Synergy! A world where the tools communicate

Joshua "Jabra" Abraham
Rapid7 LLC
jabra@spl0it.org
jabra@rapid7.com

OWASP
Fall 2009

The OWASP Foundation
http://www.owasp.org

Purpose of this talk

- Raising the bar on pentesting
  - Build upon current tools
  - Leverage XML to automate pentesting tasks
  - Extract data for a correlation engine
- What we are doing today
  - High-level overview of an improved process (COE)
  - Releasing several modules

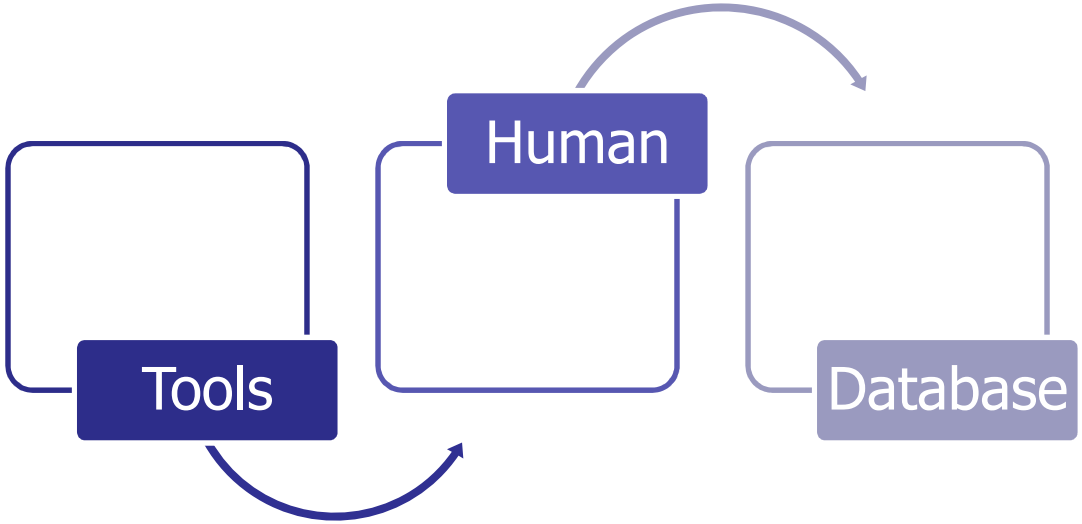
Encourage developers to build tools with XML and APIs

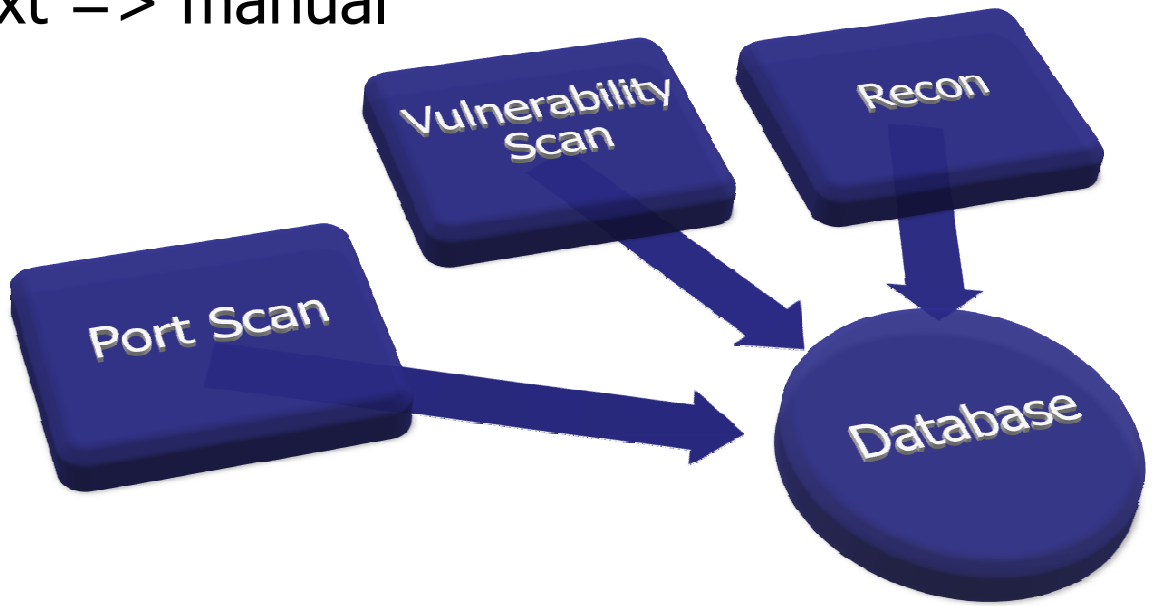
Agenda (Intense 25 minutes)

