Fuzzing이란 무엇인가?

(v.0.1)

securityproof.net

차례

1. Fuzzing이란 무엇인가?

최근 국내 해커들도 각종 어플리케이션에 존재하는 취약점들을 발견하는데 많은 노력을 기울이고 있다. 그 이유가 일시적인 유행이던 아니면 사업적인 목적을 위해서이든 많은 의미가 있다. 왜냐하면 여전히 우리나라의 보안계를 통해 주요 취약점들이 발표되는 일은 여전히 드물기 때문이다. 주요 보안 취약점을 발견하여 발표할 수 있다는 것은 한 나라의 보안 척도를 보여주는 것이다.

일반적으로 취약점을 발견하기 위해서는, 소스가 공개되어 있는 경우 그 어플리케이션의 소스 코드를 라인 별로 분석하거나, 소스가 공개되지 않은 경우 바이너리를 reversing enginnering 등을 통해 분석하는 경우가 보통이다. 그리고 이 두 방법은 특별한 일이 없는 한 지속적으로 사용될 것이다.

소스 코드 분석 및 reversing enginnering 이외에 많이 사용되고 있는 방법이 바로 fuzzing이다. 외국의 경우 수년 동안 많은 논의가 있었으며, 주요 컨퍼런스 등을 통해 새로운 fuzzing 기법들이 추가되었다. 또한 다양한 종류의 fuzzing 툴들이 개발되어 사용되고 있다. 그리고 이 fuzzing 툴들을 이용해 우리에게도 익숙한 주요 어플리케이션들의 취약점을 찾아내는 경우가 많이 있었다. 그러나 아직 국내에서는 논의조차 활발하지 않은 상태이다. 변변한 fuzzing 툴 하나 없는 상태이다. 이 글이 fuzzing에 대한 이해에 조금이라도 도움이 될 수 있길 바란다.

Fuzzing은 일종의 소프트웨어 보안 테스트 기법이다. 기본적인 아이디어는 프로그램실행 시 무작위 데이터를 attach하는 것이다. 만약 프로그램 실행이 제대로 되지 않는다면 프로그램에 수정할 결함이 있다는 것을 의미한다. 즉, 무작위 데이터를 어플리케이션에 입력하고, 그 결과 어플리케이션에 exception, crash, 또는 서버가 다운되는 등의 에러가 발생할 경우 보안 취약점이 존재할 가능성이 높다는 것이다. 'fuzz'라는 단어를 영영 사전에서 찾아보면 "mistake in a performance"[1]라고 나온다. 단어의 의미를 따져보면 어플리케이션 등에 존재하는 취약점들을 찾아내기 위한 테스트라고 볼 수 있다.

Fuzz testing(Fuzzing)은 Wisconsin-Madison 대학의 Barton Miller 교수와 대학원생들에 의해 1989년에 개발되었다[2]. 이것에는 3가지 특징이 있으며, 이에 대해 당사자들은 그들의 웹 사이트[3]에서 다음과 같이 설명하고 있다.

---

[1] *Oxfoard Quick Reference Dictionary* 참고
[2] http://en.wikipedia.org/wiki/Fuzz_testing
[3] http://www.cs.wisc.edu/~bart/fuzz/

1. 입력 값은 무작위적이다. 우리는 어떤 모델의 프로그램 행위, 어플리케이션 타입, 또는 시스템 기술을 사용하지 않는다. 이것은 때론 *black box* 테스트[4]라고 불린다. 기존의 명령 라인 연구(X-Window 연구[5] (1995), Windows NT 연구[6](2000), 그리고 Mac OS X 연구[7](2006))에서 무작위의 입력 값은 단순히 무작위의 ASCII 문자의 연속(stream)이었다. 기존의 이 세가지 연구를 위해 사용된 입력 값은 단지 유효한 키보드 및 마우스 이벤트를 가진 경우만 포함했다.
2. 우리의 신뢰 기준은 간단한데, 만약 어플리케이션의 실행이 멈추거나(crash) 실행이 일시적으로 보류(hang)되면 테스트에서 실패한 것으로 간주되고, 그렇지 않다면 테스트를 통과하는 것이다. 어플리케이션이 입력 값에 대해 적절하게 반응을 하지 않는다면 아무런 표시 없이 종료될 수도 있다는 것을 주목해라.
3. 첫 두 가지 특징의 결과로 fuzz 테스트는 더 높은 수준으로 자동화될 수 있고, 테스트 결과는 어플리케이션이나 운영체계, 그리고 벤더들과 각자 비교될 수 있다.

