CODENOMICON

CERT/CC Vulnerability Discovery Workshop

Realizing the Fuzzing Potential: Precision and Accuracy vs. Coverage

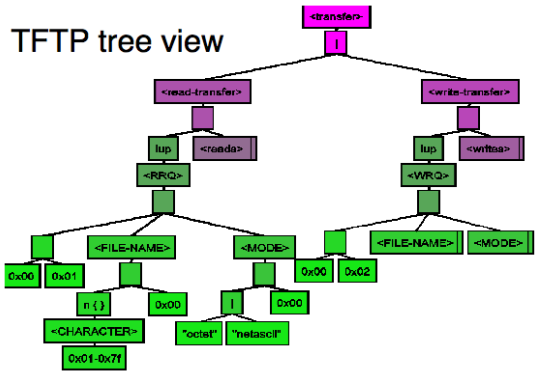
http://www.codenomicon.com/products/coverage.shtml

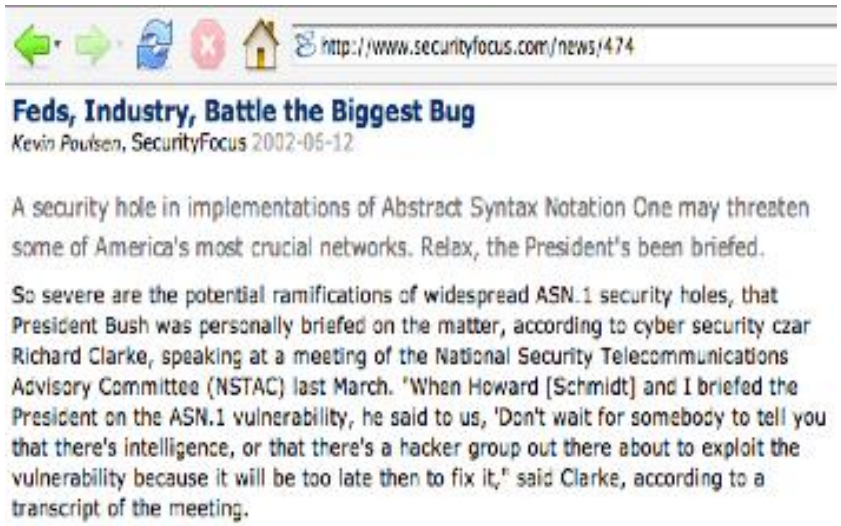
Ari Takanen
CTO
Codenomicon

Mikko Varpiola
Fuzzing Specialist
Codenomicon

About Codenomicon

- **Founded in 2001, after five years of research in product security at University of Oulu (1996-2001)**
- **Customers include:**
  - Manufacturers
  - Telco Service Providers
  - Defense
  - Finance and Leading Enterprises
- **CROSS (IPv6, xml,…, FTP)**





CODENOMICON

About Mikko Varpiola

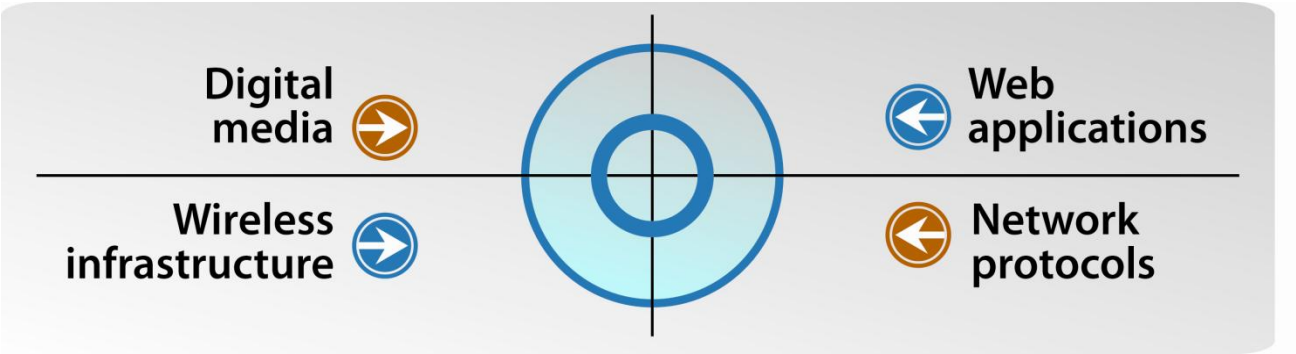
- **One of the original PROTOS team members at University of Oulu**
  - WAP, LDAP, SNMP, …
  - Fuzzing (and otherwise breaking SW) since 1996 or so

- **Co-founder of Codenomicon, key customer services specialist in USA market**
  - Need any weird protocols extensively fuzzed: contact Mikko ;-)

What is Codenomicon DEFENSICS?

- **Product line of model-based fuzzers for over 200 protocols and interfaces**
- **Both Client/Server**
- **20-30% of tools are customer proprietary**

- **Only solution covering whole application stack**
  - Wireless / Layer 2
  - Network protocols
  - File formats
  - Application / XML
  - API



