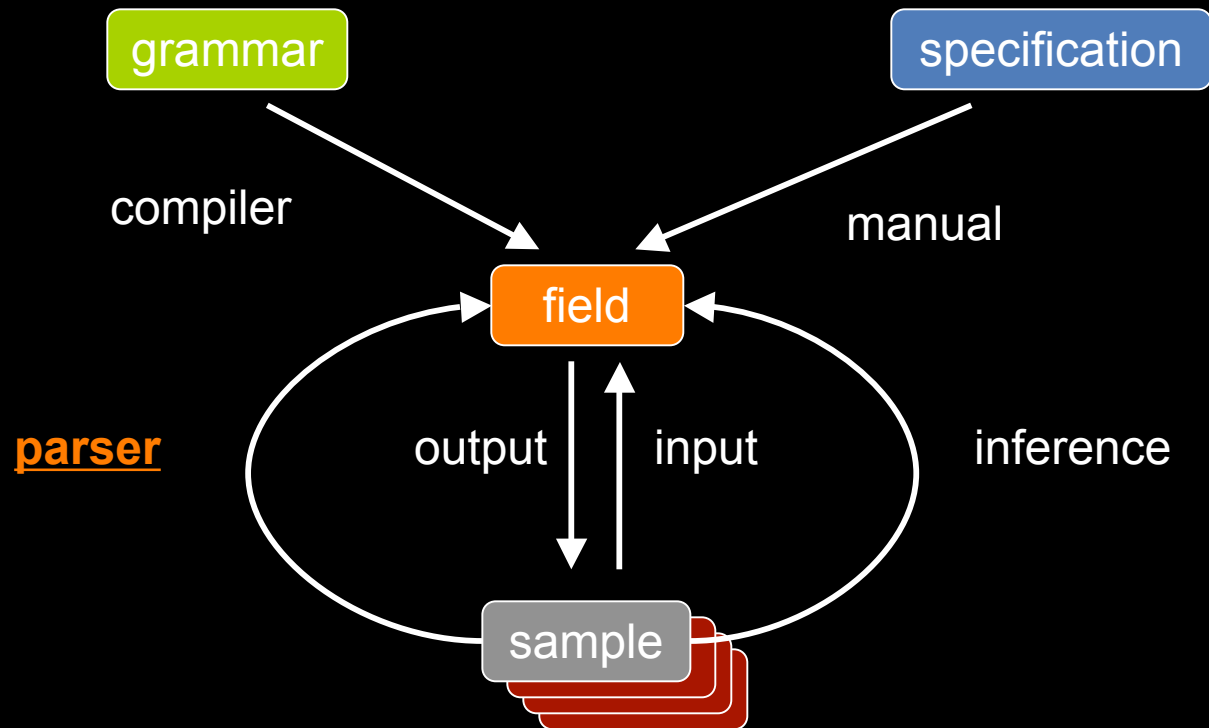
```
struct {  
  foo = ascii.line {  
    ascii.dsv(:delimiter => ' ') {  
      ascii.c_string :value => 'hello world'  
    }  
  }  
  ascii.length :of => foo  
}
```

fieldomatic complexity



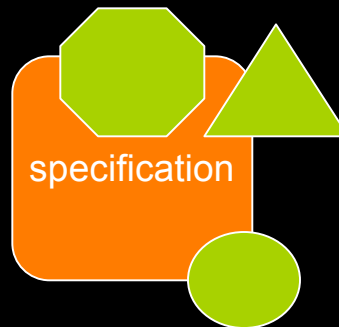
- ✓ fields interact and relate to each other
 - ✓ output of one drives the other
 - ✓ form an acyclic graph
- ✓ use for dynamic alternates
 - ✓ length, offsets
 - ✓ ordering, prerequisites and constraints
- ✓ dependencies
 - ✓ structure, semantics and state
- ✓ #inbound-edges == cyclomatic complexity

laziness



parser

- ✓ specifications to fields take time
 - ✓ very manual
- ✓ extension-space is unbounded
- ✓ not a static problem
 - ✓ constraints and semantics not always obvious
- ✓ being an rfc bigot doesn't do you any good