- Programming focused talk
- A boat load of XML and parsers
- Automating the "stupid" stuff…
  - Several new modules

Flow is Key

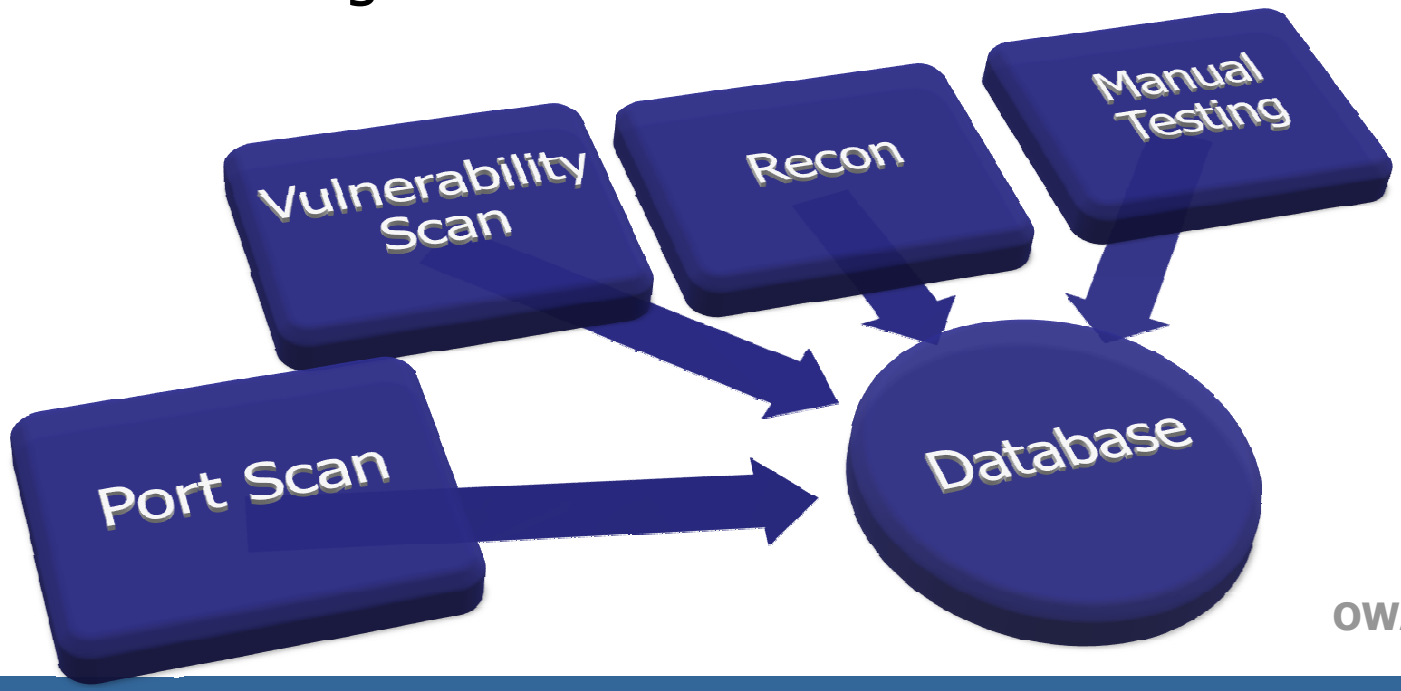
- UNIX tools
  - Program
  - Shell script
- Data processing
  - txt => manual