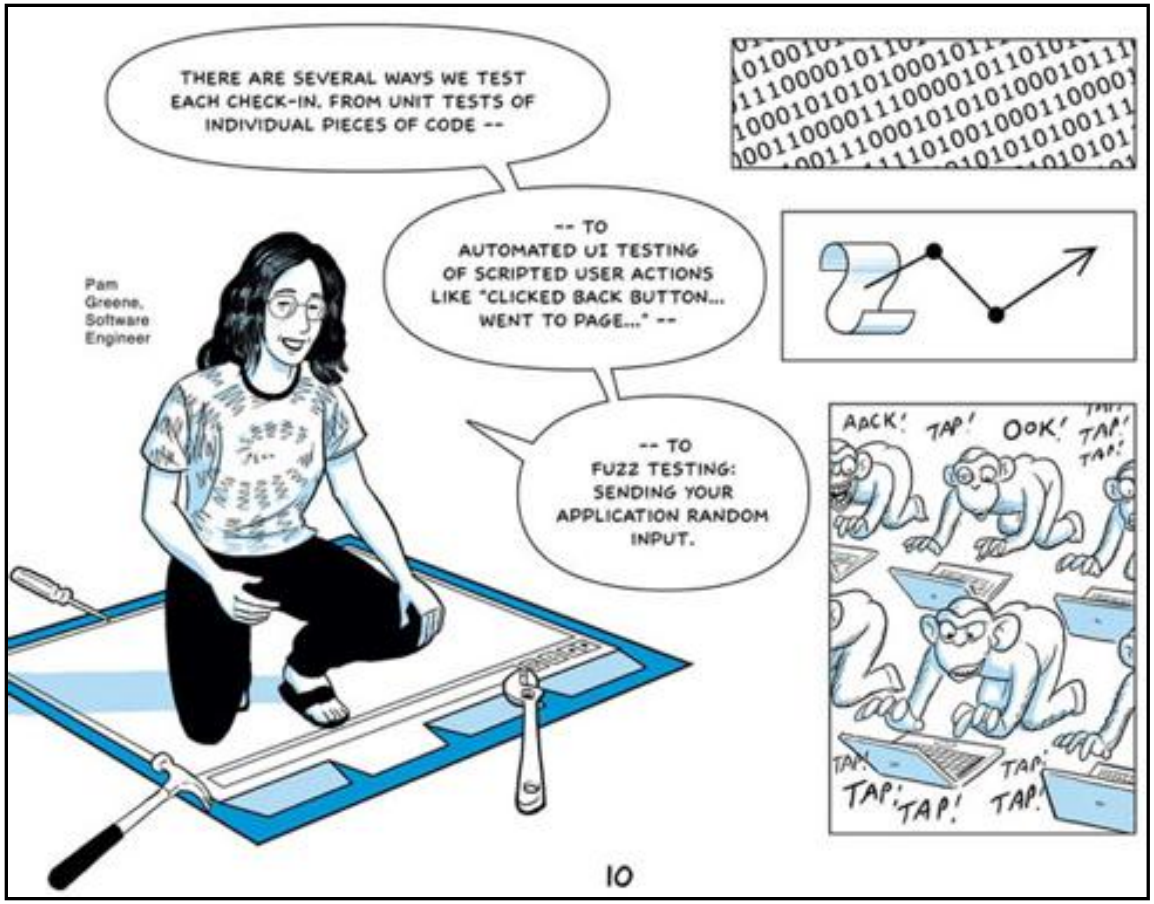
CODENOMICON

Some Helpful Definitions

- **Vulnerability – a weakness in software, a bug**
- **Threat/Attack – exploit/worm/virus against a specific vulnerability**
- **Protocol Modeling – Technique for explaining interface message sequences and message structures**
- **Fuzzing – process and technique for security testing**
- **Anomaly – abnormal or unexpected input**
- **Failure – crash, busy-loop, memory corruption, or other indication of a bug in software**

CODENOMICON

Fuzzing is a NEW way of doing QA. Its robustness, its security! Its here, today. And it works! Lets make sure it is used!



From: http://www.google.com/googlebooks/chrome/

CODENOMICON

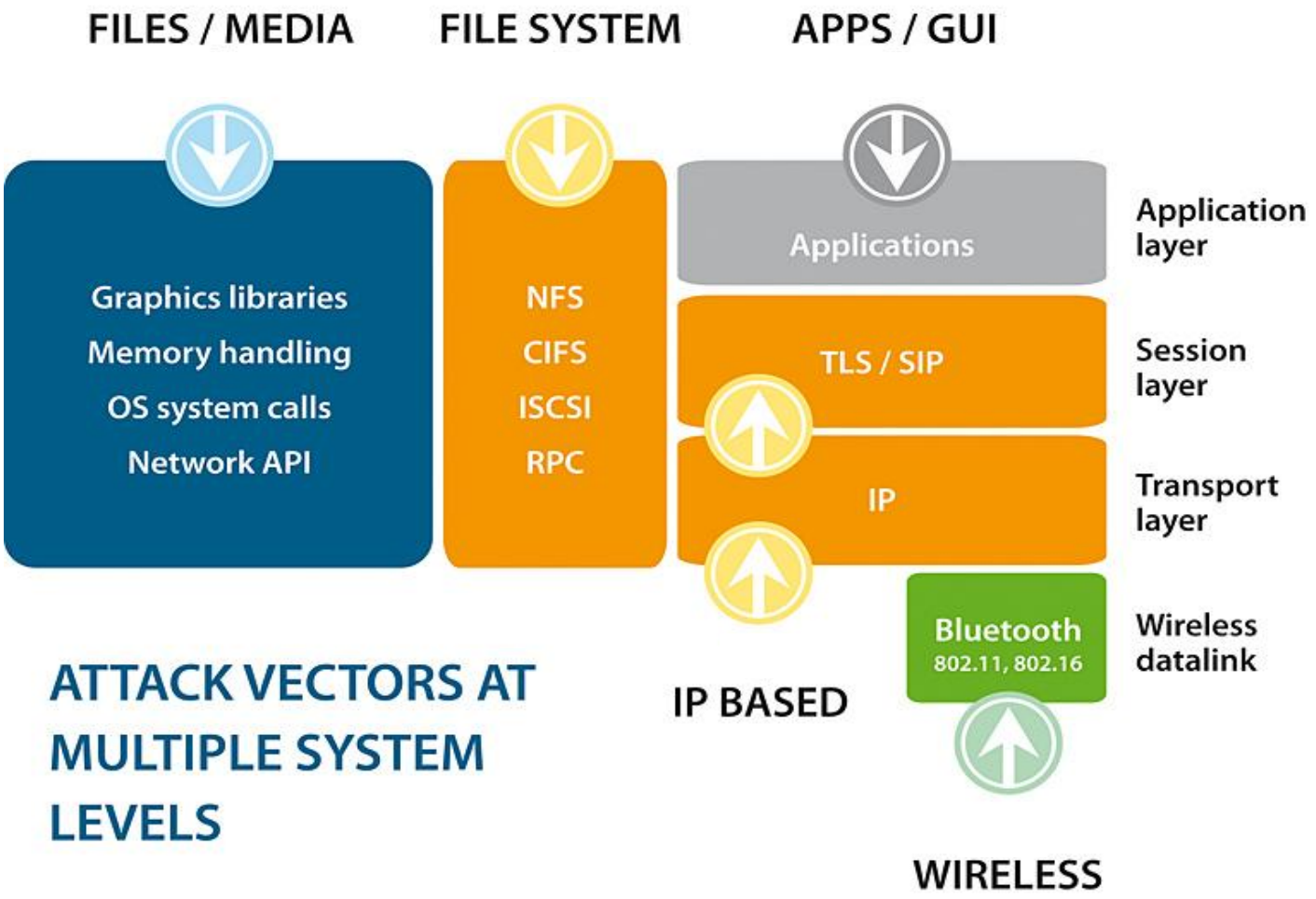
Facts about fuzzing

- **<u>All</u> fuzzing approaches find vulnerabilities and failures. Some more than others.**
- **More fuzzing is usually the better**
- **There is no shortage of tools today – both FOSS and commercial**
- **Fuzzing is right thing to do – and its cheap and cost effective!**
- **[almost] Everyone SHOULD fuzz!**

A bug trying to hide from us

CODENOMICON

Example: Multitude of Attack Vectors / Attack Surface



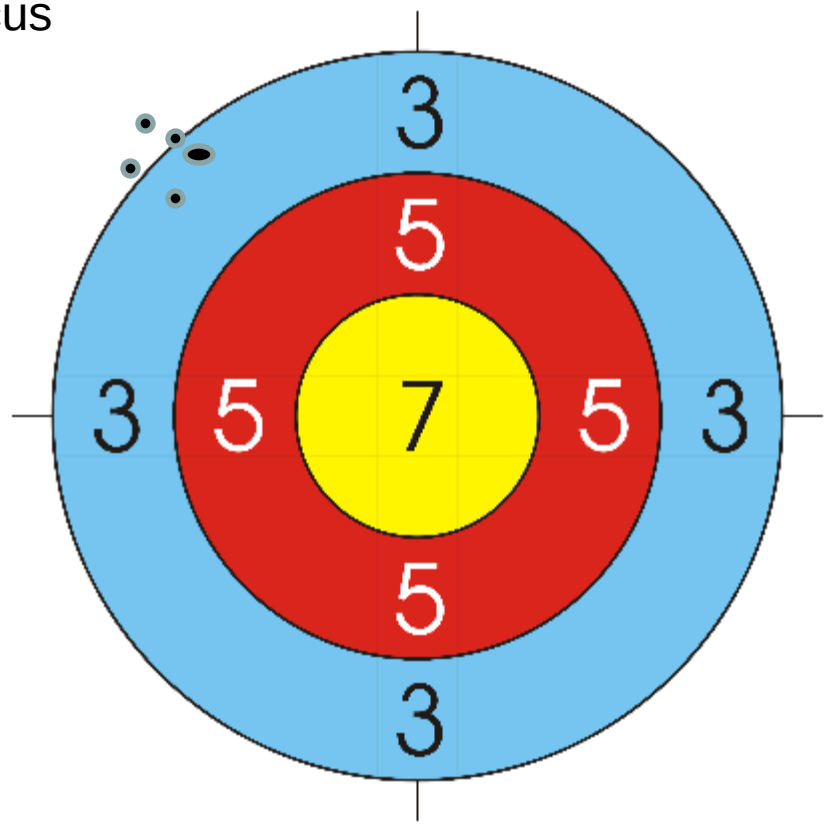
The Greatest Challenge in Fuzzing?