다양한 fuzzing 툴들이 지속적으로 등장하고 있지만 어플리케이션에 존재하는 특정 취약점을 구체적으로 완벽하게 지적해주는 것은 드물다. Fuzzing 테스트는 전체 과정에서 1차 관문에 해당된다고 생각할 수 있다. Fuzzing 테스트 동안 발생하는 특정 부분의 exception, crash 등을 파악하고, 이 부분을 디버깅 또는 분석하는 추가 작업이 필요하다. 물론 어떤 툴의 경우 취약점을 발견한 후 기본적인 exploit까지 만들어주는 것도 있다. 하지만 아직까지는 우리가 완벽하다고 느낄 수 있는 fuzzing 툴은 없으며, 또한 fuzzing 테스트를 통해 취약점 여부를 완벽하게 파악할 수도 없다.

2. Fuzzing 테스트 방법

개발자들은 프로그램에 존재하는 문제점을 사전에 찾아 해결하기 위해 일부러 에러를 유발시키는 테스트 과정을 거친다. 그리고 fuzzing 테스트 과정에서 나오는 데이터를 기록하여 문제 해결에 참고한다. 이런 fuzzing 테스트는 보통 소프트웨어를 공식적으로 발표하기 전에 실시하며, 테스트 데이터는 보존된다. 만약 fuzz stream이 pseudo-random number[8]이면 그 fuzzing 테스트를 다시 하기 위해 seed 값을 저장한다. Seed 값은 rand() 함수가 변화를 주는데 있어서 기준이 되는 초기값을 말한다.

---

[4] http://en.wikipedia.org/wiki/Black_box_testing
[5] ftp://ftp.cs.wisc.edu/paradyn/technical_papers/fuzz-revisited.pdf
[6] ftp://ftp.cs.wisc.edu/paradyn/technical_papers/fuzz-nt.pdf
[7] ftp://ftp.cs.wisc.edu/paradyn/technical_papers/Fuzz-MacOS.pdf
[8] http://en.wikipedia.org/wiki/Pseudo-random_number

Fuzzing 테스트를 위해 참고해야 할 현대 소프트웨어가 가지고 있는 다른 타입의 입력 값들은 다음과 같다.

* 그래픽 사용자 인터페이스 또는 임베디드 시스템의 메커니즘으로부터 나오는 이벤트 관련 입력 값
* 파일이나 소켓과 같은 데이터 스트림(data stream)으로부터 나온 문자 관련 입력 값
* 표 데이터로부터 나온 데이터베이스 입력 값
* 환경변수와 같은 고유한 프로그램 상태

fuzzing 테스트에는 다음과 같은 형태가 있다.

1. 무작위 입력 값이 합리적이거나, 또는 실제 제품 제작 데이터에 적합한지 확인하기 위한 fuzzing
2. 입력 값을 보통 허위 난수 생성기(pseudo random number generator)[9]를 사용하는 단순한 fuzzing
3. inject된 완전히 무작위의 입력 값의 비율에 맞게 유효한 테스트 데이터를 사용하는 fuzzing

3번은 1번과 2번의 기법이 혼합된 형태이다. 이 테크닉들을 모두 혼합하여 사용함으로써 fuzzing 테스트의 무작위성은 시스템 상태에 대해 광범위하게 점검하고 분석하는데 도움이 될 수 있다. 아래서 살펴볼 fuzzing 툴들을 보면 위의 세 가지 기법 중 특정 하나만을 적용하거나 2개 이상을 혼합하는 형태를 취하고 있다.

3. Fuzzing 테스트의 장단점

프로그램의 결함을 찾기 위해 사용하는 fuzzing 테스트의 가장 큰 약점이자 단점은 정확한 분석보다는 무작위성에 의존한다는 것이다. 따라서 보통의 경우 단순한 결함들은 쉽게 찾아내지만 아주 심각한 보안 취약점을 찾아내는 데는 그렇게 뛰어난 성능을 발휘하지는 못하고 있다.
원시적인 fuzzer(fuzzing tool)는 테스트 대상이 되는 소프트웨어의 전체 코드를 완벽하게 점검할 수 없을지 모른다. 예를 들어, 만약 입력 값이 다른 랜덤한 변화와 맞도록 적절하게 업데이트되지 않은 체크섬을 포함하고 있다면 이 fuzzing 툴은 내장된 체크섬 유효성 코드(checksum validation code)

[9] 난수(random number)란 규칙성이 배제된 숫자를 의미한다. 그러나 엄격한 의미에서 난수는 아니다. fuzzing에서는 특정 알고리즘에 의한 난수 생성기를 쓰기 때문에 'pseudo random number'라는 말을 사용할 뿐이다.