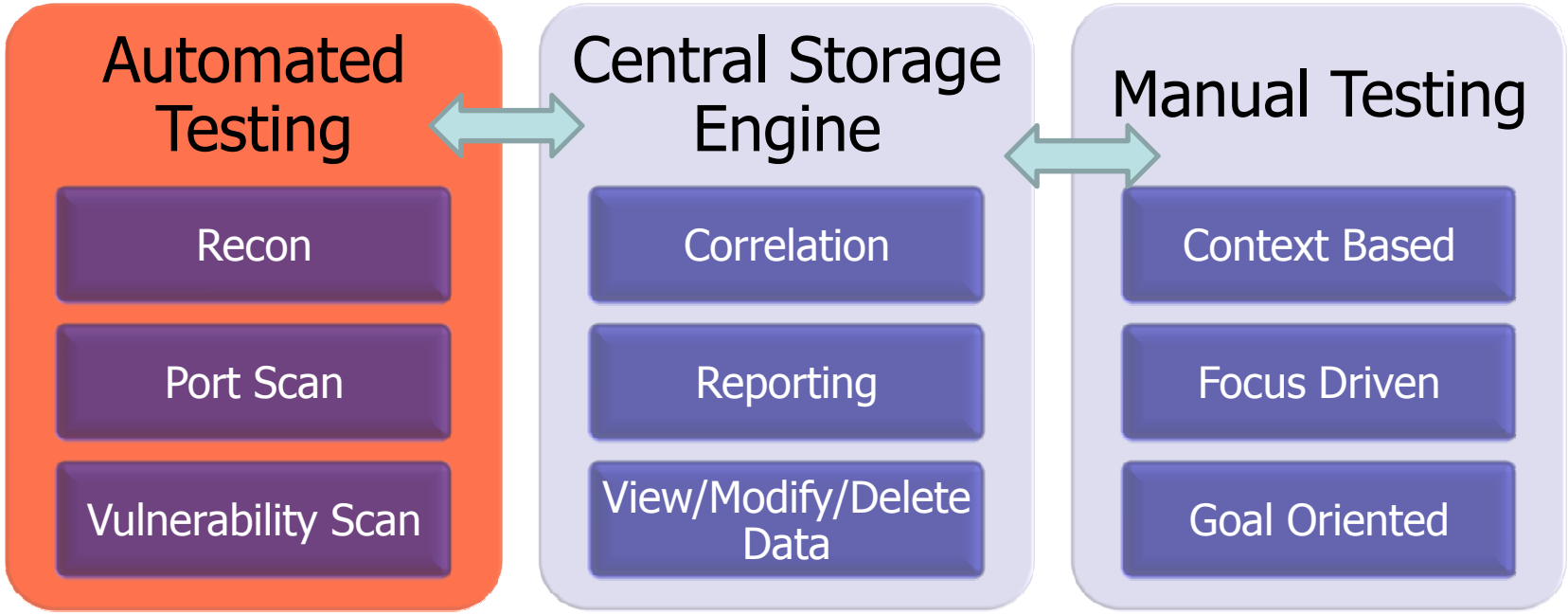
Add the Manual Aspect

- Computers are good at doing specific tasks
  - Identifying Open Ports, Finding XSS and Bruteforcing passwords
- Humans are good at doing non-specific task
  - Reasoning based on context