Test Coverage, or how:
Just Getting IT done – affects it?

CODENOMICON

PRECISION

Precision is about focus

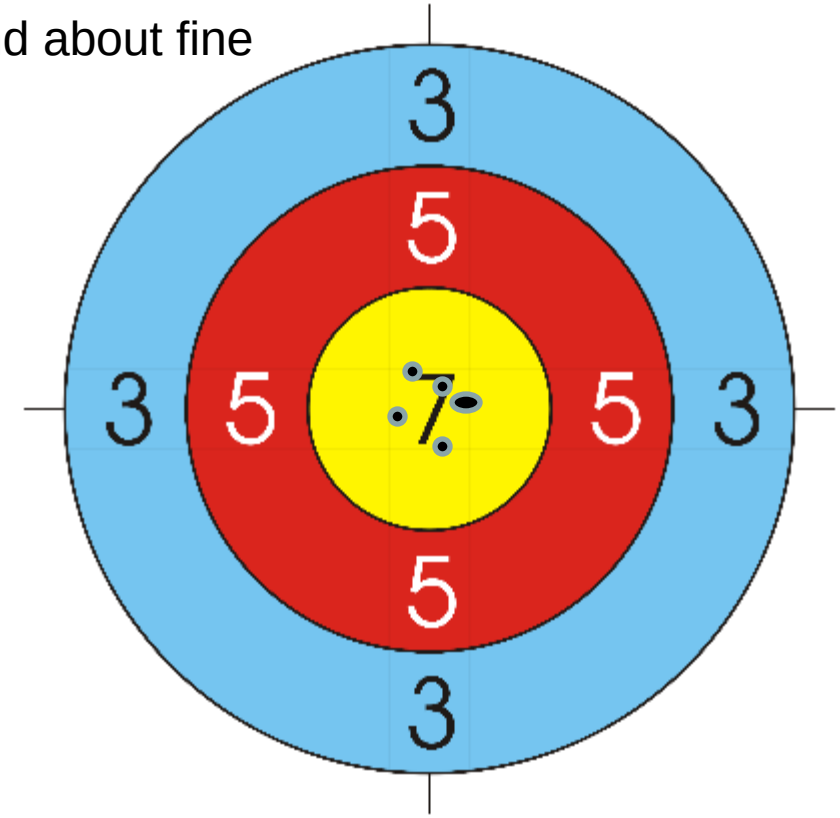


Attack surface
Protocol layers
Protocol use cases
…

CODENOMICON

ACCURACY

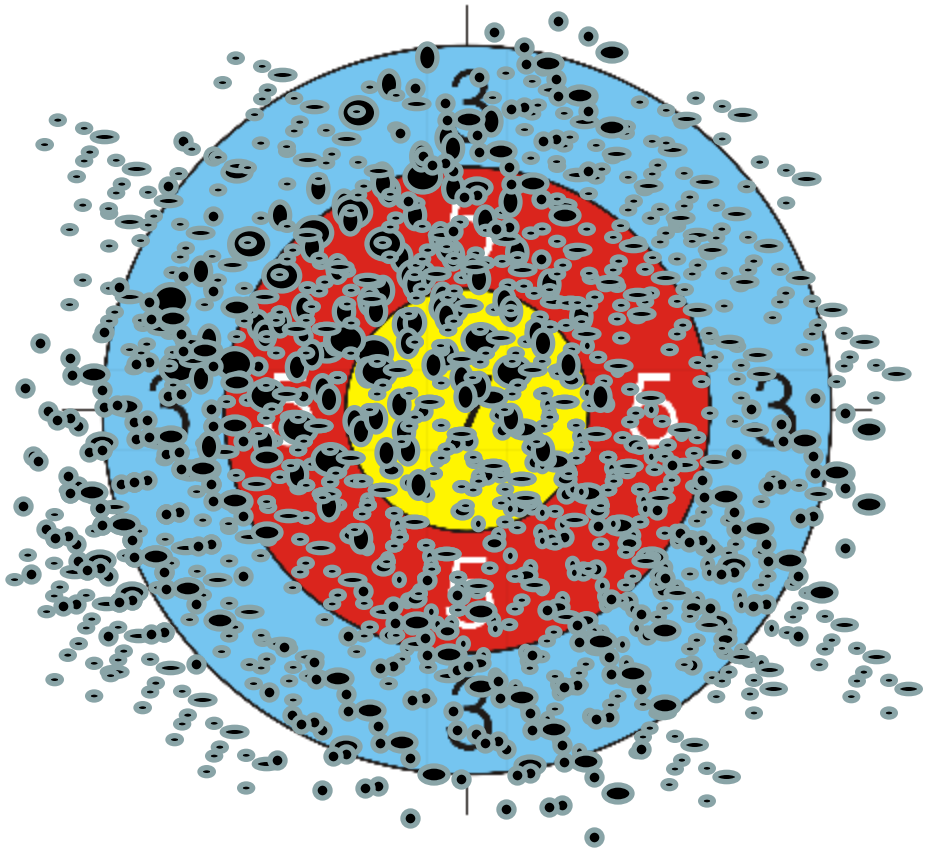
Accuracy is skills and about fine tuning your “tools”



SQL anomalies
ASN.1 anomalies
XML anomalies
Integer anomalies
Structural anomalies
…
Monitoring
…

CODENOMICON

ROBUSTNESS TESTING - FUZZING



CODENOMICON

So whats the problem?

- What we a trying to find is usually microscopically ”small” – so {shot/mini}gunning is usually not the optimum solution
- Various methods are needed to optimize the fuzz... This is what we and others have been doing past 10 years++
- This translates to coverage. Coverage seems to be function of accuracy and precision!
  (in part anyway)

CODENOMICON

What is Fuzzing Today?