만으로 fuzzing 테스트를 진행할 것이다. 코드 전체를 얼마만큼 정확하게 확인할 수 있는가는 fuzzer의 성능을 평가하는 중요한 기준이 될 것이다.

반면에 fuzzing 테스트를 통해 심각한 보안 취약점으로 이어지는 버그들이 발견되기도 하는데, 이런 버그는 원격 공격자에 의해 공격을 받아 시스템을 장악 당하는 그런 취약점들이 될 수 있다. 이것은 fuzzing 테스트가 해커들에 더 널리 알려지면서 점점 사실이 되어가고 있다. 갈수록 뛰어난 fuzzing 기법이나 fuzzer들이 등장하고 있는 것이다. 다양한 fuzzing 테크닉들과 툴들이 소프트웨어의 취약점을 찾아내는데 사용되고 있다. 이것은 바이너리 또는 소스 분석, 그리고 fuzzing의 사촌뻘 되는 fault injection[10]에 대해 가지는 중요한 장점이다. Fault injection에 대한 좀더 자세한 설명은 다음 검색 결과에서 나오는 자료들을 참고하길 바란다.

http://www.google.co.kr/search?sourceid=navclient&hl=ko&ie=UTF-8&rls=GGLG,GGLG:2005-46,GGLG:ko&q=Fault+injection

4. Fuzzing의 분야와 툴 소개

fuzzing의 분야를 알아보기 위해서는 어떤 종류의 fuzzing 툴들이 있는지 확인해보면 된다. Fuzzing 툴들의 목록을 제공하고 있는 사이트[11]가 있는데, 대표적인 fuzzing 툴들이 다 있는 것 같다. 여기서 제공된 툴들은 기존에 발표된 어플리케이션에 존재하는 취약점을 발견하는데 사용된 툴이기도 하다.

이 글을 작성하면서 찾은 사이트[12]에서 fuzzing 툴들을 잘 분류를 하고 있었다. 여기서는 hacksafe에서 분류한 것을 거의 전적으로 이용할 것이다. 일부는 hacksafe에서 설명한 것과는 다르다. 가능한 소개되는 모든 툴들에 대해 테스트를 해볼 것이며, 각 툴의 사용법과 세부 설명은 다음 장에서 언급하겠다. 먼저 각 툴의 기능과 성격에 따라 분류를 하고, 간단한 설명을 추가한다. 먼저 fuzzer API와 Framework부터 소개한다.

Fuzzer API 및 Framework

SPIKE – SPIKE는 “Fuzzer Creation Kit”이며, 새롭고 알려지지 않은 네트워크 프로토콜들을 리버싱 엔지니어링 하는데 도움이 될 수 있는 프로토콜 API를 사용하기 쉽도록 한다. 웹 서버 NTLM

[10] http://en.wikipedia.org/wiki/Fault_injection
[11] http://www.infosecinstitute.com/blog/2005/12/fuzzers-ultimate-list.html
[12] http://www.hacksafe.com.au/blog/2006/08/21/fuzz-testing-tools-and-techniques/

Authentication brute forcer, 웹 어플리케이션과 DCE-RPC (MSRPC)를 분석하는 예제 코드를 포함하고 있다.

Scratch – Scratch는 간단한 패킷으로부터 아주 다양한 취약점들을 찾아낼 수 있는 protocol destroyer (”fuzzer”)이다. Scratch는 Python으로 만들어졌으며, 어떤 데이터로 무엇을 fuzzing할지 결정하기 위해 바이너리 파일을 분석한다. Scratch는 SSL과 SMB과 같은 바이너리 프로토콜을 fuzzing하기 위한 구조(framework)도 갖추고 있다. "ui" 클래스를 이용해 HTTP와 FTP와 같은 ascii 기반의 프로토콜을 분석할 수 있다.

