Developing Fuzzers with Peach 2.0

MICHAEL EDDINGTON
MIKE@LEVIATHANSECURITY.COM

Agenda

- Quick Introduction to Fuzzing
- Peach!
- Demo

INTRODUCTION TO FUZZING

Introduction to Fuzzing

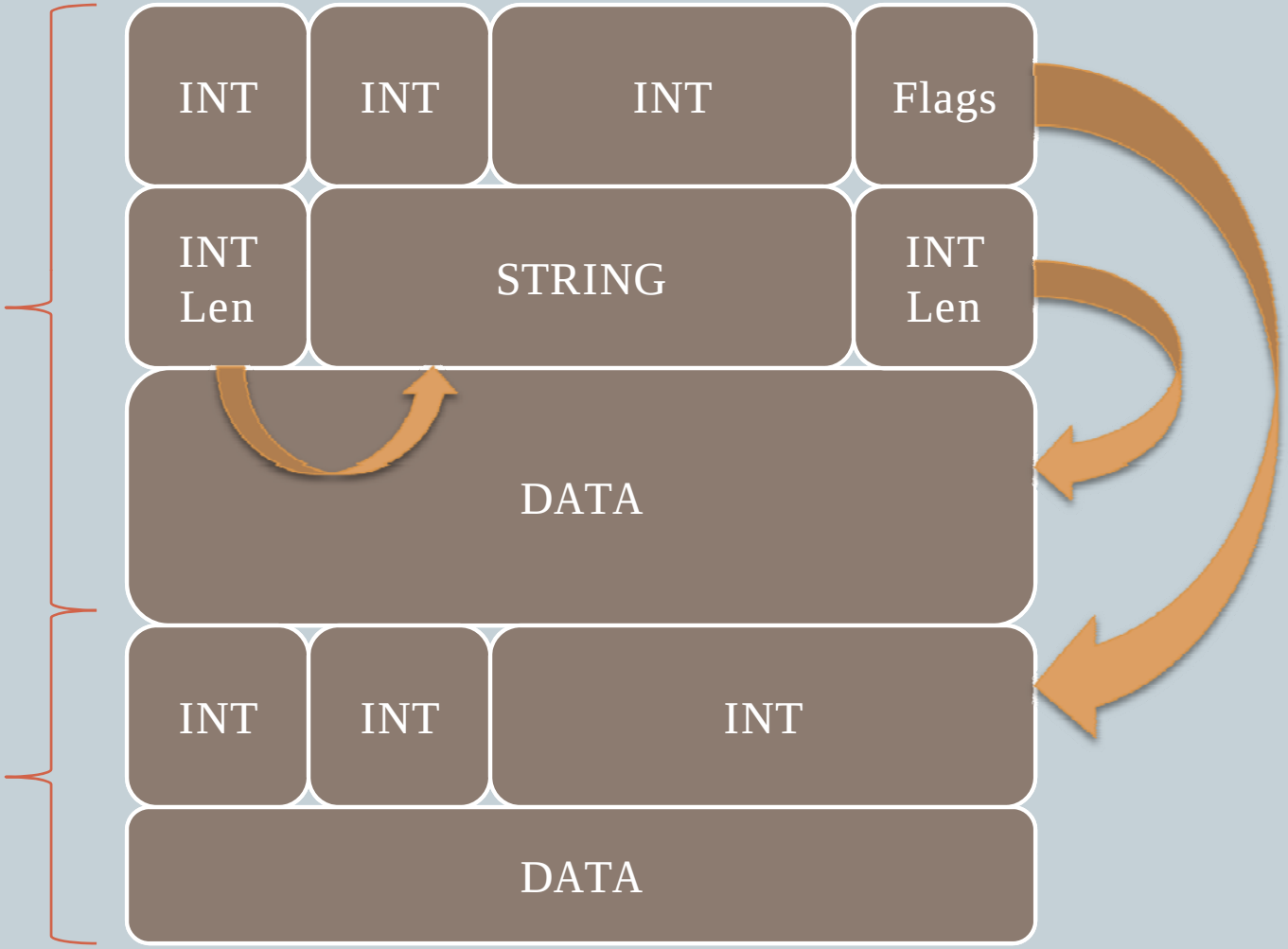
- Long time tool of the security researcher
- Unexpected input mutations
- Typically good bang for buck
- Range from dumb to smart

- Dumb: Bit flipping, sending random data to interface
- Smarter: Understanding relationships in the data