- **Fuzzing tests SW/HW by providing anomalous input to communication protocols/interfaces**
- **Attempts to cause the software to fail or behave in unexpected ways**
- **Models communication protocols, APIs, or file formats**
  - Pretty much all fuzzing today is model based (only variable is what the model is based on and how complex it is)
- **Creates large amounts of anomalous input automatically (which leads to some problems debated later on)**
- **Interacts with the software under test and observes the behavior (!!!!)**

CODENOMICON

Some technical (traditional?) methods to assess the coverage

- **By observing the test suite**
  - **Specification coverage of the test suite**
  - **Functional coverage of the test suite**
  - **Engine capabilities** (How the model is build, Evolving, Traffic capture, Specification, How test cases are generated, Systematic, Mutation, Anomaly libraries, Bit-flips,…)
- **By observing the test target**
  - **Code coverage (depth, breadth?) (!!!)**
  - **Binary coverage (e.g. PaiMei, IDAPRo…)**
  - **Number actual unique bugs observed**
  - **(Resource utilization)**
    - **CPU, memory leaks, file handles, hidden exceptions,…**

CODENOMICON

Couple of observations on coverage (accuracy, precision)

- **Affected by several factors outside (often) direct control of the [fuzzing] test suite**
  - **Configuration of DUT**
  - **Execution environment (ripple effect)**
  - **Configuration and capabilities of the test suite** (people just run the default settings)
- **[better, automatic, easier] observation facilities increase the accuracy of the tests**
  - **E.g. use Valgrind to catch memory leaks**
  - **E.g. use OllyDbg, windbg to catch bad use of try {} catch {} to hide exceptions…**
  - **E.g monitoring /var/log/messages :-S**

CODENOMICON

Reality check on technical coverage

- depth-first might be sexier but breadth-first seems to helps in getting better overall results

- Fuzzing multiple layers or fields at the same time may sound fancy, but usually results in significantly poorer results!

CODENOMICON

Accuracy, precision, coverage – beyond technical means – socio-economical coverage

- **It's not only technical challenges that we are facing when trying to get best out of fuzzing**
- **I assert that <u>organizational, educational, processual, usability, etc. aspects of fuzzing ecosystem are bigger challege than technical issues</u>**
  - These challlenges in part also prevent fuzzing being able to realize its potential

CODENOMICON

Accuracy, precision == Coverage == function of usage

- **All fuzzing finds flaws**
  - IF users know HOW to use the tools
  - AND how to read and use the results (big issue with commercial fuzzers)
- **Some reasons for IF/HOW/AND(s) above**
  - As fuzzing goes more main stream, people don't do fuzzing because of its inherent benefits, BUT because they are told to. Which means they don't get it the way we do.
  - Fuzzing is not exact "science" – e.g. results need interpretation, its not always clear what to look for or what happened?
  - Something needs to be done with results – which is not always obvious!

CODENOMICON

Commercial deployments of fuzzing suffer from non technical challenges...

- **For commercial deployments there are few key challenges**
  - Fuzzing is NOT recognized as integral part of modern SDLC
    - Even when it is, there is big confusion on how/where to deploy it (security vs. QA vs. choose)
  - There are NO fuzzing career paths or certifications
  - There are NO compliance requirements that fuzzing would fulfill (CC EAL4?)
  - Fuzzing is NOT taught in colleges, universities,… (at least not in India, China, Indonesia,… where the testing is done)
- **<u>As a result fuzzing is not realizing its full potential!</u>**

(and that has almost nothing to do with technical capabilities of fuzzers)

CODENOMICON

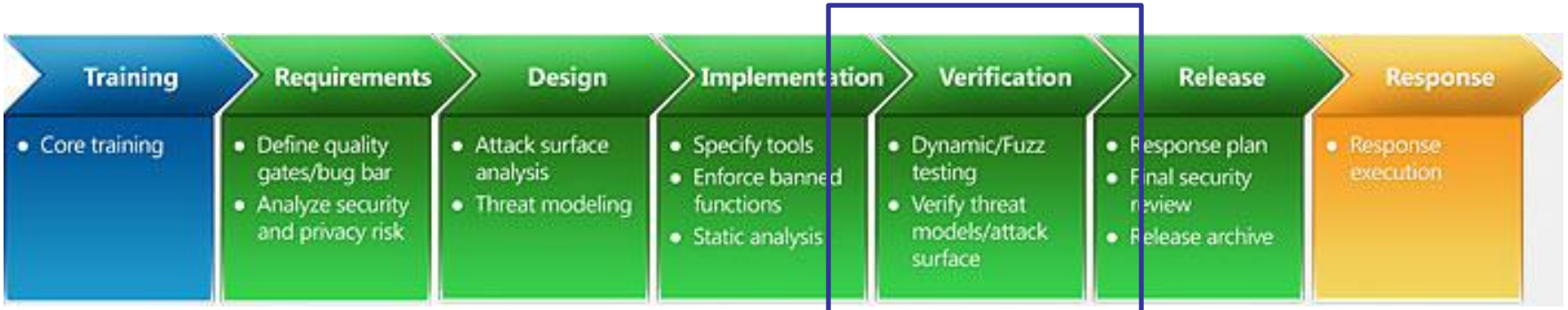
Increasing the coverage of commercial fuzzers (in non technical way)

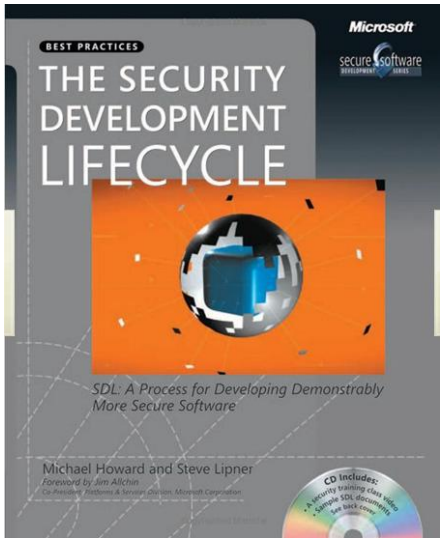
- **Fuzzing added as a compliancy requirement of its own**
  - Likely decreases the actual ”quality”, BUT overall outcome is likely light years ahead of where we are today!
  - Solves the problem that fuzzing is skipped because people are worried about their job security
  - Career paths, certifications (!!!!)
    (you need a permit to hold a gun in most countries, should same be applied to fuzzers? ☺)

- **Education, training, support!!!**
  - What to do with the vulnerability after its found? Fix it, report it (whom, where?), mitigate it, forget about it?
  - To make people aware of fuzzing...

Microsoft SDL Example: Fuzz Here?

http://msdn.microsoft.com/en-us/security/dd219581.aspx





Many organizations choose to deploy fuzzing in other parts of the SDL as well.

CODENOMICON

Microsoft SDL helps in so many ways

- **Well defined SDL with fuzzing presented as an integral part of it**
- **SDL how ever does not define HOW, hence we need:**
  - Provide tools / test suites / instructions / guides / encouragement
  - Provide the baseline test plans, information,..
  - Support with monitoring facilities (Core dumps, Memory leaks, CPU, Other (...))
- **The key question is how fuzzing is done (and what/when is enough)?**
  - **This is both technical and <u>non-technical</u> question!**

CODENOMICON

Summary of part #1

- **Fuzzing may be ready for the prime time, but in order to get there we need *support***
- **We need vendor independent**
  - Best practises and recommendations
  - Fuzzing added as a compliancy requirement
  - Education on fuzzing as part of QA / testing curriculums
- **Fuzzing to be made even easier and more accessible to the end users**

CODENOMICON

Case 1: XML Fuzzing – more coverage by fuzzing more layers

XML Introduction, XML Vulnerabilities, XML Fuzzers

http://www.codenomicon.com/defensics/xml

CODENOMICON

Case: XML

- **Early 2009, Codenomicon developed fuzzers for XML-based telecommunication protocols**
- **All open source XML parser libraries failed**
  - All applications using these libraries are vulnerable for the very same issue

- ***Note that this is very similar to the PROTOS/OUSPG ASN.1/SNMP discovery… Covering potentially thousands of bugs… ASN.1/SNMP bugs were never really fixed either… Codenomicon SNMP suite still today crashes all commercial SNMP implementations***

CODENOMICON

Wait a Sec! What is XML?

