KNOWN VULNERABILITIES

What you can find with your current security tests.

Anton von Troyer

UNKNOWN VULNERABILITIES

What Codenomicon can help you reveal.



About Codenomicon

Founded in Autumn 2001

Commercialized the academic approach built since 1996

Technology leader in security test automation

- Model-based, intelligent, targeted tests
- Software-based, with subscription licensing
- Horizontal market, supports 150+ technologies
- 200+ customers globally

State-of-the art Network Analysis and Security assessment services

Highly successful, constantly growing

- 2009 was 9th subsequent growth year!
- Profitable year 2009! §



Glossary of Terms: Product Security Terminology

Vulnerability – a weakness in software, a bug. Most

software has bugs

Threat/Attack –

exploit/worm/virus against a specific vulnerability

Protocol Modeling – Technique

for explaining interface message sequences and message structures

Anomaly – abnormal or unexpected input

Failure – crash, busy-loop, memory corruption, or other indication of a bug in software





Robustness Testing

Discover unknown vulnerabilities with Testing & QA





Definition of fuzzing

Fuzzing is a technique for

- intelligently and
- automatically
- generating and passing into a target system
 - valid and
 - invalid

message sequences to see if the system breaks, and if it does, what it is that makes it break.



Model Based Fuzzing Techniques

Specification Based Fuzzing

- Full test coverage
- Always repeatable
- Short test cycle, more optimized tests
- Easy to edit and add tests



Template Based Fuzzing

- Base test cases on a sample
- Quality of tests is based on the used seed and modeling technique
- Very quick to develop
- No need for protocol specification

CODENOMICON Codenomicon is Riding the Wave: The Last 8 Minutes of Fuzzing

Fuzzing = Security Testing

- 1990 Random fuzzing becomes popular. Hackers use Fuzzing for zero day discovery;
 1999 Model-based fuzzing becomes popular. 100% of zero-days found with fuzzing;
- **2001** Network equipment manufacturers start using Fuzzing for protection against hackers
- **2006** Telcos integrate Fuzzing into acceptance testing and test for zero-day threats
- 2010 Large-scale propagation of Fuzzing at
- Finance
- Government
- SCADA

2015-> Fuzzing as a purchase criteria and requirement by GOV





Fuzzing means crash testing... ANY TYPE OF DEVICE!





Why to fuzz?

Fuzzing a popular method also among hackers Fuzzing mimics the ways attackers search for security holes

Most exploits/zero days nowadays found by fuzzing



Unknown Vulnerability Management: Be Proactive with Security

- Modern security testing is about finding unknown zero-day vulnerabilities in devices and software before and after release
- Provides a quick technique for security assurance for any device or software

www.codenomicon.com/unknown/





Benefits of fuzzing

"All software has undetected exploitable vulnerabilities" - Security Vendor 2009

"You would be a fool not to Fuzz." – Forrester 2011

"All our zero-day vulnerabilities were found with Fuzzing."– Software Vendor 2010





Benefits of fuzzing

 "The Codenomicon tools are *amazing*. Using them is like being attacked by the most relentless adversary who uses every possible method to find flaws in your code

We fixed subtle crash bugs that had been in our code for over ten years. We would *never* have found those bugs without the Codenomicon tools.

If you're serious about implementing protocols correctly, you need the Codenomicon tools."



Defensics 10

File Suites Groups Utilities Help	Defensics 4 - http10-10-3-33snmp							
 HTTP-Server [de] I Basic configuration 2 Interoperability with Sur 3 Advanced configuration 4 SUIT instrumentation 5 Test case selection 6 Test run 7 Test cases 8 Test cases 9 Test cases 	File Suites Groups Utilities Help							
All P Hilp-Server (die) I Basic configuration I Basic configuration I Test groups I TTP request and response (selectable from 4 sequences) I Test cases selection I Test cases selection I Test cases		🥖 📩 連	🏷 🏘 📝					
✓ 1 Basic configuration ✓ 1 Default suite settings 600k cases ✓ 3 Advanced configuration ✓ 4 SUT instrumentation ✓ 5 Test case selection ✓ 6 Test run ✓ Results Test cases ✓ Test cases ✓ 1 Benent ✓ 2 Benent	All 🗧 HTTP-Server [idle]							
 2 Interoperability with SUT 2 Interoperability with SUT 3 Advanced configuration 4 SUT instrumentation 6 Test run 6 Test run 7 Results Test cases selection Valid Test cases Test cases Test cases Test cases Number of the sequences Test case selection Test case selection Test cases Test cases Test cases Number of the sequences Test cases Test cases Test cases Number of the sequences Test cases Number of the sequences Test cases Number of the sequences Number of the sequences Test cases Number of the sequences Number of the sequen	🖋 1 Basic configuration	Test groups	Test groups Default suite settings 600k cases Image: Product and response (selectable from 4 sequences) Image: Product and response (selectable from 4 sequences)				🥝 GUI Help 🔞 Su	
3 Advanced configuration I + JTTP request and response (selectable from 4 sequences) I + gular-payload </td <td>2 Interoperability with SUT</td> <td></td> <td>PVLACE</td>	2 Interoperability with SUT						PVLACE	
* 4 SUT instrumentation ************************************	3 Advanced configuration	HTTP re					S REVP RTCP R	
Image: S Test case selection Image: Defensics 3 T Image: Results Image: Test cases Image: Test cases Image: Test cases view let Image: Test cases Image: Test cases Image: Test cases Image: Test cases <td< td=""><td>● ④ 4 SUT instrumentation</td><td>thur ⊡≣ chur</td><td>nked-payload</td><td>TCP Tenne</td><td colspan="2"></td></td<>	● ④ 4 SUT instrumentation	thur ⊡≣ chur	nked-payload	TCP Tenne				
6 Test run	5 Test case selection						Defensics 3 T	
Results main - 1 est groups Basic operation The test cases view let Test cases The test cases view let # Element 0 regular-payload.valid 0 regular-payload.valid 0 regular-payload.valid 0 select val 0 select val 0 select val 0 select val	6 Test run							
Test cases The test cases view let Image: Test cases Image: Test cases Image: Test cases Image: Test cases <	Results					<u>n</u>	<u>iain</u> - Test groups	
Test cases *.valid * # Element 0 regular-payload.valid 0 regular-payload.valid none 614454 chunked-payload.valid none The test cases view let Typical flow of actions 1. Run valid test ca Select val Select val Start the test cases view let						Basic operatio		
Test cases Typical flow of actions *.valid # Element 0 regular-payload.valid 614454 chunked-payload.valid none Select val Start the t						The	test cases view let	
*.valid * # Element 0 regular-payload.valid 614454 chunked-payload.valid none • Select val • Start the t		Test cases				Typic	Typical flow of actions	
# Element Anomaly 0 regular-payload.valid none 614454 chunked-payload.valid none Select val Start the total			*.valid			•	Due velid test	
0 regular-payload.valid none • Configure 614454 chunked-payload.valid none • Select val • Start the t		#	Element		Anomaly	1.	Kun valid test ca	
614454 chunked-payload.valid none Select Val Start the 1		0	regular-payload.valid		none		Configure	
		614454	chunked-payload.valid		none		 Select val Start the t 	