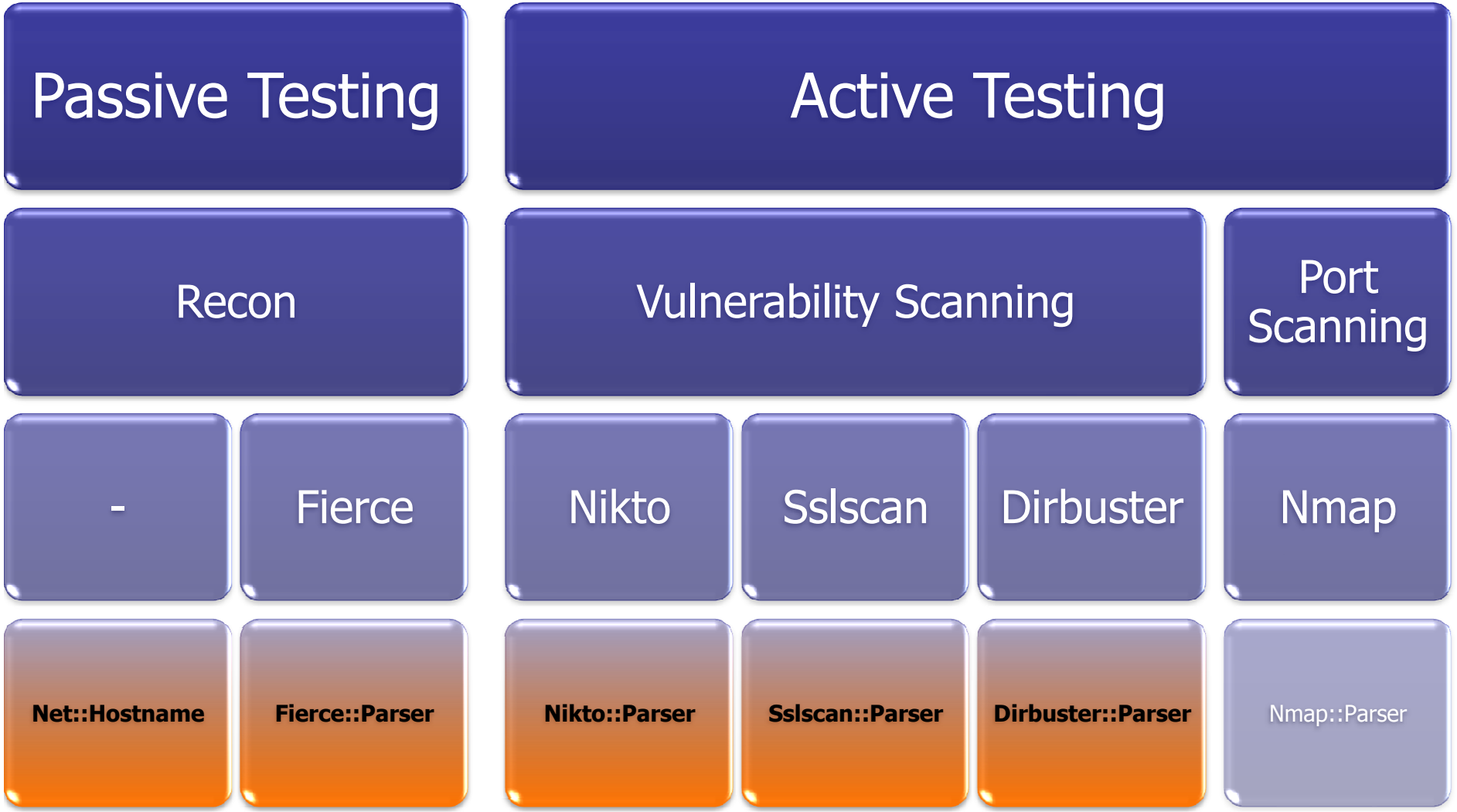


Level the playing field

- All components are equal. However, some components are more equal than others.
- We will focus on automated testing



Techniques


Passive Testing
Active Testing
Recon
Vulnerability Scanning
Port Scanning
-
Fierce
Nikto
Sslscan
Dirbuster
Nmap
Net::Hostname
Fierce::Parser
Nikto::Parser
Sslscan::Parser
Dirbuster::Parser
Nmap::Parser

Programming Language

Sounds like Earl, but starts with a "P"

- The programming language is Perl
  - The following are NOT programming languages:
    - PERL, perl, Pearl
- Cross Platform
- Built for Scripting and Object Orientation
- Libraries = modules
  - Load a module: use My::Module;
- Docs
  - perldoc perl
  - perldoc My::Module