- XML is not a protocol
- XML is a syntax for messages, similar to ASN.1 in binary messages

```
<?xml version='1.0'?>
<methodCall>
  <methodName>AttachFile</methodName>
  <params>
    <param>
      <value><string>FrontPage</string></value>
    </param>
    <param>
      <value><string></string></value>
    </param>
    <param>
      <value><string>list</string></value>
    </param>
    <param>
      <value><string></string></value>
    </param>
    <param>
      <value><boolean>0</boolean></value>
    </param>
  </params>
</methodCall>
```

CODENOMICON

XML Is Used In

- **XML used by IETF, 3GPP, W3C and commercial vendors**
- **XML-based standard protocols**
  - SOAP, XMPP, CWMP (TR-069), UPnP, NETCONF, SIP/IMS, SyncML
- **XML file formats**
  - HTML/XHTML, XSLT, RSS/Atom, WAP/WML, SAML, SMIL, Office applications
- **Proprietary protocols and file formats**
  - IBM, Microsoft, SAP, Oracle, BEA etc.
- **Custom application logic built on XML messaging using Web Services platforms (Various vendors)**

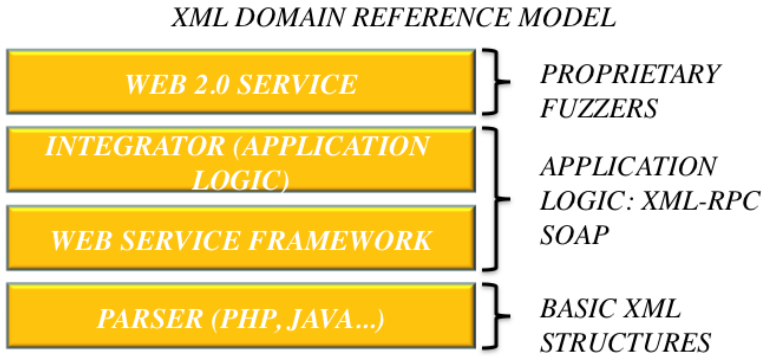
CODENOMICON

XML: The Discovery

- **A single exploit can be used to attack:**
  - Security products such as Firewalls
  - Application platforms
  - Programming languages: PHP, Ruby, Python, C, Java
- **Repair process:**
  - Fix the parser libraries
  - Deploy updates to operating systems and platforms
  - Urge application developers to re-build if needed

XML: A Look At The Test Coverage

- **XML-based systems**
  - Complex and highly critical applications
  - Built in “layers”
  - Parsers, SOAP and XML-RPC applications



- **Fuzzers for XML**
  - Advanced model-based fuzzing capability required for testing anything above parser-level
  - Customized fuzzers needed for application layer

CODENOMICON

Coverage

- Precision?
- All messages tested?
- All message structures tested?
- All data definitions tested?
- All “tags” tested?
- Precision seems like it is about protocol coverage

- Accuracy?
- Anomaly categories? SQL? Buffer overflow?
- All values: 0..65k, a..z, 0x00..0x255 ?
- Combinations of anomalies?
- Accuracy seems like it is about anomaly coverage

CODENOMICON

Templates-Based General Purpose Fuzzing

-- more coverage via less precision and accuracy

Traffic Capture Fuzzing (==fuzzing for the masses?)