LXAPI - Library Exploit API – 버그 테스트 그리고 로컬 및 원격 취약점에 대한 공격을 위해 고안된 툴로써, python으로 만들어졌으며, 아직 개발 중이라고 보는 것이 좋다.

PEACH - Peach Fuzzer Framework – Peach는 Python으로 쓰여진 fuzzing framework이다. Peach의 주요 목표는 개발시간 단축, 코드 재사용, 사용의 용이성과 효율성이다. Peach는 NET, COM/ActiveX, SQL, 공유 라이브러리와 DLL의 요소를 가진 어플리케이션을 fuzzing할 수 있다.

antiparser – antiparser는 fuzzing 테스트 및 fault injection API이다. Antiparser의 목적은 모델 네트워크 프로토콜 및 파일 포맷에 사용될 수 있는 API를 제공하는 것이다. 일단 모델이 만들어지면 antiparser는 소프트웨어 버그나 보안 취약점 공격에 사용될 수 있는 무작위의 데이터를 만드는 다양한 방법을 가지게 된다. Python 2.3 이상 버전이 필요하다.

Autodafe – Autodafe는 프로토콜이나 어플리케이션에 존재하는 바운드리 유효성, 즉 overflow 문제와 다른 문제들을 확인하는데 사용될 수 있는 fuzzing framework이다.

dfuz – dfuz는 공격자가 원하는 데이터와 함께 무작위의 데이터를 무작위의 크기로 보내는 것과 같은 많은 것을 할 수 있는 원격 프로토콜 fuzzer이다. WS-FTP overflow나 Microsoft RPC DoS 취약점과 같은 것을 찾아낼 수 있었다.

Web Application Fuzzing 툴

MielieTool - MielieTool v.1.0은 사용하기 편한 Perl 기반의 웹 어플리케이션 fuzzer이다. 이 툴은 form 및 링크 형태로 된 CGI를 fuzzing하는 것을 지원하며, 2개 이상의 사이트를 동시에 테스트할 수 있다. HTTrack, Lynx, grep, find, 및 rm이 필요하다.

Wapiti – Wapiti는 Python으로 구현된 웹 어플리케이션 fuzzer이다. 이 툴의 홈페이지에는 웹 취약점 스캐너로 나와 있다. 그런데 fuzzer로 분류한 것은 “black-box” 스캐닝을 하기 때문이다. 이 툴은

어플리케이션의 소스 코드를 분석하는 것이 아니라, 웹 서버에 설치되어 운영되고 있는 웹 어플리케이션의 웹 페이지를 스캐닝하고, 데이터를 인젝트할 수 있는 스크립트와 form을 찾는다. 이 목록을 가지게 되면 어떤 스크립트가 취약한지 확인하기 위해 payload를 인젝트하면서 fuzzer처럼 작동한다. Libtidy, ctypes, uTidylib가 설치되어 있어야 한다. Wapiti가 탐지할 수 있는 취약점들은 다음과 같다:

- File Handling Errors (로컬 및 원격 include/require, fopen, readfile...)
- Database Injection (PHP/JSP/ASP SQL Injection 및 XPath Injection)
- XSS (Cross Site Scripting) Injection
- LDAP Injection
- Command Execution detection (eval(), system(), passtru()...)
- CRLF Injection (HTTP Response Splitting, session fixation...)

WebFuzzer – WebFuzzer는 sql injection, cross site scripting, 원격 코드 실행, file disclosure, directory traversal, php including, shell escaping[13] 및 안전하지 않은 perl open() 호출과 같은 원격 취약점을 점검하는 웹 어플리케이션 fuzzer이다. 현재 Windows에서도 사용할 수 있도록 포팅되고 있는 상태이다.

SPI WebInspect – 이것은 SPI사의 상용 WebInspect toolkit이다. 여기에는 SPI Fuzzer도 포함되어 있는데, buffer overflow 취약점을 확인하기 위해 입력 변수 값을 수정하거나 fuzzing한다. 상용이라 테스트 및 그 수준을 확인할 수가 없었다.

cfuzzer – 이것은 클라이언트와 서버에 존재하는 HTTP chunked encoding 문제[14]를 테스트하기 위해 사용되는 fuzzer이다.

Browser Fuzzing 툴

MangleMe – 비정상적인 HTML 태그를 생성 및 이용하는 간단한 fuzzer이다. 브라우저를 자동으로 실행시키며, IE IFRAME 버그를 발견하는데 사용되었다.