Setup Phase

- The rest of the talk will be all code!
- Loading the following modules:

```
use Nikto::Parser;
use Dirbuster::Parser;
use Sslscan::Parser;
use Fierce::Parser;
use Net::Hostname;
```

Setup Phase

- Creating parser objects:

```
my $np = new Nikto::Parser;
my $dp = new Dirbuster::Parser;
my $sp = new Sslscan::Parser;
my $fp = new Fierce::Parser;
```

Net::Hostname

- Resolves Hostnames for IPv4 and IPv6

```
my $h = Net::Hostname->new(hostname => "www.google.com");

 print $h->resolveIPv4 . "\n";
#64.233.169.104
 print $h->resolveIPv6 . "\n";
#2001:4860:b002::68
```

Fierce (Network Reconnaissance tool)

- Built to find IPs owned by your target
  - Version 1.0 built by Rsnake
  - Version 2.0 re-written by Jabra
- Techniques
  - Enumerate DNS servers and check for Zone Transfer
  - Enumerate prefixes, extensions and subdomains
  - Virtual Host detection
  - Check for MX records and Wildcards
  - Reverse Lookups based on Hostnames
  - Range enumeration based on subnet
  - ARIN, ARPNIC, etc enumeration….

Fierce (Network Reconnaissance tool)

```
                <node ip="121.101.152.99" hostname="ns3.yahoo.com"/>
                <node ip="68.142.196.63" hostname="ns4.yahoo.com"/>
                <node ip="119.160.247.124" hostname="ns5.yahoo.com"/>
                <node ip="202.43.223.170" hostname="ns6.yahoo.com"/>
                <node ip="68.142.226.82" hostname="ns7.yahoo.com"/>
                <node ip="202.165.104.22" hostname="ns8.yahoo.com"/>
                <node ip="202.160.176.146" hostname="ns9.yahoo.com"/>
                <node ip="69.147.114.29" hostname="ns13.yahoo.com"/>
                <node ip="216.252.122.25" hostname="ns14.yahoo.com"/>
    </bruteforce>
    <!--  Subdomain Brute Force DNS -->
    <subdomainbruteforce starttime="1242762092" starttimestr="Tue May 19 15:41:3
2 2009" endtime="1242762099" endtimestr="Tue May 19 15:41:39 2009" elasptime="7"
>
                <node ip="206.190.60.37" hostname="www.mail.yahoo.com"/>
                <node ip="206.190.60.37" hostname="www.mail2.yahoo.com"/>
                <node ip="206.190.60.37" hostname="www.mail3.yahoo.com"/>
                <node ip="206.190.60.37" hostname="www.mail4.yahoo.com"/>
                <node ip="206.190.60.37" hostname="www.mail5.yahoo.com"/>
                <node ip="206.190.60.37" hostname="www.mail6.yahoo.com"/>
                <node ip="206.190.60.37" hostname="www.mail7.yahoo.com"/>
                <node ip="206.190.60.37" hostname="www.mail8.yahoo.com"/>
                <node ip="206.190.60.37" hostname="www.mail9.yahoo.com"/>
                <node ip="206.190.60.37" hostname="www.mail10.yahoo.com"/>
```

Fierce::Parser

- Fierce has many output formats
  - TXT, HTML and XML
- Parse Data from Fierce XML

Fierce::Parser

```
my $parser = $np->parse_file('google.xml');
my $node = $np->get_node('google.com');
my $bf = $node->bruteforce;

print "Prefix Bruteforce:\n";
foreach my $n ( $bf->nodes ) {
    print "Hostname:\t" . $n->hostname . "\n";
    print "IP:\t\t" . $n->ip . "\n";
}
```

Fierce::Parser

```
Prefix Bruteforce:
        Hostname:       www.yahoo.com
        IP:             69.147.76.15
        Hostname:       mail.yahoo.com
        IP:             69.147.112.160
        Hostname:       www2.yahoo.com
        IP:             206.190.60.37
        Hostname:       www3.yahoo.com
        IP:             206.190.60.37
        Hostname:       dns.yahoo.com
        IP:             68.142.196.63
        Hostname:       www4.yahoo.com
        IP:             206.190.60.37
        Hostname:       www5.yahoo.com
        IP:             206.190.60.37
        Hostname:       www6.yahoo.com
        IP:             206.190.60.37
        Hostname:       www7.yahoo.com
        IP:             206.190.60.37
        Hostname:       ns.yahoo.com
        IP:             66.218.71.63
        Hostname:       mail2.yahoo.com
        IP:             209.191.90.107
        Hostname:       mail3.yahoo.com
:_
```

Dirbuster

- Web Application Traversing
- Identifying locations that do not require authorization
- Runs on Linux, Windows and BSD
- OWASP project!
- New version has XML Output!