http://www.codenomicon.com/defensics/traffic-capture-fuzzer

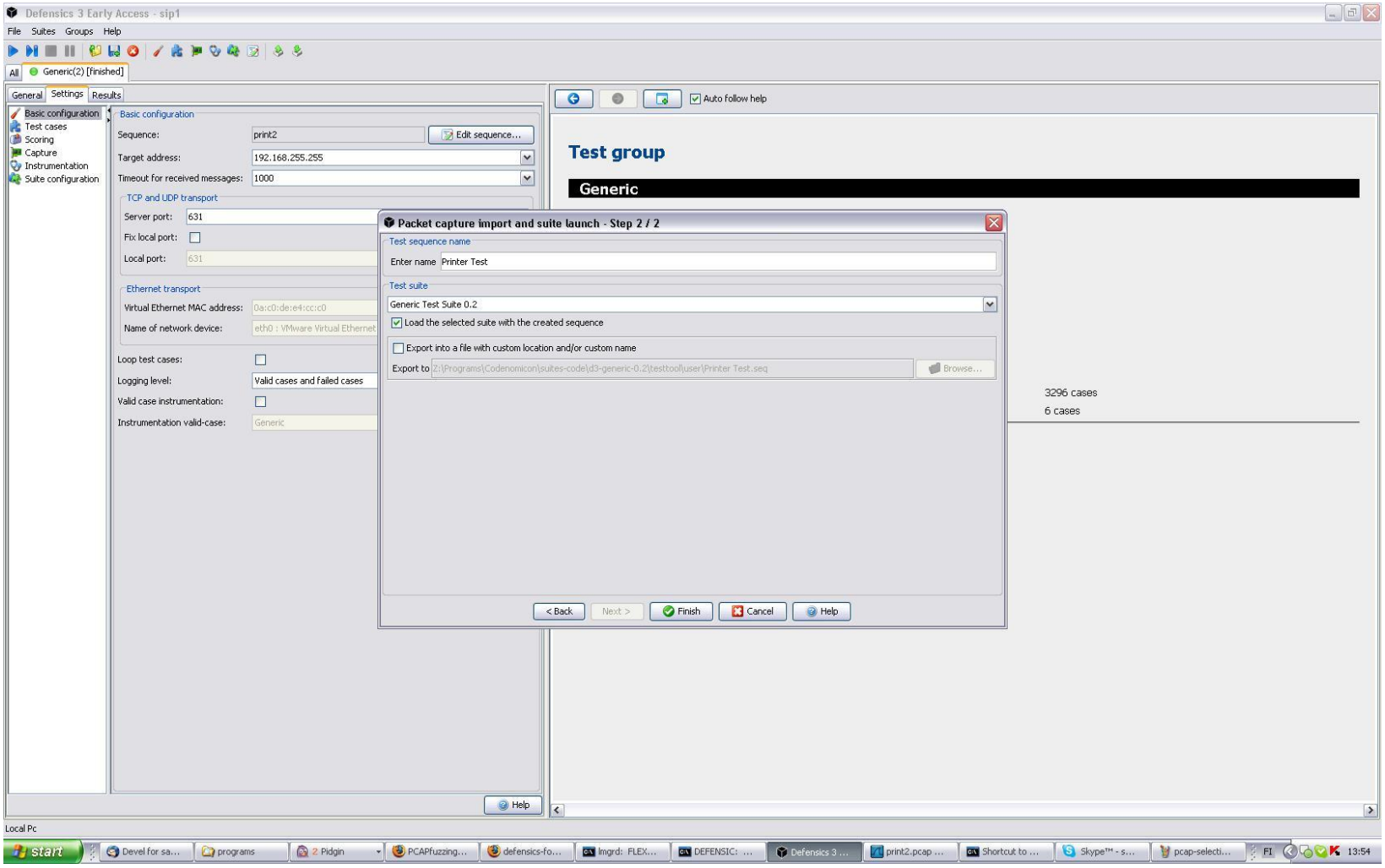
CODENOMICON

Traffic Capture Fuzzing

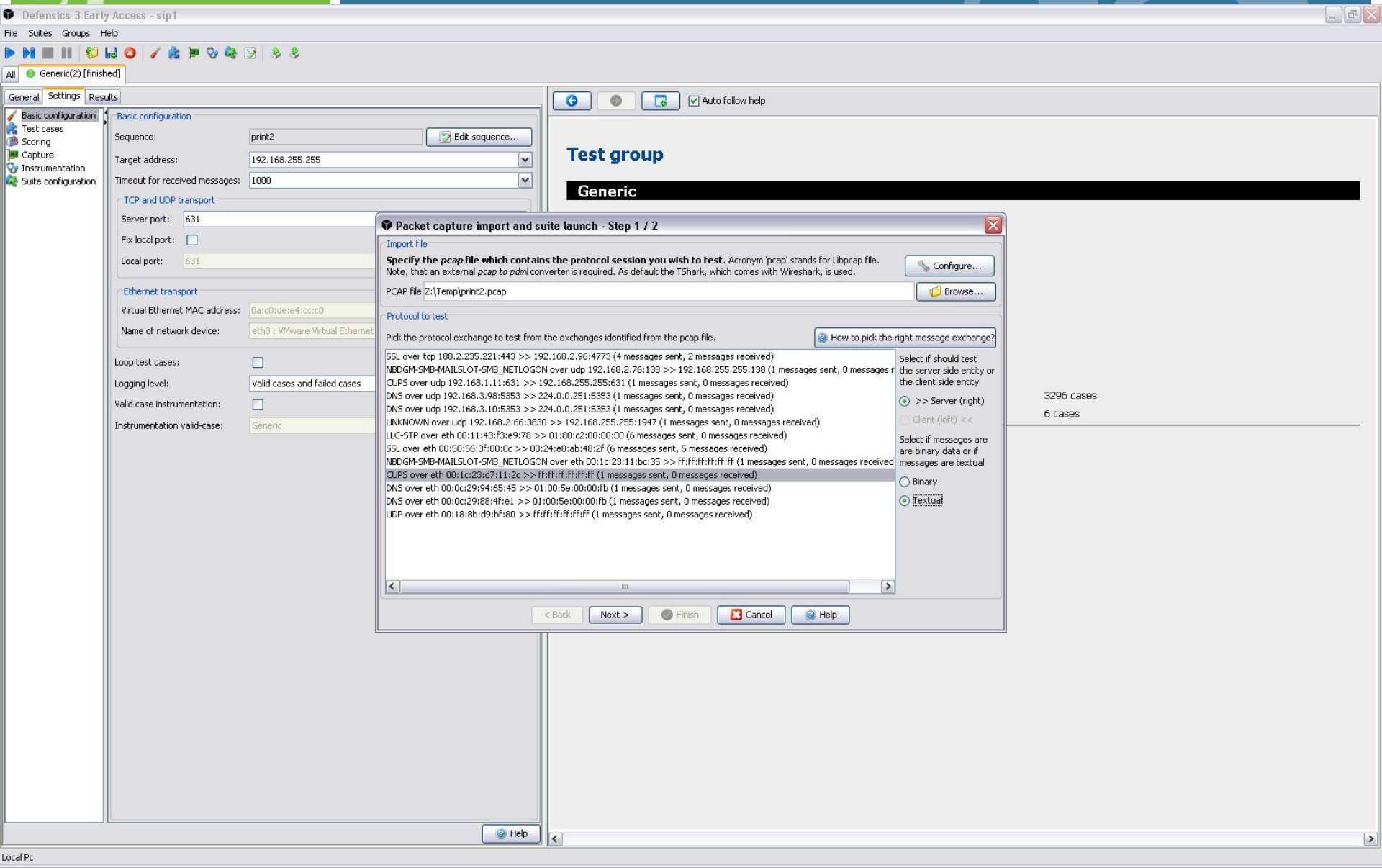
- **All fuzzing needs a model of correct behavior**
- **The easiest method for acquiring default functionality is from templates**
  - Files
  - PCAP traffic flows
- **The model is easily built by e.g. Wireshark protocol dissectors**
  - Open source has had this a while (autodafe, peach,…)
  - Commercial fuzzers following, with a twist…

CODENOMICON

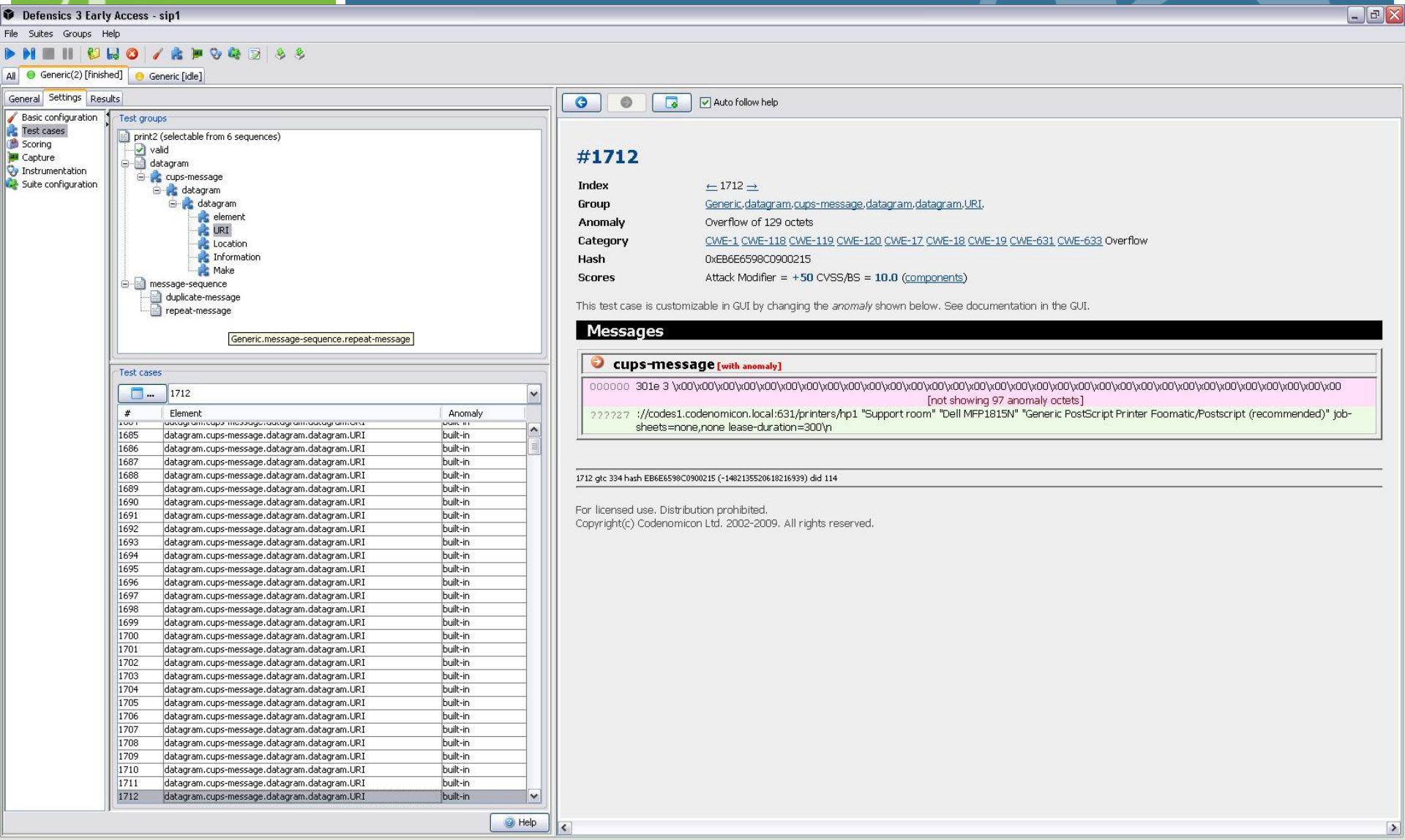
Acquire traffic capture from analyzers, vulnerability feeds or bug reports