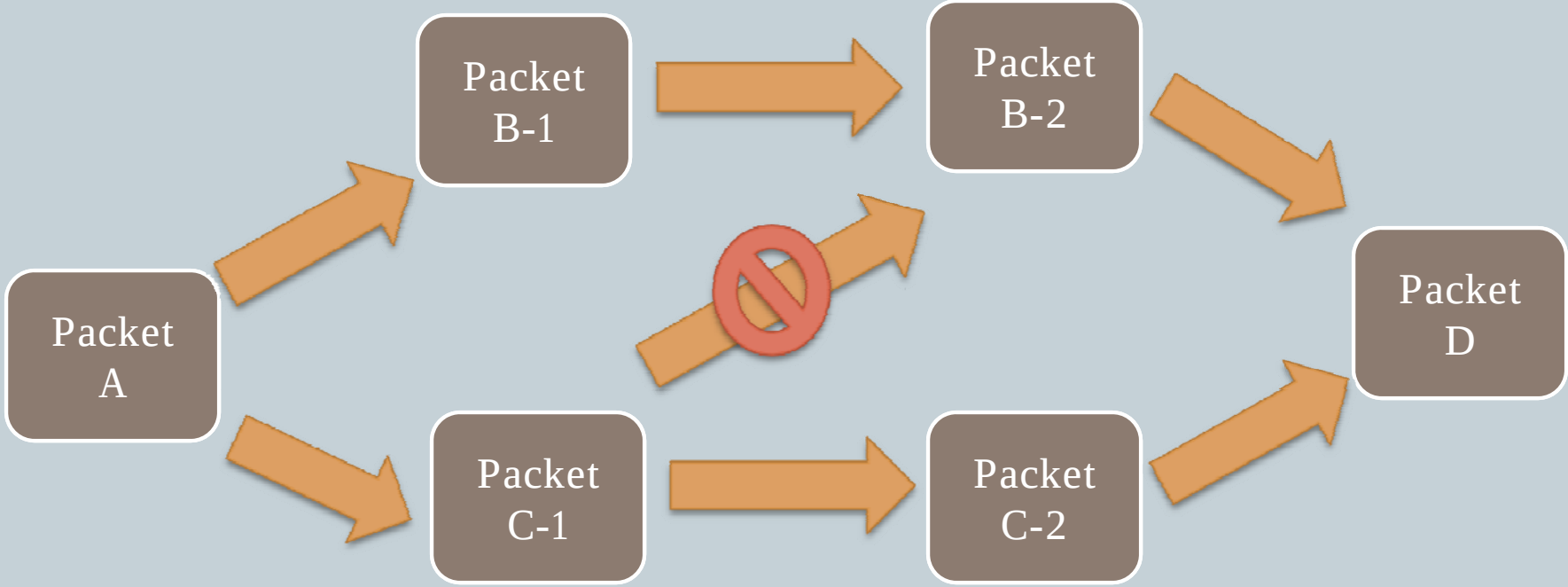
Dumb Fuzzers

- setuidproggy `perl –e “print ‘A’ x 1024”`
- nc 127.0.0.1 9345 < /dev/random
- export UID=`perl –e “print ‘A’ x 2000”`

Smarter



State Aware



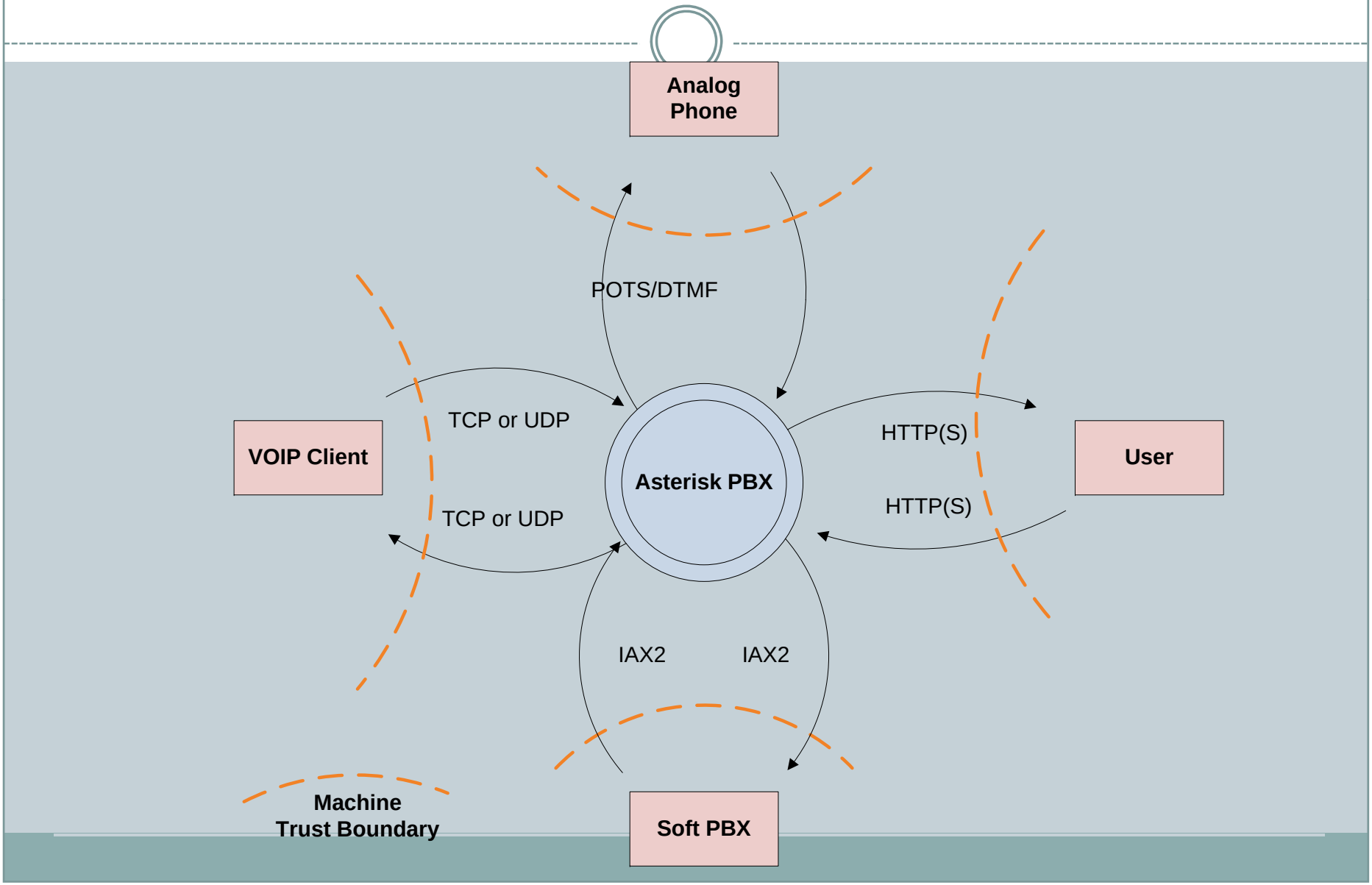
Is Fuzzing Right For You?