Dirbuster::Parser

```
my $parser = $dp->parse_file('dirbuster.xml');
my @results = $parser->get_all_results();

print "Directories:\n";
foreach(@results) {
    print "Path" . $_->path . "\n";
    print "Type" . $_->type . "\n";
    print "Response " . $_->response_code . "\n";
}
```

Dirbuster::Parser

```
Directories:
Type: Dir
Path: /
Response Code: 200
Type: Dir
Path: /cgi-bin/
Response Code: 403
Type: Dir
Path: /icons/
Response Code: 200
Type: Dir
Path: /doc/
Response Code: 403
Type: Dir
Path: /icons/small/
Response Code: 200
Type: Dir
Path: /base/
Response Code: 302
Type: Dir
Path: /base/index/
Response Code: 302
Type: File
Path: /base/index.php
:_
```

Encourage developers to build tools with XML and APIs

Sslscan

- SSL Cipher testing
- Similar to SSLDigger
- Sslscan runs on Linux, Windows and BSD
- XML Output
- Supports both HTTPS and SMTP

Sslscan::Parser

```
my $parser = $sp->parse_file('domain.xml');
my $host = $parser->get_host('domain.com');
my $port = $host->get_port('443');

foreach my $i ( grep($_->status =~ /accepted/,
 @{ $port->ciphers })  ) {
   print "sslversion " . $i->sslversion . "\n";
   print "cipher " . $i->cipher . "\n";
   print "bits " . $i->bits . "\n";
}
```

Sslscan::Parser

```
ip is: google.com
port: 443
accpeted ciphers are
sslversion is SSLv2
ciphers is DES-CBC3-MD5
bits is 168
sslversion is SSLv2
ciphers is DES-CBC-MD5
bits is 56
sslversion is SSLv2
ciphers is EXP-RC2-CBC-MD5
bits is 40
sslversion is SSLv2
ciphers is RC2-CBC-MD5
bits is 128
sslversion is SSLv2
ciphers is EXP-RC4-MD5
bits is 40
sslversion is SSLv2
ciphers is RC4-MD5
bits is 128
sslversion is SSLv3
ciphers is AES256-SHA
bits is 256
:_
```

Nikto::Parser

- Options for usage:
  - Scan and save XML for parsing later.
  - Scan and parse XML inline

Nikto::Parser

```
my $parser = $np->parse_file('nikto.xml');
my $h = $parser->get_host('127.0.0.1');
my $p = $h->get_port('80');

print "Target is: " . $h->ip . ":" . $p->port .  "\n";
print "Banner is: " . $p->banner . "\n\n";
foreach my $v ( @{ $p->get_all_items(); } ) {
    print $v->description . "\n\n";
}
```

Nikto::Parser

```
Target is: 127.0.0.1:80
Banner is: Apache/2.2.9 (Ubuntu) PHP/5.2.6-bt0 with Suhosin-Patch

Non-standard header keep-alive returned by server, with contents: timeout=15, ma
x=100

Apache/2.2.9 appears to be outdated (current is at least Apache/2.2.11). Apache
1.3.41 and 2.0.63 are also current.

Number of sections in the version string differ from those in the database, the
server reports: 5.2.6.45.98.116.0 while the database has: 5.2.6. This may cause
false positives.

Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE

HTTP method ('Allow' Header): 'TRACE' is typically only used for debugging and s
hould be disabled. This message does not mean the server is vulnerable to XST.

HTTP TRACE method is active, suggesting the host is vulnerable to XST

/doc/: The /doc/ directory is browsable. This may be /usr/doc.

/server-status: This reveals Apache information. Comment out appropriate line in
 httpd.conf or restrict access to allowed hosts.
:_
```

