



Codonomicon whitepaper:

How to integrate FUZZING and security testing into SDLC

1

Introduction to fuzzing

By Heikki Kortti, Codonomicon

2

End user perspective to fuzzing

By Jon Oltsik, Enterprise Strategy Group

3

Security Test Product of the Year 2008

By Juan Rosales, research analyst, Frost and Sullivan

CODENOMICON Ltd. | info@codenomicon.com | www.codenomicon.com

Tutkijantie 4E | FIN-90570 OULU | FINLAND | +358 424 7431
10670 North Tantau Avenue | Cupertino, CA 95014 | UNITED STATES | +1 408 252 4000

1 Introduction to fuzzing

By Heikki Kortti, Codenomicon

Fuzzing or fuzz testing means sending malformed or invalid inputs to a software, device or system in order to find critical flaws and vulnerabilities. During the past 10 years, fuzzing has become increasingly popular as a low-cost but highly effective way of hardening implementations against external attacks.

Fuzzing enables software testers, developers and auditors to easily find defects that can be triggered by malformed inputs via external interfaces. This means that fuzzing is able to cover the most exposed and critical attack surfaces in a system relatively well, and identify many common errors and potential vulnerabilities quickly and cost-effectively.

Fuzzing is especially useful in analyzing black-box systems, as it does not require any access to source code. Having access to information such as source code, design or implementation specifications, debugging or profiling hooks, logging output, or details on the state of the system under test or its operational environment will help in root cause analysis of any problems that are found, but none of this is strictly necessary.

Fuzzing is often compared to code auditing and other white-box testing methods. While code auditing is another highly valuable technique in a software tester's or developer's toolbox, code auditing and fuzzing are really complementary to each other. Fuzzing focuses on finding some critical defects quickly, and the found errors are usually very real. With fuzzing, there are no false positives.

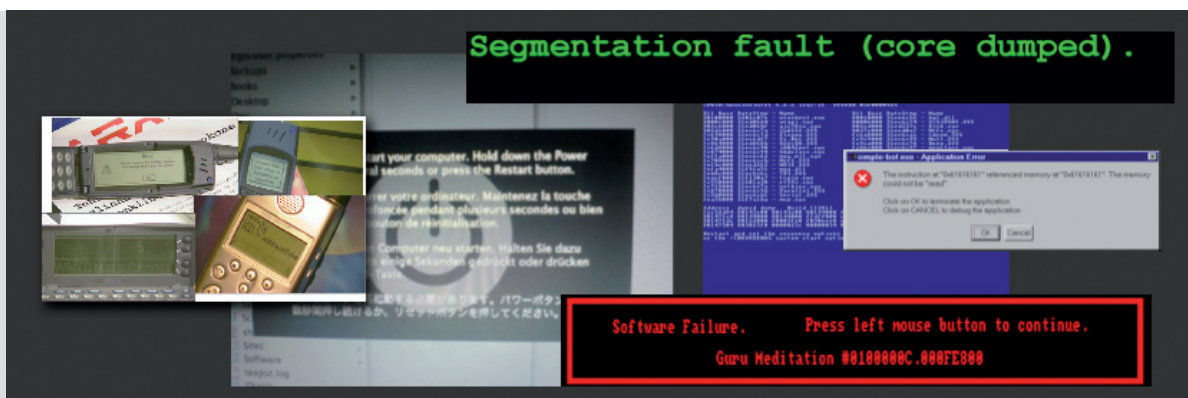
Some fuzzers are implemented as fuzzing frameworks, which means that they provide an end-user with a platform for creating fuzz tests. Fuzzing frameworks typically require a considerable investment in time and resources to develop tests for a new interface. If the framework does not offer ready-made test data for common structures and elements, efficient testing also requires considerable expertise in designing inputs that are able to trigger faults in the tested interface.

In contrast to fuzzing frameworks, another category of fuzzers consists of test suite-based fuzzers. These package a set of tests that have been pre-generated with fuzzing methods into a test suite or test tool that can be used in actual testing. Usually this type of fuzzing requires only minimal work from the end-user (tester), as the interface model as well as test case definitions have already been created beforehand. The tester needs to configure only a few basic settings to start running the tests, and does not even need to fully understand the specifications or other details of the tested protocol or file format.

Another dimension for comparing fuzzers stems from whether they are model-based or not. Compared with a static, non-stateful fuzzer which may not be able to simulate any protocol deeper than an initial packet, a fully model-based fuzzer is able to test an interface more completely and thoroughly, usually proving much more effective in discovering flaws in practice. Tests executed by a fully model-based fuzzer are usually able to penetrate much deeper within the system under test, exercising the packet parsing and input handling routines extremely thoroughly, and reaching all the way into the state machine and even output generation routines.

Security needs to be integrated into every step of the software development life-cycle (SDLC). Neither fuzzing nor code auditing is able to provably find all possible bugs and defects in a tested system or program. As a rule of thumb, the effectiveness of fuzzing is based on how thoroughly it covers the input space of the tested interface (input space coverage), and how good are the representative malicious and malformed inputs for testing each element or structure within the tested interface definition (quality of generated inputs). Fuzzers which use templates or network captures as the model will reach very low input space coverage. Fuzzers which populate their tests with random or semi-random data will have bad quality of inputs and can have significant number of meaningless tests.