---

[13] http://www.a-k-r.org/escape/rdoc/classes/Escape.html 참고

[14] http://search.securityfocus.com/swsearch?sbm=%2F&metaname=alldoc&query=HTTP+ chunked+ encoding+ &x=33&y=5

AxMan – AxMan은 H.D.Moore에 의해 개발된 웹 기반의 ActiveX fuzzing 엔진이다. AxMan의 목적은 Internet Explorer를 통해 노출된 COM 오브젝트에 존재하는 취약점을 발견하는 것이다. AxMan은 현재로서는 Internet Explorer 6에서만 사용된다. 이것의 세부 사용법은 securityproof.net 강좌란에 올려두었다.

COMRaider – COMRaider는 David Zimmer에 의해 COM Object 인터페이스를 fuzzing하기 위해 고안된 툴이다. 경로, 파일명, 또는 guid로 COM 오브젝트를 스캔할 수 있는 능력을 가지고 있다. 자세한 사용법은 다음 URL에 있다.

http://labs.idefense.com/files/labs/releases/previews/COMRaider/

취약점에 대해 스캐닝을 한 이후 취약점이 발견되면 완벽하지는 않지만 exploit까지 만들어 주는 기능까지 있다. 사용하기가 용이한 장점이 있다.

TagBruteForcer – TagBruteForcer는 Internet Explorer 내에서 디폴트로 오픈될 수 있는 어플리케이션들에 존재하는 overflow 취약점을 찾아내기 위해 고안된 툴이다. Overflow 취약점뿐만 아니라 ActiveX 오브젝트와 Internet Explorer 그 자체를 점검하는 기본적인 기능도 가지고 있다. 이 툴은 eEye Digital Security의 Drew Copley가 만든 것이다. 참고로 eEye Digital Security사의 사이트[15]에 유용한 툴들을 제공하고 있으니 참고하길 바란다.

Hamachi – Hamachi는 H.D.Moore와 Aviv Raff에 의해 쓰여진 것이다. 원리는 method argument와 속성 값에 대해 일반적인 '나쁜 값'을 지정함으로써 DHTM 구현상의 결함을 찾는 툴이다. 테스트 시간이 오래 걸린다는 단점이 있다.

Service 및 Protocol Fuzzing 툴

FTPFuzz – FTPFuzz는 FTPD 서버 구현상의 문제점을 테스트하기 위한 GUI 기반의 fuzzer이다. 사용자는 FTP 명령과 fuzzing할 파라미터들, 사용할 테스트 문자열의 패턴을 지정할 수 있다. 많은 유명 FTP 서비스에 존재하는 원격 공격이 가능한 취약점들이 이 툴을 이용해 발견되었다.

PROTOS - PROTOS 프로젝트는 black box 테스트 방법들을 이용하여 프로토콜 구현 상의 문제를 점검한다. 이 프로젝트의 홈페이지[16]에는 다른 좋은 자료들도 제공되므로 홈페이지를 참고하는 것이 좋겠다. 많은 PROTOS 테스트 관련 자료들이 제공되고 있으며, WAP, LDAP, SNMP, 그리고 DNS fuzzing도 제공된다.

---

[15] http://research.eeye.com/html/tools/
[16] http://www.ee.oulu.fi/research/ouspg/protos/

IRCfuzz - Digital Dwarf Society: 클라이언트용 fuzzing 툴

iCalFuzz - Digital Dwarf Society: iCal calendar 포맷용 fuzzing 툴

tftpFuzz - Digital Dwarf Society: tftp 프로토콜용 fuzzing 툴

dhcpFuzz - Digital Dwarf Society: dhcp 프로토콜용 fuzzing 툴

SMTPFuzzer - SMTP 프로토콜 구현을 하고 있는 서버의 취약점을 찾기 위한 fuzzing 툴

RIOT 및 Faultmon - plain text 프로토콜들(Telnet, HTTP, SMTP)을 공격하는데 사용될 수 있다. Riley Hassell이 eEye에서 일할 때 IIS .printer overflow 취약점[17]을 발견할 때 사용한 툴이며, *The Shellcoder's Handbook*이란 책에도 소개되어 있다.

TCP/IP Fuzzing 툴

Fuzzball2 - Fuzzball2는 nologin[18]의 warlord가 만든 Linux용 TCP/IP fuzzing 툴이다. 이 툴은 특정 호스트에 bogus 패킷을 보낸다.