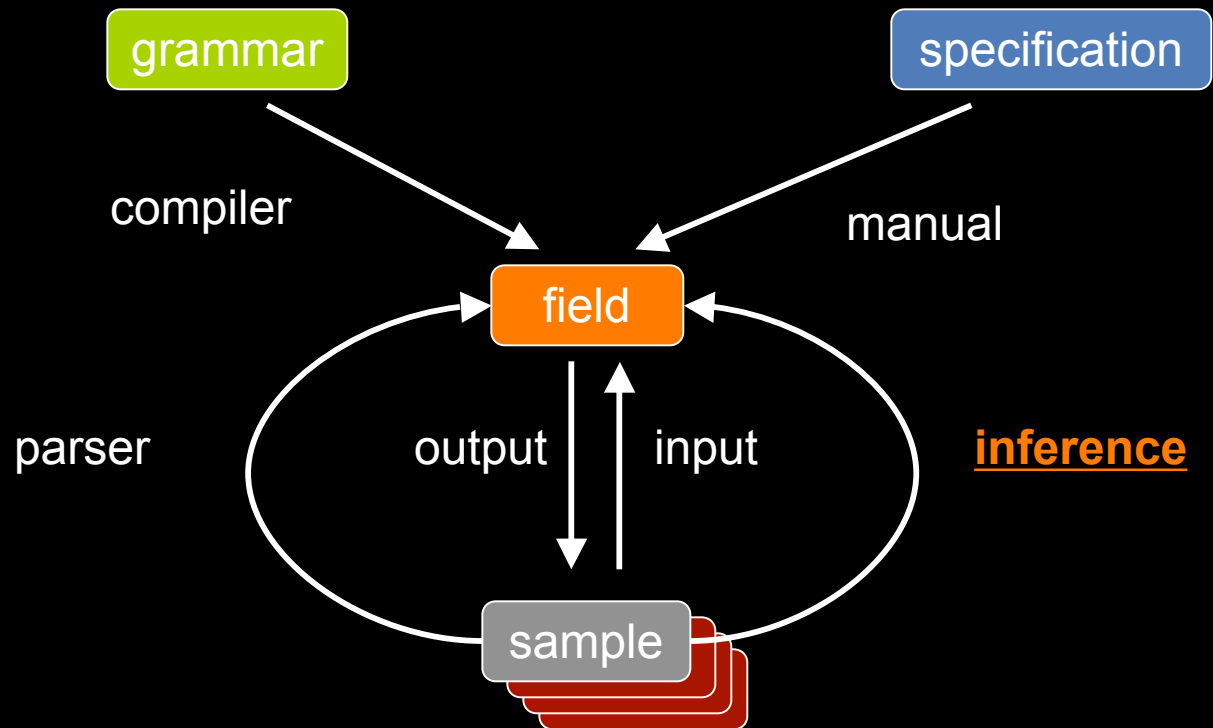
parser



quicktime parser (snippet)

```
def parse_atom_elst
  type.uint8 'version'
  type.flags24 'flags'
  nentries = type.count32('num-entries')
  group('entries') { |g|
    nentries.of = g
    nentries.value.times do |i|
      group("entry-#{i}") {
        type.uint32 'track-duration'
        type.time32 'media-time'
        type.uint32 'media-rate'
      }
    end
  }
end
```