Protocol (and protocol layer) selection from a set of protocol captures



Protocol model and test cases are automatically created



CODENOMICON

Model-based vs Template-based

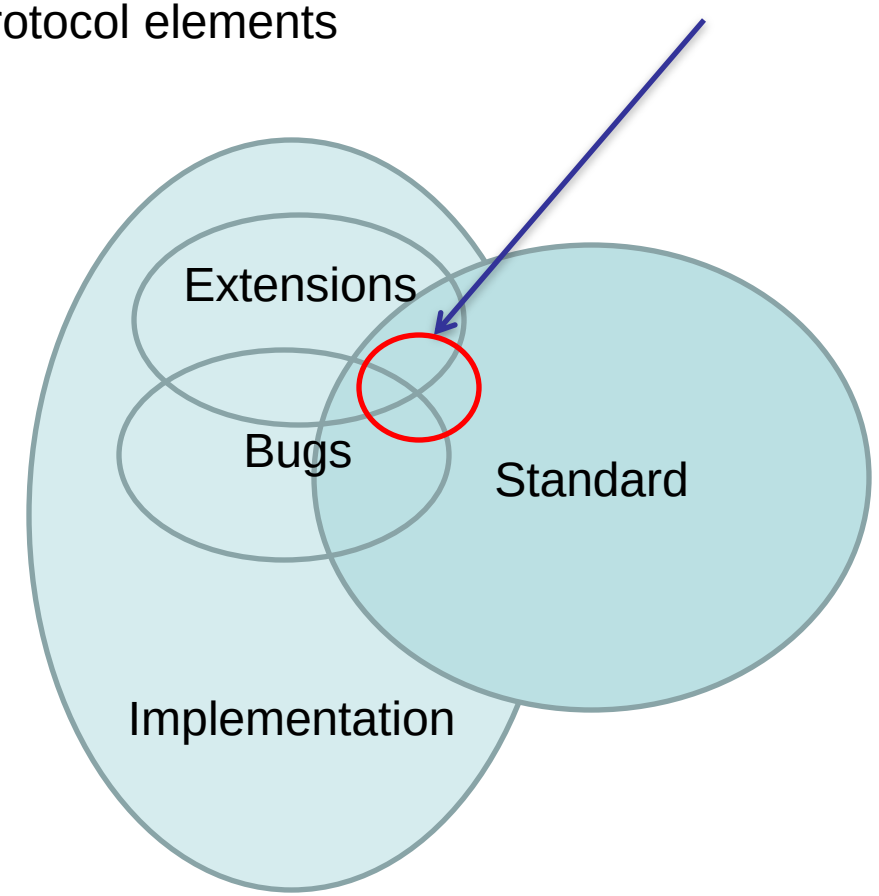
- **Benefits of Model Based Fuzzing**
  - Full test coverage (all elements, all anomaly categories)
  - Short test cycle
  - More optimized tests
  - Easy to edit and add tests to an easy to understand model

- **Template Based Fuzzing**
  - Quality of tests is based on the used seed
  - Covers only visible protocol elements
  - Blind sets of anomalies (if no meta-data on fields)
  - Very quick to develop, but slow to run
  - Editing requires deep protocol know-how

CODENOMICON

CAPTURE-BASED TESTING

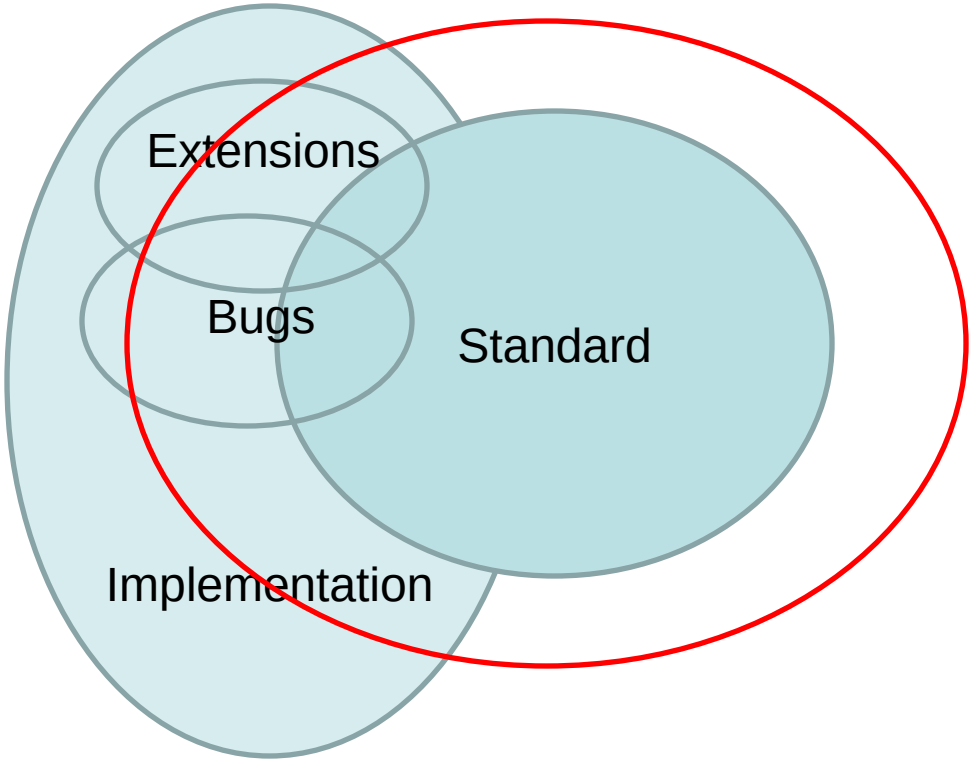
Claims in the industry:
Traffic capture fuzzing tests all used protocol elements



CODENOMICON

MODEL-BASED TESTING

Claims in the industry:
Specification-based fuzzing tests all used protocol elements