ISIC – ISIC는 IP 스택과 그것의 구성 스택(TCP, UDP, ICMP 등)의 안정성을 테스트하기 위한 유틸리티들을 모아둔 것이다. 이것은 목표 프로토콜에 허위의 무작위 패킷을 생성한다. 그런 다음 타깃 머신의 방화벽 규칙을 공격하고, IP 스택에 존재하는 버그를 찾기 위해 이 패킷이 보내진다. ISIC는 하드웨어 구현상의 문제도 점검하기 위한 기능도 가지고 있다.

ip6sic - ip6sic는 IPv6 스택 구현을 주로 테스트하기 위한 툴이다. ISIC와 비슷하게 작동한다. 주로 FreeBSD에서 개발되었으며, OpenBSD와 Linux에서 작동하는 것으로 알려져 있다. 이론상으로는 libdnet이 작동하는 곳이면 작동할 수 있다.

다른 Fuzzing 테스트 툴

SyscallFuzz – Linux용 system call fuzzer

Socket Fuzzer – 유닉스용 socket/file descriptor fuzzing 툴

---

[17] http://www.securityfocus.com/bid/2674
[18] http://www.nologin.org/

Mangle - Ilja van Sprundel가 만든 바이너리 파일 fuzzer

FileFuzz - iDefense에서 만든 Windows PE 바이너리에 대한 파일 포맷 fuzzer

SPIKEFile – SPIKEfile는 SPIKE 2.9기반의 파일 포맷 fuzzing 툴이다. 리눅스를 기반으로 하고 있으며, 어플리케이션 실행과 fuzzing된 파일들에 의한 exception 탐지를 자동화하도록 고안되었다. 파일을 생성하기 위해 표준 SPIKE 스크립트를 사용하며, 흥미로운 시그널을 골라내고, 레지스터 상태를 dump하기 위해 ptrace를 활용한다.

FuzzyFiles - Reed Arvin가 만든 파일 fuzzer이다. 한 파일에 대한 다양한 변종을 만들고, 로컬 어플리케이션에 존재하는 결함을 찾는데 유용한다.

FuzzySniffandSend - Reed Arvin가 만든packet sniffer 및 replayer이다. 데이터를 캡처하여 다양한 방식으로 수정하고, 타깃에 다시 발송하는데 사용될 수 있다. 프로토콜과 어플리케이션 취약점들을 테스트하는데 사용된다.

radiusfuzzer - SuSe Security Team의 Thomas Biege가 C로 만든 Radius protocol[19] fuzzer이다.

msn_fuzzer – 간단한 MSN protocol fuzzer. MSN 클라이언트 소프트웨어에 존재하는 취약점들을 발견하기 위해 사용된다.

Mistress - 설명서가 독일어로 되어 있음

5. Fuzzing 툴 사용법

추가 예정(정리 중)

6. Fuzzing 툴 개발 방법론

추가 예정

[19] http://en.wikipedia.org/wiki/RADIUS

7. Reference

Oxfoard Quick Reference Dictionary

* http://en.wikipedia.org/wiki/Fuzz_testing

* http://www.cs.wisc.edu/~bart/fuzz/

* http://en.wikipedia.org/wiki/Black_box_testing

* ftp://ftp.cs.wisc.edu/paradyn/technical_papers/fuzz-revisited.pdf

* ftp://ftp.cs.wisc.edu/paradyn/technical_papers/fuzz-nt.pdf

* ftp://ftp.cs.wisc.edu/paradyn/technical_papers/Fuzz-MacOS.pdf

* http://en.wikipedia.org/wiki/Pseudo-random_number

* http://en.wikipedia.org/wiki/Fault_injection

* http://www.infosecinstitute.com/blog/2005/12/fuzzers-ultimate-list.html

* http://www.hacksafe.com.au/blog/2006/08/21/fuzz-testing-tools-and-techniques/

* http://www.a-k-r.org/escape/rdoc/classes/Escape.html

*http://search.securityfocus.com/swsearch?sbm=%2F&metaname=alldoc&query=HTTP+chunked+encoding+&x=33&y=5

* http://research.eeye.com/html/tools/

* http://www.ee.oulu.fi/research/ouspg/protos/

* http://www.securityfocus.com/bid/2674

* http://www.nologin.org/

* http://en.wikipedia.org/wiki/RADIUS