For further information on fuzzing, check out: <http://www.codenomicon.com/products/buzz-on-fuzzing.shtml>



2 End user perspective to fuzzing

By Jon Oltsik, Enterprise Strategy Group

Fuzzing has proven to be particularly effective when testing for software exposure and system failure risks because it:

- Has broad application. Since black box fuzzing is independent of individual development projects, it can be applied in numerous use cases. For example, fuzzing can be used to analyze the behavior of file formats, networking and application protocols, APIs, etc.
- Provides “wide and deep” coverage. Leading fuzzing tools support a wide range of protocols from Layer 2 through Layer 7 of the OSI stack. This in itself is valuable, but many offer extensive test suites to exercise all aspects of these protocols. Many users have found problems in secondary esoteric protocols that impact overall software security, performance, and availability. With this “wide and deep” capability, black box fuzzing tools can test each and every protocol, not just the obvious ones.
- Eliminates the need for source code and protocol knowledge. Most test engineers would agree that more testing is always better, but many organizations don’t have the resources, time, or skills to develop hundreds of new scripts to test software on their own. Fuzzing tools eliminate this restriction by aggregating testing expertise and a multitude of test routines into a turnkey solution. With this type of coverage, software engineers can spend their time testing code rather than studying protocols or developing their own test scripts.

The following findings are based on interviews conducted with users of commercial fuzzing tools.

The key driver in fuzzing is the capability to extend testing into new domains, namely security testing, and through a turnkey solution:

- “There is so much going on in the operating system in terms of protocol support and we wanted to make sure that we understood how these protocols impacted the security and robustness of our software. Black box testing gave us a much broader use case.” (Software Company)
- “Originally, we found some unstable behavior with some protocols when using Nessus for testing purposes. We adopted [fuzzing] tools soon afterward, because we wanted to exercise more protocols through a thorough set of test cases. [Fuzzing] tools certainly fit this requirement. (Telecommunications Equipment Company)
- “We needed protocol specific tests and didn’t have the resources to develop our own test for CIFS, RTP, SSL and lots of others. [The commercial fuzzing tool] gave us support for every protocol we needed.” (Software Company).

What criteria do the customers use when choosing the fuzzing product? Vendor reputation and the test coverage came up high on the list:

- “Some of our team members had worked with Codenomicon before and suggested that we try it. I can’t tell you how important these types

of personal references are when you are about to purchase and use something you have little experience with.” (Software Company).

- “The most important thing is this: Codenomicon’s protocol depth and state is the best in the market. The documentation is also awesome; you can see every test case.” (Telecommunications Equipment Company)
- “We really thought we would use open source tools, but the more protocols we decided to test, the more work we had to do. Codenomicon supports almost every protocol we wanted to test, so it made economic sense to purchase DEFENSICS rather than spend dedicated time and resources toward customizing open source.” (Software Company)

People also like using software based solutions compared to appliance based tools. Software will be easier to use corporate-wide, and results in more users and better results in a shorter time frame.

- “We like the idea of a software-based system. We can easily modify the system with new operating systems, more network cards, or more complex configurations. We are also able to select preferred hardware vendors that have established good SLAs.” (Telecommunications Equipment Company)
- “Since Codenomicon is a software-based system, we were able to purchase DEFENSICS using our operating rather than capital budget. This made the decision easy.” (Telecommunications Equipment Company).

Finally, the ultimate purpose of fuzzing is to find flaws. The value of fuzzing is in the proactive elimination of security critical issues in software.

- “Based upon our models, we know that if customers find software bugs, it cost about \$32,000 to fix. If we catch a bug during our software verification phase, it costs between \$4,000 and \$8,000 to fix. If we find software bugs in development, it only costs around \$600 to \$800 to fix them. Obviously, our goal is to find bugs as early as we can, and Codenomicon is helping us do this.” (Telecommunications Equipment Company)

In summary, black box testing is a simple matter of doing more with less. In this case, commercial fuzzing tools provide more test suites, protocol support, and fuzzing capabilities while decreasing the need for deep protocol knowledge, test suite development, and source code expertise.

For more customer comments on fuzzing, check out:

<http://www.codenomicon.com/resources/whitepapers/2008-customer-view.shtml>

“Why did we adopt [fuzzing] tools?

That’s easy – our customers told us to. We are finding this true of more and more of our carrier customers.”