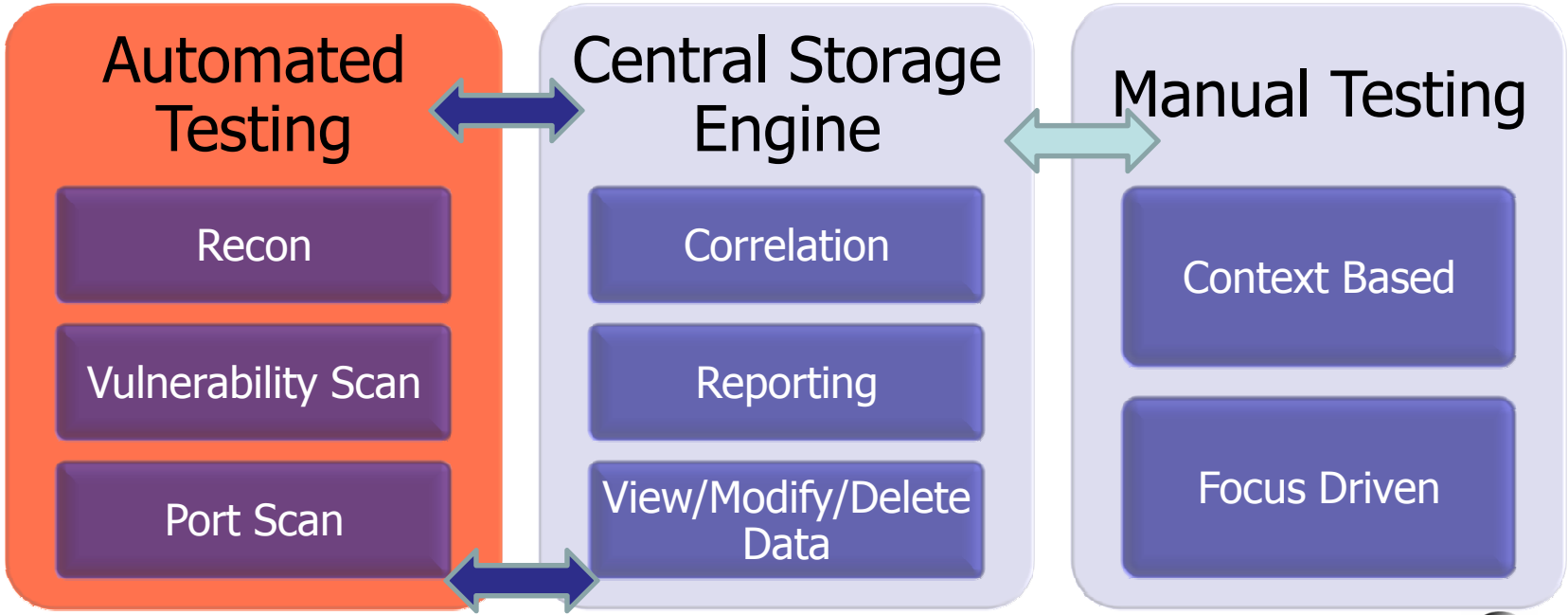
Results

- Nikto::Parser – parse Nikto data
- Sslscan::Parser – parse Sslscan data
- Fierce::Parser – parse Fierce data
- Dirbuster::Parser – parse Dirbuster data
- Net::Hostname – resolve hostnames

- All code will be available at:
- http://spl0it.org

Summary

- Extracting Data for the Central Storage Engine…
- Many tools, we have the choice how
  - Shell scripts, XML Parsers or manually



Encourage developers to build tools with XML and APIs

Contact Information

- Joshua "Jabra" Abraham
  - jabra@spl0it.org
  - jabra@rapid7.com

  - http://spl0it.wordpress.com
  - http://spl0it.org/files/talks/appsec09
  - (Final version of the slides, demos and code)