impatience

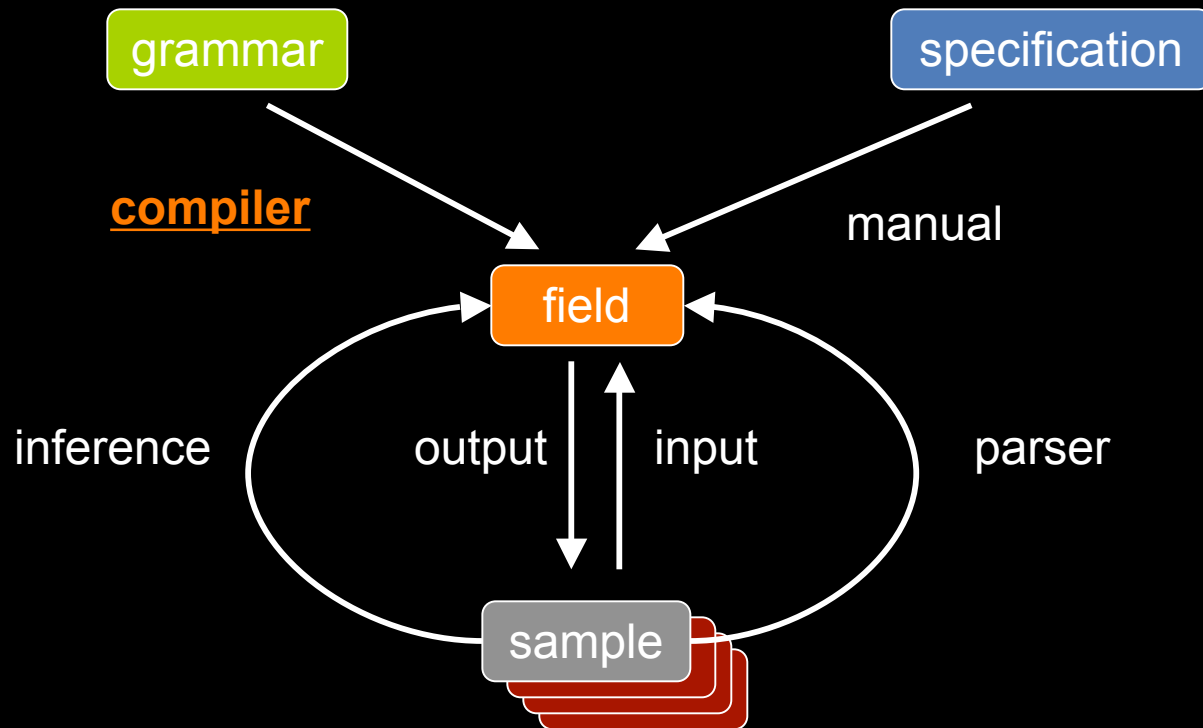


inference



- ✓ similar to edge detection
 - ✓ extract fields and relationships
- ✓ structural and semantic inference
- ✓ results in a semantic dom
- ✓ field's input method guides inference

hubris

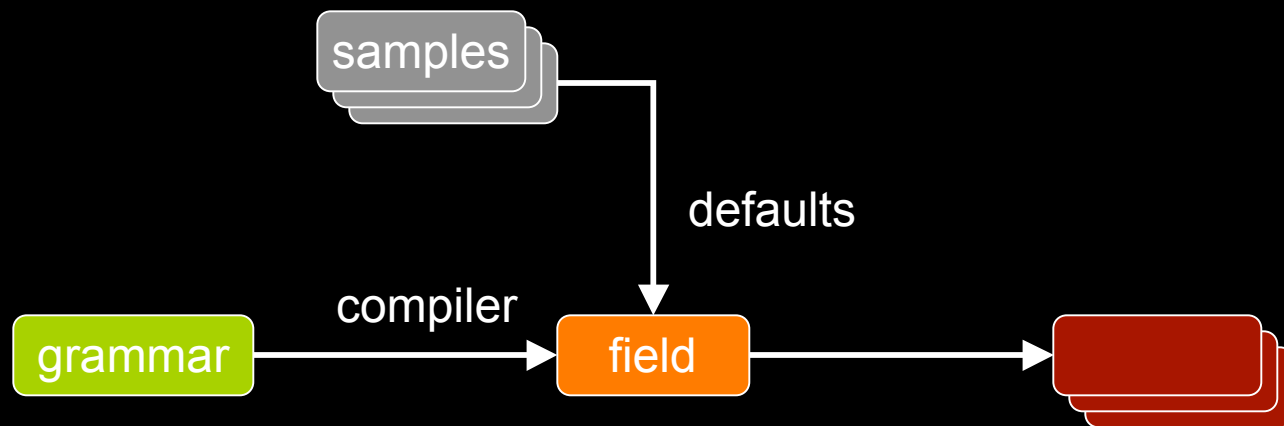


compiler



- ✓ machine parsable grammars
 - ✓ ASN: asn1c
 - ✓ XDR: rpcgen
 - ✓ IDL: midl (pymrpc)
 - ✓ ...
- ✓ remember: a string is a string is a ...
- ✓ what's missing?
 - ✓ transactions, scenarios and state
 - ✓ and yeah, encoding and transport

compiler



summary



- ✓ protocols *do* unify in the 11th dimension
- ✓ semantic dom is all there to it
- ✓ there's no such thing as a CLI fuzzer
 - ✓ it's just a different channel
- ✓ laziness, hubris and impatience
 - ✓ not just for perl programmers
- ✓ don't write fuzzers
 - ✓ build a semantic dom instead
 - ✓ fuzzing "just" happens

questions?



kowsik@musecurity.com

<http://labs.musecurity.com>