- What types of bugs are you looking for?
- Managed or unmanaged code?
- Faults must be detectable
  - Access violations
  - File system access
  - Etc.
- Not so good with managed languages like Java, .NET
  - Exception: Calling into unmanaged resources and libraries

Determining Targets

- Threat Model
  - Identify trust boundaries (DFD)
  - Input points (DFD)
  - Custom parsers
  - Complex logic

Asterisk: Data Flow Diagram



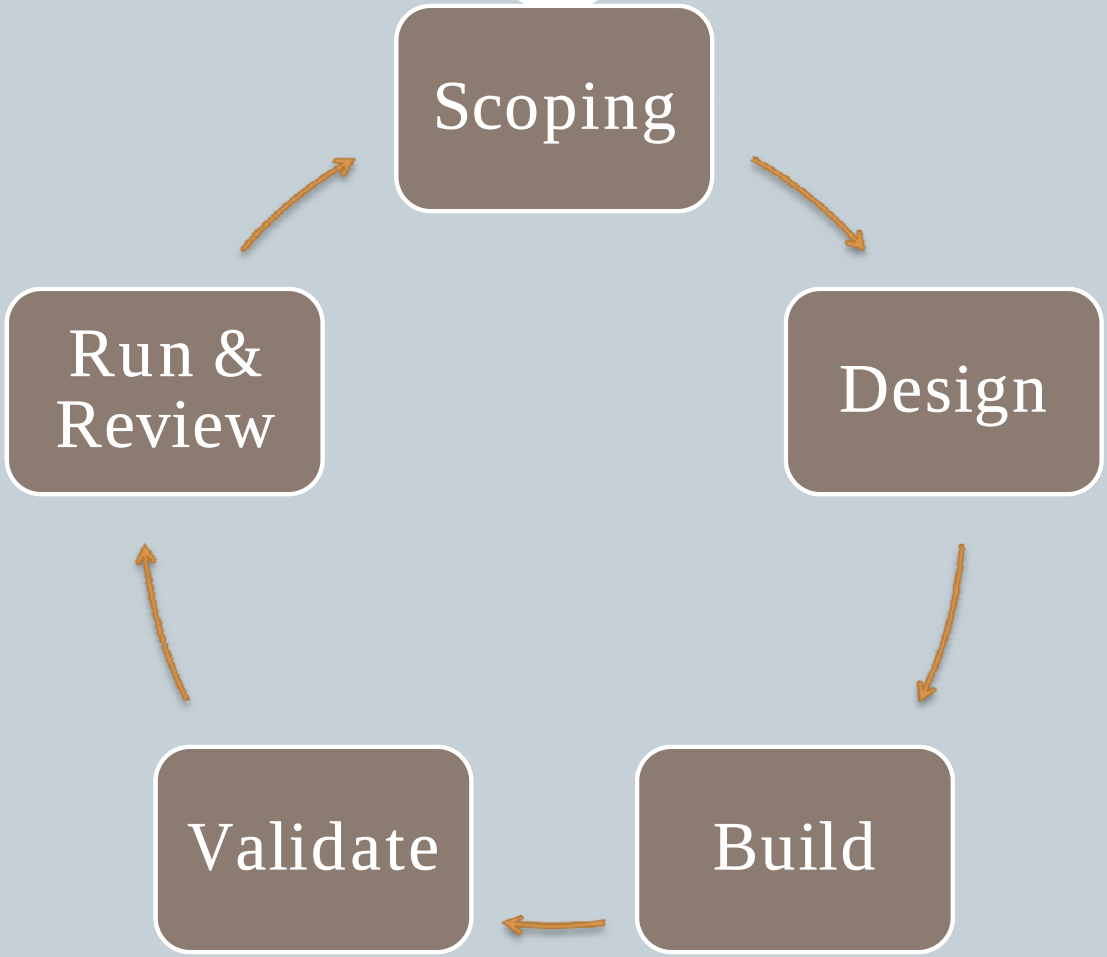
Asterisk: Protocols/Parsers

- VOIP Client
  - SIP
  - MGCP
  - H.323
  - SRTP/ZRTP
- Server 2 Server
  - IAX/IAX2
- Admin/User
  - HTTP(S)
- Backend Storage
  - SQL
  - IMAP
- Analog POTS
  - DTMF
  - Management

Asterisk – Risk Ranking

Ranked Entry Points	Ranked Parsers
• VOIP • HTTP • Server 2 Server • POTS	• IAX/IAX2 • SIP • SRTP/ZRTP • MGCP • H.323

Fuzzer Life Cycle



Design

- Investigate selected code areas
- Determine test delivery method
  - Is custom code required
  - Are we extending existing test code
- Design any custom code
- Determine interesting code paths and data
- How will faults be detected?
  - Page Heap
  - Attached debugger

Build

Validation

- Critical to validate the fuzzer!
  - Is data produced in correct format?
  - Is target accepting data and processing it to correct depth?
  - Are data relationships correct? As data changes is it still accepted?
- May require looping back to design/dev phase a few times…

Runs

- Run times can be long
  - Days to weeks
  - Speeding up tests may be needed
- Monitoring
  - Is fuzzer still alive?
  - Is target?
- Reviewing of logs and detected faults
  - Tracking down bug cause can be time intensive!

PEACH

Peach – The Mission

Peach 1.0	Peach 2.0
• Fast development time • Fuzz anything • Extensibility • Reusability • Repeatability	• Reduce learning curve • No coding required • Faster development • Same extensibility • Same reusability • Same repeatability

Peach 1.0

- Fuzzers written in Python
- Instrumentation via wrapper APIs
- No data definition layer (DDL), just fuzzer
- Steep learning curve
- Complex fuzzers result in complex fuzzer code

Peach 2.0

- Data definition layer
  - Full XML Schema
  - Data relationships
- Engine
- Agents
- Peach API
  - Python
  - .NET
  - Java
  - COM
- Automatic DDL
  - Wireshark Captures
  - C struct's
  - IDL/TLB
- Peach Builder API

Benefits of Data Definition

- Easy reuse of definitions
- Complex mutations can be applied to a DDL tree
- Definitions do not need changing to improve data generation or mutation
- Data read into definition as well as generated

Data Definition

- Namespaces
- Templates
- Data sets
- Agents
  - Monitors
- Tests
  - Publishers
- Runs
  - Logger

Namespaces

- Core of reuse for Peach DDLs
- Files can be hosted remotely (HTTP/S)
- Libraries of common types can be built
- Large DDLs can be split up

```
<Peach>
  <Include ns="default" src="file:default.xml" />
  <Include ns="pt" src="file:PeachTypes.xml" />
</Peach>
```