- Telecommunications Equipment Company

3 Security Test Product of the Year 2008

By Juan Rosales, research analyst,
Frost and Sullivan

The 2008 Frost & Sullivan Award for Product of the Year in the World security test market is presented to Codenomicon Ltd. (Codenomicon) in recognition of DEFENSICS 3.0, the most recent version of the company's flagship software-based test platform targeted at developers, service providers, and enterprises who deal with a variety of interfaces in the network equipment they make or utilize in their networks. Providing preemptive security and robustness testing for Internet, wireless and digital media systems, DEFENSICS 3.0 covers over 140 network protocols, digital media formats, and wireless interfaces, allowing for fast test execution. Extremely user-friendly, DEFENSICS 3.0 features a new and improved graphical user interface (GUI) that enables technicians to control multiple test runs from a single interface. The new platform also allows for easy migration from version 2.0, with each important test tool receiving a full update. Furthermore, DEFENSICS 3.0 has improved upon the test coverage capabilities of its preceding versions, allowing the end users to alter their test cases based on time and priority while producing custom tests. As a result, DEFENSICS 3.0 is poised to be a premier solution for robustness testing in security test applications.

Headquartered in Oulu, Finland, Codenomicon markets its testing software and services directly and through international partners. Codenomicon's customers include Alcatel-Lucent, AT&T, Cisco Systems, F5 Networks, Nordea, Nortel, Microsoft and Siemens AG among many others. The company is privately held with investments from Equitec Partners and Prime Technology Ventures. Codenomicon, whose main objective is to ensure the security and robustness of any application or service implementation, has been recognized by the industry for its innovations in systematic blackbox negative testing. Development and security personnel in lab or staged environments utilize Codenomicon's DEFENSICS platform to fortify quality and security assurance- quickly, easily and reliably. The test software features a systematic blackbox and negative test methodology uniquely capable of revealing undesired behavior and issues in protocol implementations.

Codenomicon teams its Protocol Modeling Engine and Attack Simulation Engine with protocol support that covers network, wireless, and digital media. Thousands of pre-built, highly targeted, and well-documented test cases allow users to see results as soon as the platform is connected to the target system – accelerating time-to-value. Codenomicon's test tools benefit all end-users in the software, networking, service provider and defense industry. Developers can cut development and maintenance costs by catching bugs early in the software development lifecycle, while operators can test and compare software from different vendors and fix any outstanding bugs. Enterprises and independent labs can test the robustness of software to provide insights for purchase decisions or risk analysis.

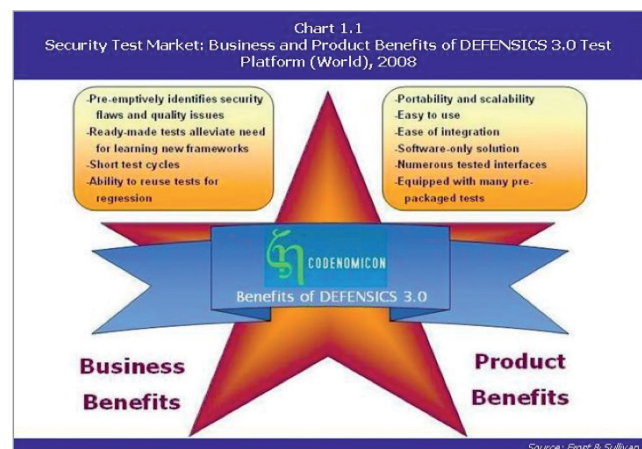
Among DEFENSICS 3.0's most important business-related benefits is its ability to enable vendors, carriers, and enterprises to identify and fix security flaws and quality issues pre-emptively. This process, which is performed via intelligent negative testing and maintained RFC coverage, ensures that such flaws are eliminated before they can be discovered by third parties. Also, the inclusion of automated, ready-made tests removes the need for end users to



learn new frameworks or to design tests from scratch. DEFENSICS 3.0 comes equipped with thousands of pre-defined, fully configurable test cases that are optimized to efficiently discover irregular responses, slower system reaction, or terminated processes or system crashes. By knowing only the test target protocol interfaces, DEFENSICS 3.0 users can readily start testing and experiencing immediate results. The most recent platform version also offers shorter test cycles than its predecessors. Furthermore, identified flaws are repeatable and traceable, and tests can be reused for regression purposes. Users have fully-integrated documentation, the exact test case construct, and input context to determine the main cause of any identified defects.

DEFENSICS 3.0, which is a highly portable and scalable solution, allows for easy integration to satisfy end user needs. The platform runs on various operating systems and nominal hardware, including laptops. This software-only solution provides engineering and security professionals the flexibility to immediately test any system or device in field or lab settings. The software supports remote users, multiple sites, multiple protocols, external audits, and third-party license management systems. By making the system accessible to different teams and users, organizations can increase usage and optimize resources while reducing expert staff utilization, as well as extra travel and preparation costs. Much like with DEFENSICS 2.0, the current platform version covers many different interfaces and formats, enabling the testing of systems from link-level communications all the way up to the application protocol. However, DEFENSICS 3.0 is able to cover a variety of new types of interfaces and usage scenarios, due to advances in testing technology.

The Frost & Sullivan Award for Product of the Year is presented each year to the company that has demonstrated excellence in new product development and launch within its industry. The recipient company has shown innovation by launching a broad line of emerging products and technologies.



For more information on the Frost & Sullivan Security Test Product of the Year 2008 award, see: <http://www.codenomicon.com/resources/whitepapers/2008-product-of-the-year.shtml>