CODENOMICON

COMPARISON SUMMARY

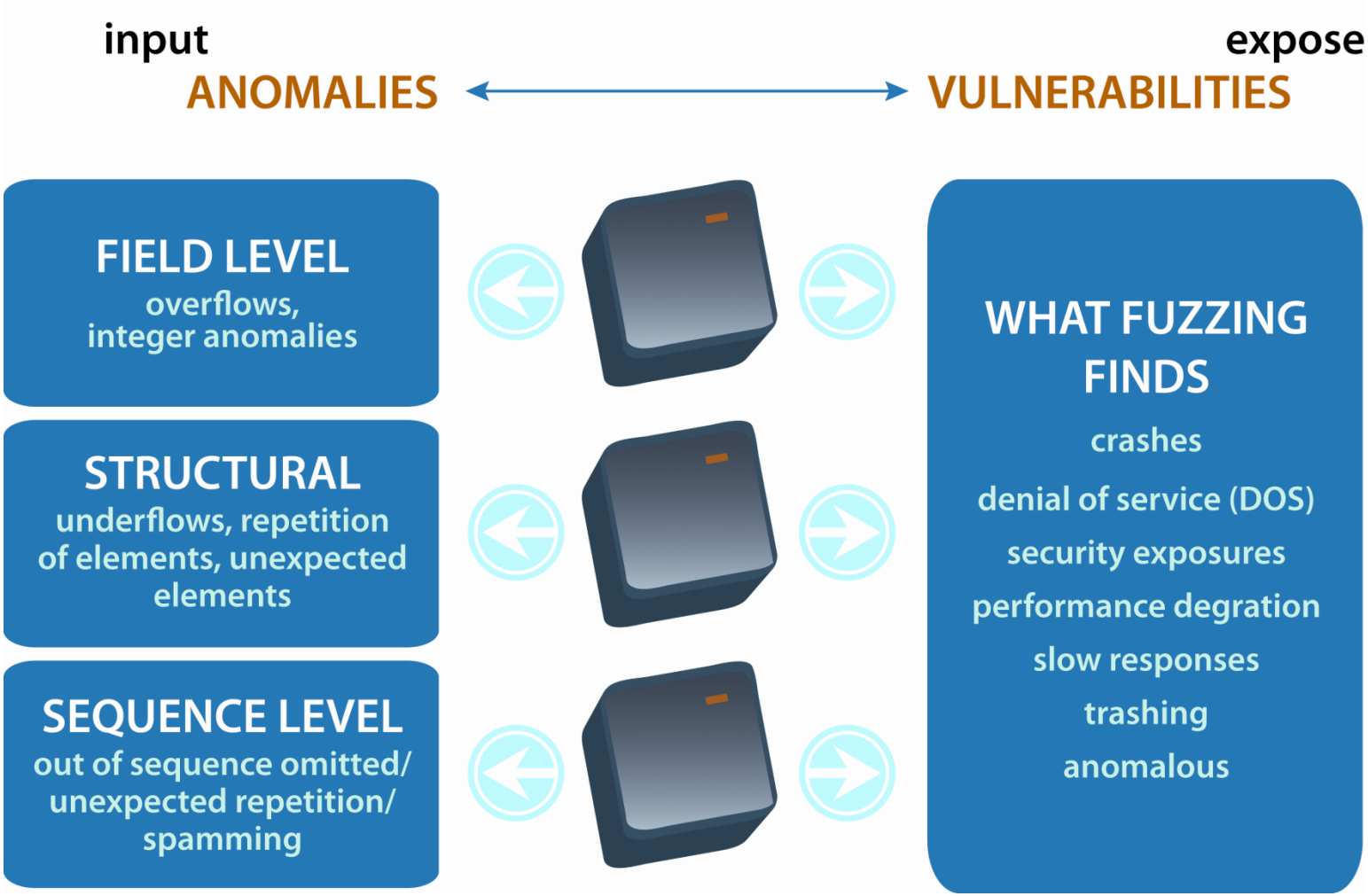
- **Capture fuzzing is not substitude for (specification) model based testing**
  - Valuable as entry level solution when model-based fuzzer is not available
  - Model based fuzzer can be adapted to proprietary extensions
  - Capture based fuzzer can't be taught rarely used elements specified by standard (*

(Security problems are often found from the attack surface parts which are not usually covered in day-to-day traffic. Bugs are there because those parts of the code are usually less tested and reflect rarely needed portions of a protocol.*

However, it does not matter if a vulnerability is in unusual interface surface, the system is still 100% vulnerable.

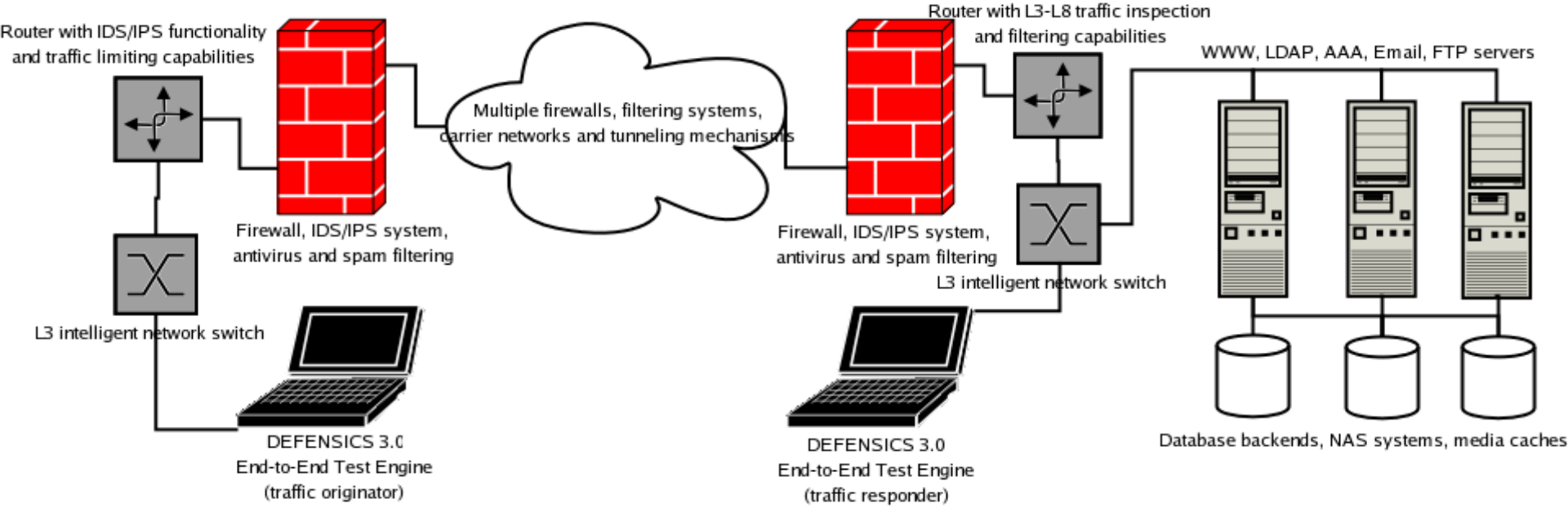
CODENOMICON

Challenge: Fuzzing: The Anomalies



Fuzz the Entire System (E2E)

- **Fuzz Testing needs to be conducted for the entire system, not just one layer on one interface**



THANK YOU – QUESTIONS?

"Thrill to the excitement of the chase!
Stalk bugs with care, methodology, and reason. Build traps for them.
....
Testers!
Break that software (as you must) and drive it to the ultimate
- but don't enjoy the programmer's pain."

[from Boris Beizer]

CODENOMICON

PROACTIVE SECURITY AND ROBUSTNESS SOLUTIONS