Templates

- Define structure of data
- Define relations in data
- Reuse definitions

- Block
- Sequence
- Choice
- String
- Number
- Flags/Flag
- Blob

- Relation
- Transformer

Template Example

```
<Template name="UdpPacket">
 <Number name="SrcPort" size="16" endian="network" />
 <Number name="DestPort" size="16" endian="network" />
 <Number name="Length" size="16" endian="network">
   <Relation type="size" of="Data" />
 </Number>
 <Number name="CheckSum" size="16" endian="network">
   <Relation type="checksum" of="UdpPacket" />
 </Number>
 <Blob name="Data" />
</Template>
```

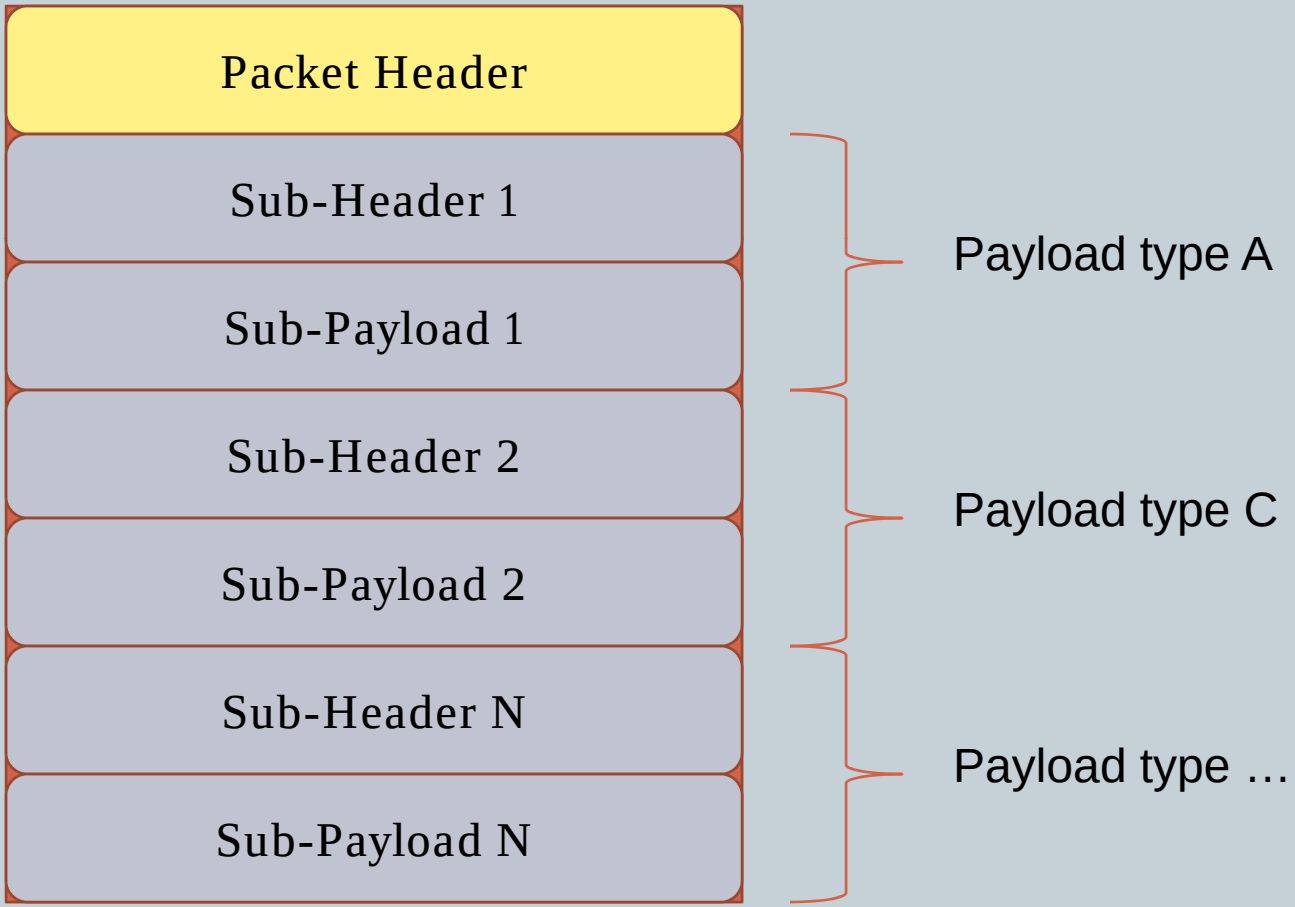
Reuse via References

```
<Template name="http-header">
  <String name="header" />
  <String value=": " />
  <String name="value" />
  <String value="\r\n" />
</Template>
```

Reuse via References

```
<Block  ref="http-header">
  <String name="header" value="Content-Length"/>
  <String name="value">
     <Relation type="size" of="body" />
  </String>
</Block>
<Blob name="body" />
```

Choice


Packet Header
Sub-Header 1
Sub-Payload 1
Sub-Header 2
Sub-Payload 2
Sub-Header N
Sub-Payload N
Payload type A
Payload type C
Payload type …

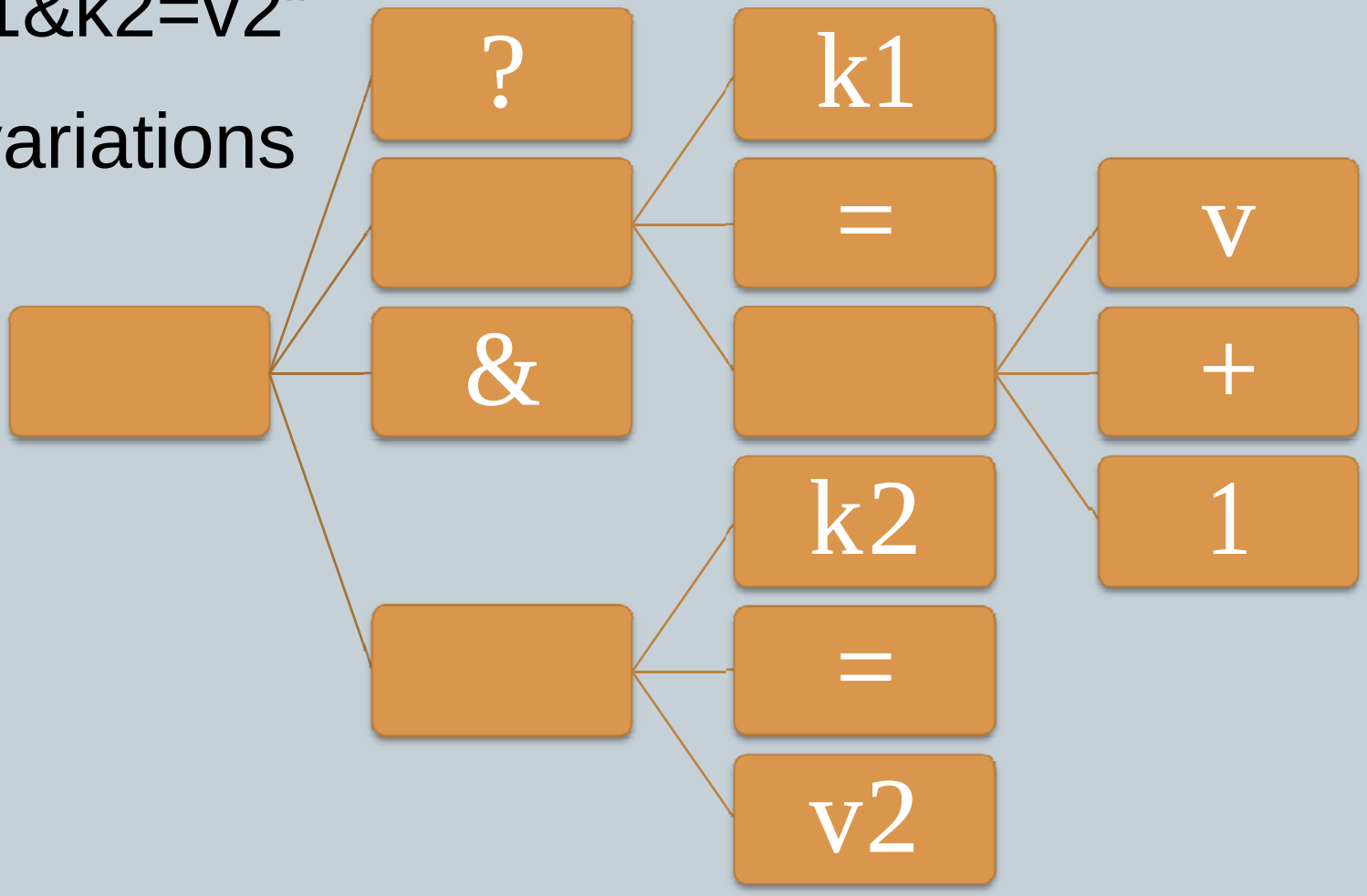
Choice

```
<Choice minOccurs="1" maxOccurs="1">
 <Block>  <!-- A -->
  <Number name="Type 1" value="1" />
  ...
 </Block>
 <Block>  <!-- B -->
  <Number name="Type 2" value="2" />
  ...
 </Block>
</Choice>
```

String

“?k1=v+1&k2=v2”

40,247 variations



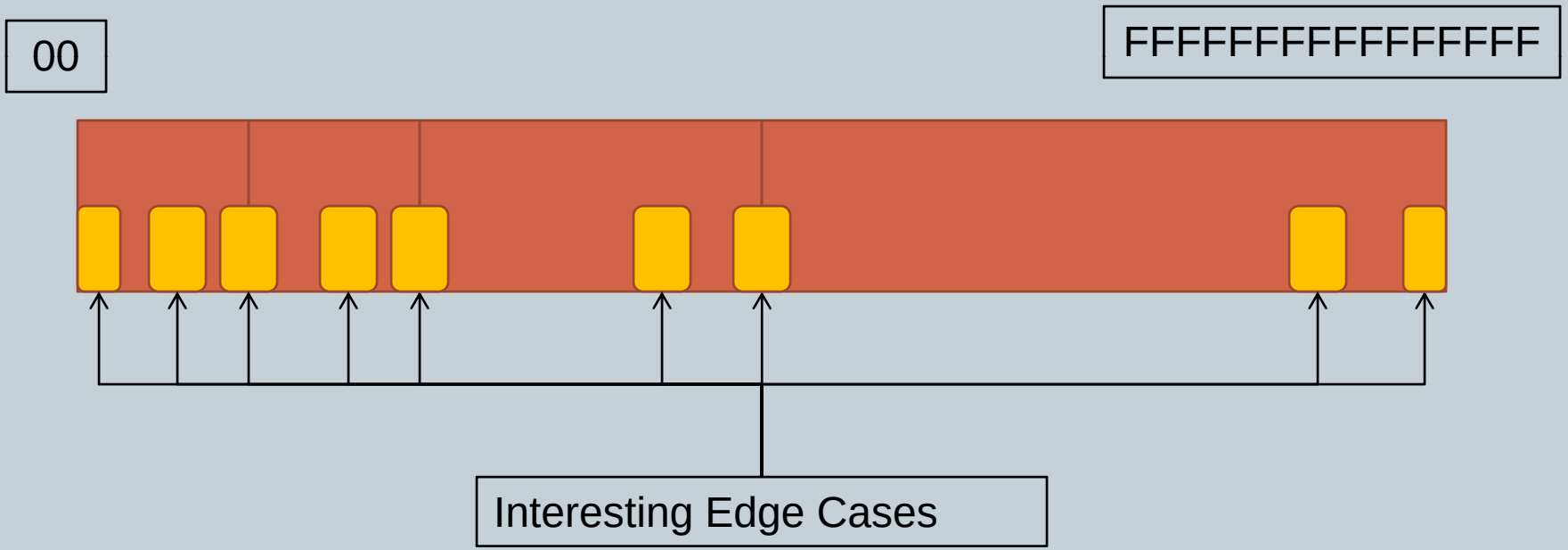
String Attributes

- Length
- Null termination
- Char vs. wchar
- Pad character
- Static

String – SIP Example

```
<Template name="SipSimpleSubscribe">
 <String value="SUBSCRIBE sip:dd2@192.168.1.194 SIP/2.0\r\nVia:
  SIP/2.0/TCP
  192.168.6.1:5060;branch=z9hG4bK754C13CB260F1EA425F4\r\nFrom:
  &lt;sip:dd@192.168.1.194&gt;;tag=806527798;epid=1234567890\r\nTo:
  &lt;sip:dd2@192.168.1.194&gt;\r\nMax-Forwards: 10\r\nCSeq: 4
  SUBSCRIBE\r\nUser-Agent: Purple/2.0.2\r\nCall-ID: " />
 <String>
   <Generator class="PerCallIncrementor" />
 </String>
 <String value="\r\nExpires: 1200\r\nAccept: application/pidf+xml,
  application/xpidf+xml\r\nEvent: presence\r\nContact:
  &lt;sip:dd@192.168.6.1:5060;transport=tcp&gt;;methods=&quot;MESS
  AGE, SUBSCRIBE, NOTIFY&quot;\r\nContent-Length: 0\r\n\r\n" />
</Template>
```

Number



Number

- Size in bits, signed, byte order

```
<Number size="16" value="1024" signed="false" />

<Number size="32"  endian="little"/>
```

Blob

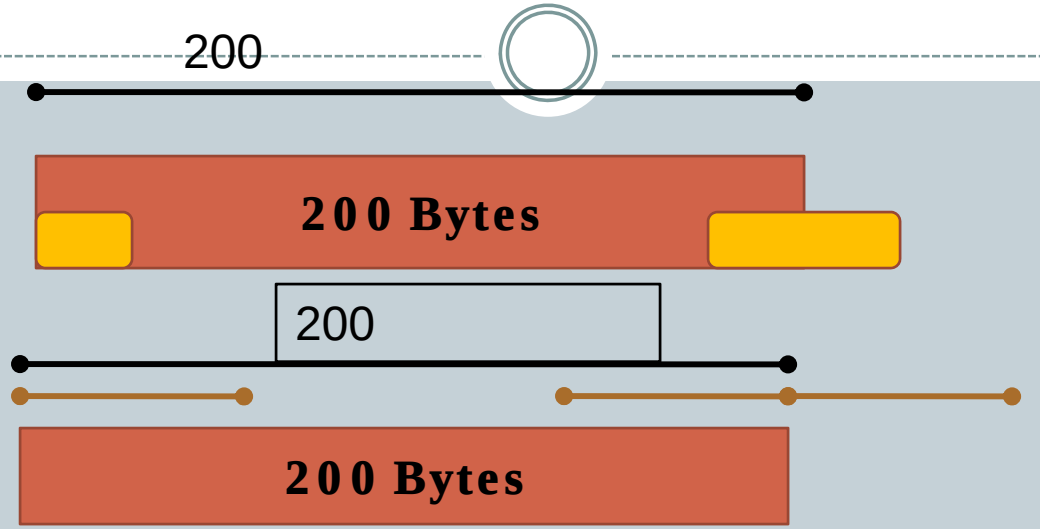
- Unknown binary data
- Random bit flipper applied
- Data supplied typically as Hex

```
<Blob valueType="hex" value="41 42 0x43" />
```

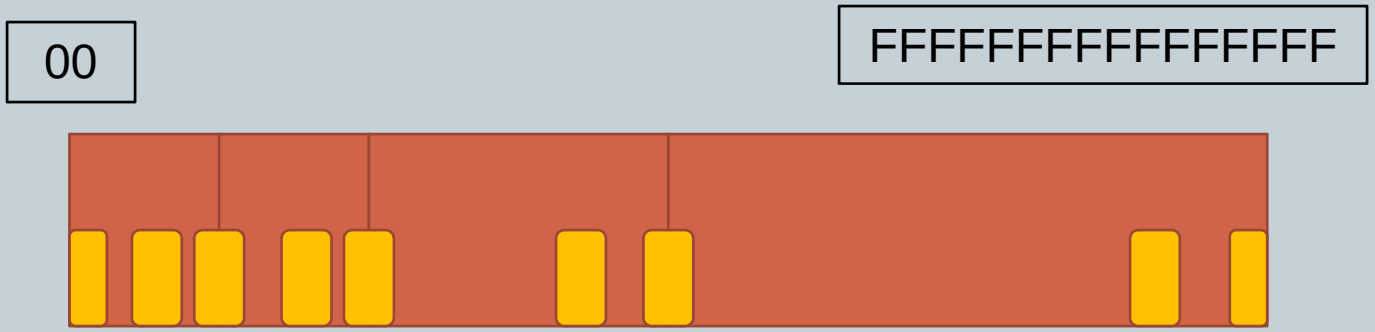
Relation

- Define simple relationships between data in a definition
  - Size of
  - Count of
  - Checksum of
  - Etc.
- Relations are used by the engine to produce interesting mutations and edge cases
  - Vary the size of something +/- N
  - Vary the data so size is near zero or integer boundaries

Relation: Size



- Length:
- Data:
- Length:
- Data:



- Data:

Relation Example

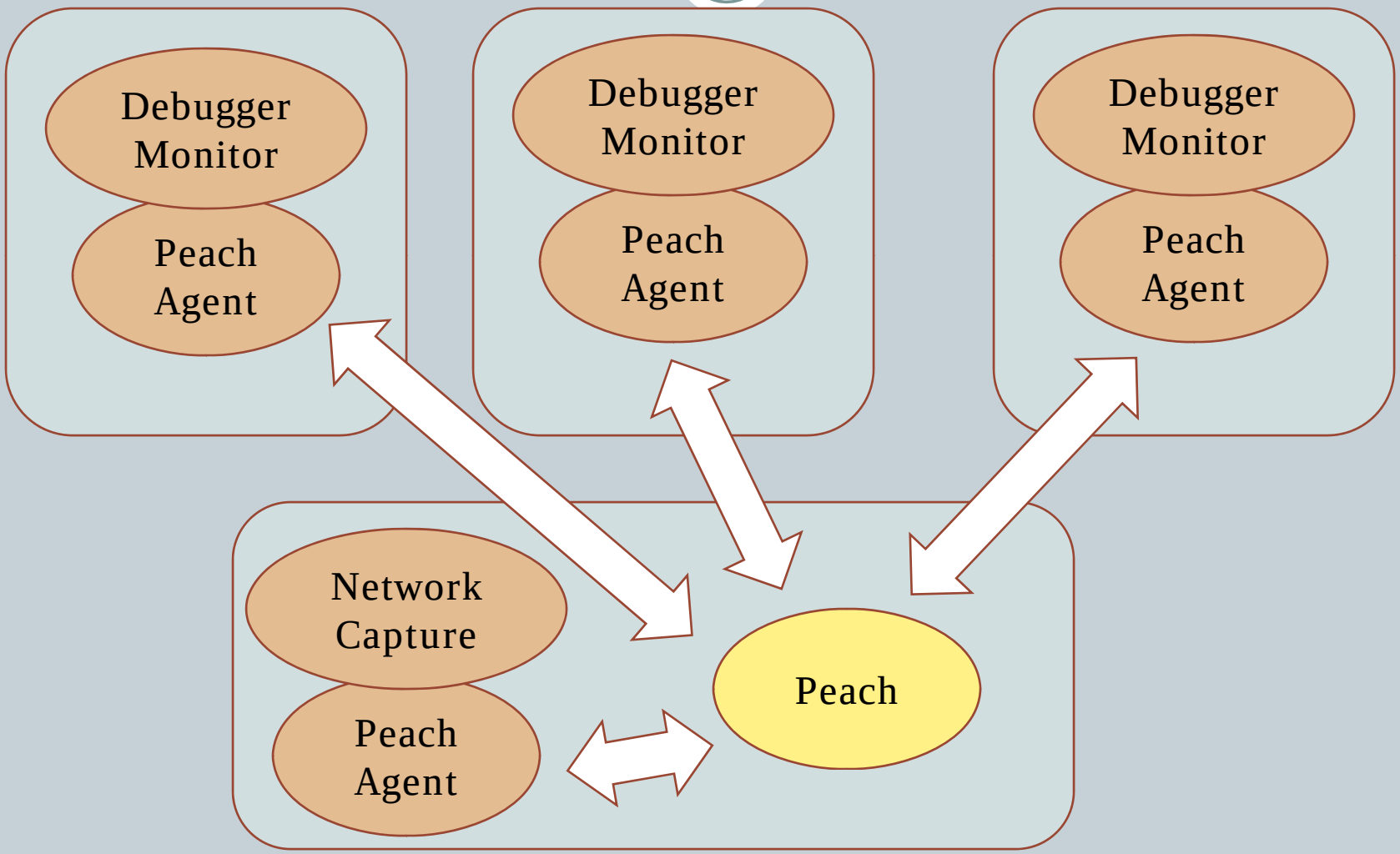
```
<Block  ref="http-header">
  <String name="header" value="Content-Length"/>
  <String name="value">
     <Relation type="size" of="body" />
  </String>
</Block>
<Blob name="body" />
```

Data Sets

Data Sets

```
<Data name="Get" template="httprequest">
  <Field name="request.method" value="GET" />
</Data>

<Data name="Post" template="httprequest">
  <Field name="request.method" value="POST" />
</Data>
```

Agents & Monitors



Monitors

- Debuggers
  - User mode
  - Kernel mode
- Network monitoring (pcap)
- Process control
- Virtual machine control

Agent Example

```
<Agent name="LocalAgent" location=".">
  <Monitor class="vm.Vmware">
    <Param name="vmx" value="E:\VM\Win.vmx" />
  </Monitor>
</Agent>
```

Agent Example

```
<Agent name="Fuzz01Agent" location="fuzz01">
 <Monitor name="Debugger"
  class="debugger.WindowsDebugger">
   <Param name="command" value="C:\CSrv\CSrv.exe"/>
   <Param name="params" value="192.168.1.10"/>
 </Monitor>

 <Monitor name="Network" class="network.PcapMonitor">
   <Param name="filter" value="tcp and port 4242" />
 </Monitor>
</Agent>
```

Publishers

- “Publish” generated test variations
- Via network, COM, to file system, etc
- Custom publishers written in python

```
<Publisher class="tcp.Tcp">
  <Param name="host" value="127.0.0.1" />
  <Param name="port" value="80" />
</Publisher>
```

Test Control

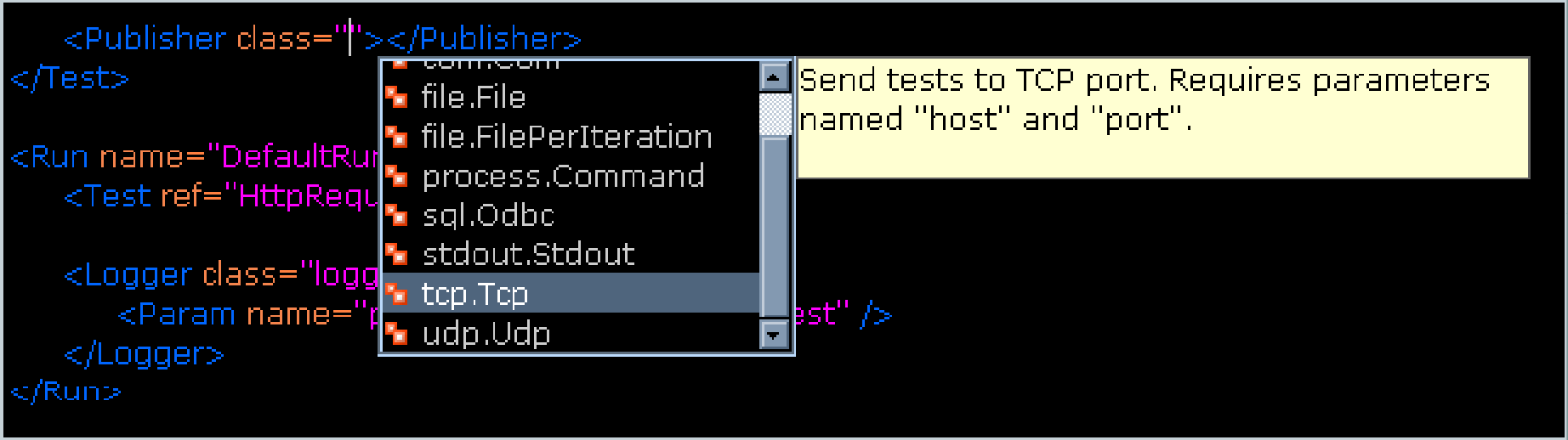
- Tests are:
  - Template
  - Data set (optional)
  - Agents (optional)
  - Publisher
- Runs are:
  - One or more Tests
  - Logger (optional)

Test & Run Example

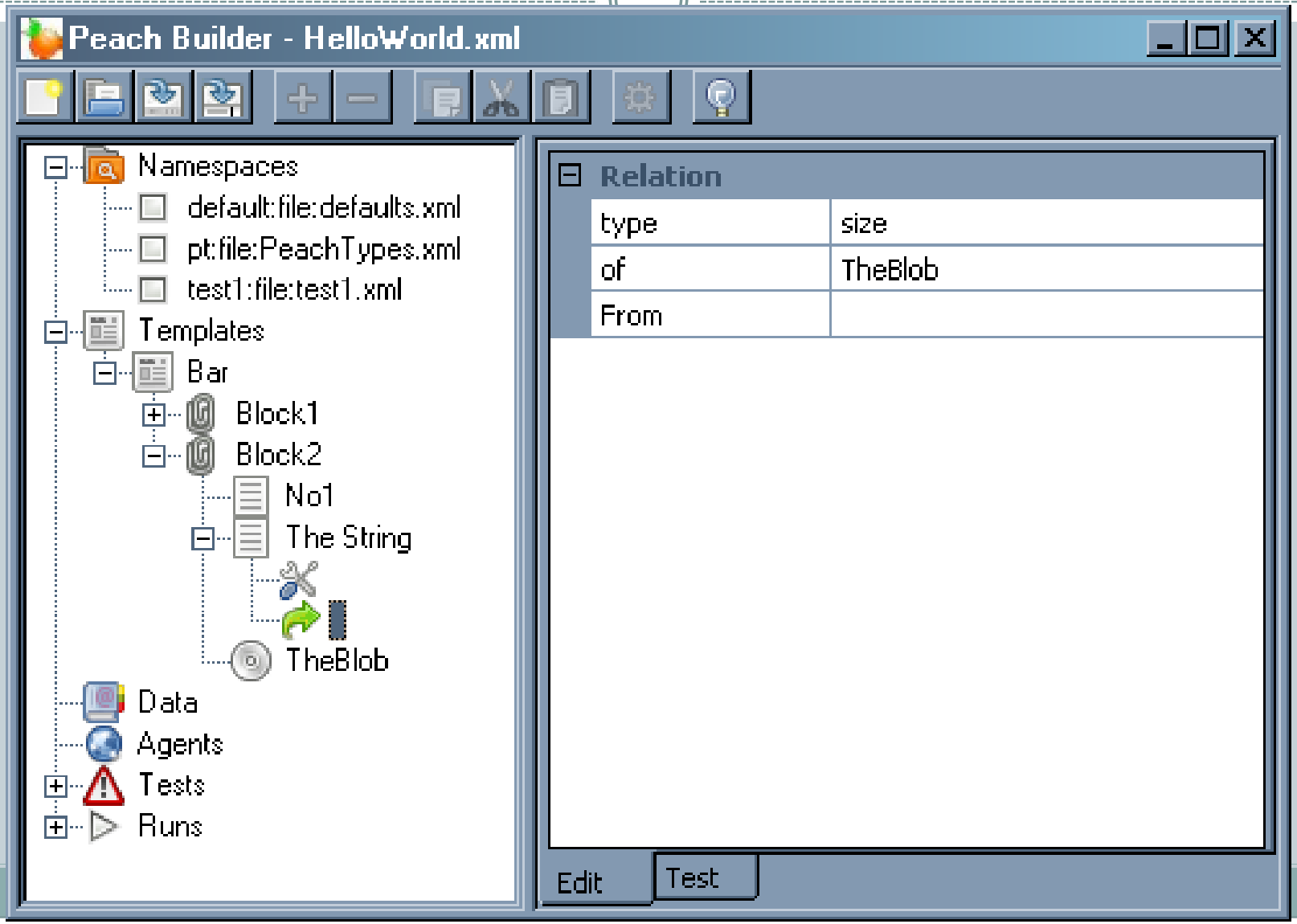
```
<Test name="HttpRequestTest" description="HTTP Req GET">
 <Template ref="HttpRequest" />
 <Agent ref="LocalAgent" />

 <Publisher class="tcp.Tcp">
  <Param name="host" value="127.0.0.1" />""
   <Param name="port" value="80" />""
    </Publisher>
   </Test>

<Run name="DefaultRun" description="HTTP Request Run">
 <Test ref="HttpRequestTest" />

 <Logger class="logger.Filesystem">
  <Param name="path" value="c:\peach\logs" />
 </Logger>
</Run>
```

Lowering the Learning Curve

Documented XML Schema



Peach Builder



Peach Shark

DEMO!

Peach in action..Rraarrrr!

Peach 2 Roadmap

- 2.0 (PacSec 2007)
- 2.1 (Jan/Feb 08)
  - Testing added to Peach Builder
  - Parallel Fuzzing
  - Agent Pools
- 2.2 (March/April 08)
  - MITM Peach Shark
  - Custom DOM walking
- Latest always in SVN!

Q & A

HTTP://PEACHFUZZ.SF.NET

HTTP://PHED.ORG

MIKE@LEVIATHANSECURITY.COM