OSSAMS - Security Testing Automation and Reporting

Adrien de Beaupré
Intru-Shun.ca Inc.
SANS Internet Storm Center Handler

SecTor 2011, 19 October 2011

Agenda

- Definitions
- Methodology
- Workflow
- Reporting
- Problems
- Solutions
- Conclusion

Definitions

- Vulnerability - flaw or weakness in a system that can be exploited.
- Security audit - assess the adequacy of controls and evaluate compliance.
- Vulnerability assessment - description and analysis of vulnerabilities in a system.
- Penetration testing - circumvent the security features of a system.

Testing

- Every test consists of a stimulus and response, and monitoring to verify the response, or lack thereof.
- Testing consists of modules.
- Each module has an input and an output.
- You must monitor closely for responses.
- Testing must be appropriate to the target.
- Testing is of limited value if nothing is fixed.

Methodology

- Logistics and Planning
- Open Source Information Gathering
- Reconnaissance
- Identification / Enumeration
- Research
- Vulnerability Identification
- Validation / Exploitation
- Reporting

Open Source Info

- Purpose: gathering information on the target organization, typically from the Internet.
- Inputs: organization name, URL, IP addresses or ranges, industry or organization type.
- Outputs: URLs, IP addresses or ranges, email addresses, ‘buzz’, technologies used, resumes, names, host names…
- Data types: text, graphics, statistics…

Reconnaissance

- Purpose: determine which systems are live and map the network/technology.
- Inputs: URLs, IP addresses or ranges.
- Outputs: Whois, DNS, IP addresses or host names of systems which are likely to be live…
- Tools: Ping, Nmap, Ike-scan, Fierce Doman Scanner, traceroute, ICMP…
- Data types: text files, XML files…

Identification / Enumeration

- Purpose: enumerate the systems that are live, determine open ports, listening services, map applications, operating systems, and versions.
- Inputs: systems known to be live/available.
- Outputs: ports, services, OS, versions, patches.
- Tools: Nmap, Amap, Ike-scan, Nessus…
- Data types: text files, XML files…

Research

- Purpose: list all potential vulnerabilities.
- Inputs: technologies in use.
- Outputs: list of potential vulnerabilities.
- Tools: vulnerability databases, search engines…
- Data types: text files, XML files, databases…

Vulnerability Identification

- Purpose: identify known or unknown vulnerabilities in the identified technologies.
- Inputs: IP addresses, ports, services, applications.
- Outputs: listing of potential vulnerabilities.
- Tools: scanners such as Nessus, NexPose, Burp, W3AF, ZAP…
- Data types: text files, XML files, databases…

Validation / Exploitation

- Purpose: assign a confidence value and validate potential vulnerabilities. Have FUN!!
- Inputs: listing of all potential vulnerabilities.
- Outputs: listing of validated vulnerabilities and confidence rating values.
- Tools: penetration testing (Metasploit, Core Impact, Canvas…), manual validation, fuzzers…
- Outputs: text files, graphics, XML files, database entries, databases...

Reporting

- Purpose: assign risk and priority ratings to confirmed vulnerabilities.
- Inputs: list of validated vulnerabilities.
- Outputs: analysis results.
- Tools: people brain power.
- Outputs: text files, database entries, documents...
- Wordsmithing.

Why Automate?

- Laziness ☺.
- Consistent results over time.
- Allows for scheduling and trending.
- Streamlined and more efficient.
- Engineering a process that can be run and maintained by an operational group.
- Allows the test team to concentrate on the areas that are not automated.

Requirements

- Process – follow consistent repeatable methodology.
- Scriptable – typically Linux CLI tools.
- Tool – result that can be parsed.
- Database – for correlation and reporting.
- Correlated – multiple sources of data.
- Analyzed – intelligent human analysis.
- Mitigation – how to respond, recommendations.
- Metrics – quantitative, measurable, trends.
- Severity – rating system.

Workflow

- Methodology is broken down into modules.
- Output from one is the input to the next.
- Unfortunately most tools do not follow the methodology flow precisely, or may not allow for data extraction between modules.
- Which means that either we must run each tool multiple times with different configurations, or different tools for each module.

Workflow

- Output from module > database import
- Database queries > inputs to next module
- Reporting module > ticketing
- Tickets > vulnerability management and mitigation
- Close the loop back to the test team process
- Re-test where necessary

Problem

- Individual tools do not always follow a methodology and do not always allow for sufficiently granular control.
- No one tool can perform all modules.
- Methodology requires use of multiple tools.
- Each tool may have a different output format or use a proprietary database.
- Correlation and analysis can be time consuming.

What is Missing

- Security Assessments collect a lot of data, but don’t always correlate the data.
- To properly identify risk and threats, correlation of collected data is necessary.
- Correlation between different tools is essential!
- Marking false positives, adding manual findings, and annotating is also required.
- Current systems – Extremely Expensive.

Solutions

- Single unified and normalized database schema for all security assessment tools.
- Obviously requires that such a schema exist!
- Requires a parser for each tool we use.
- This allows us to create an abstract layer between the tools and the common database, while still allowing us to enforce the methodology regardless of the tools used.

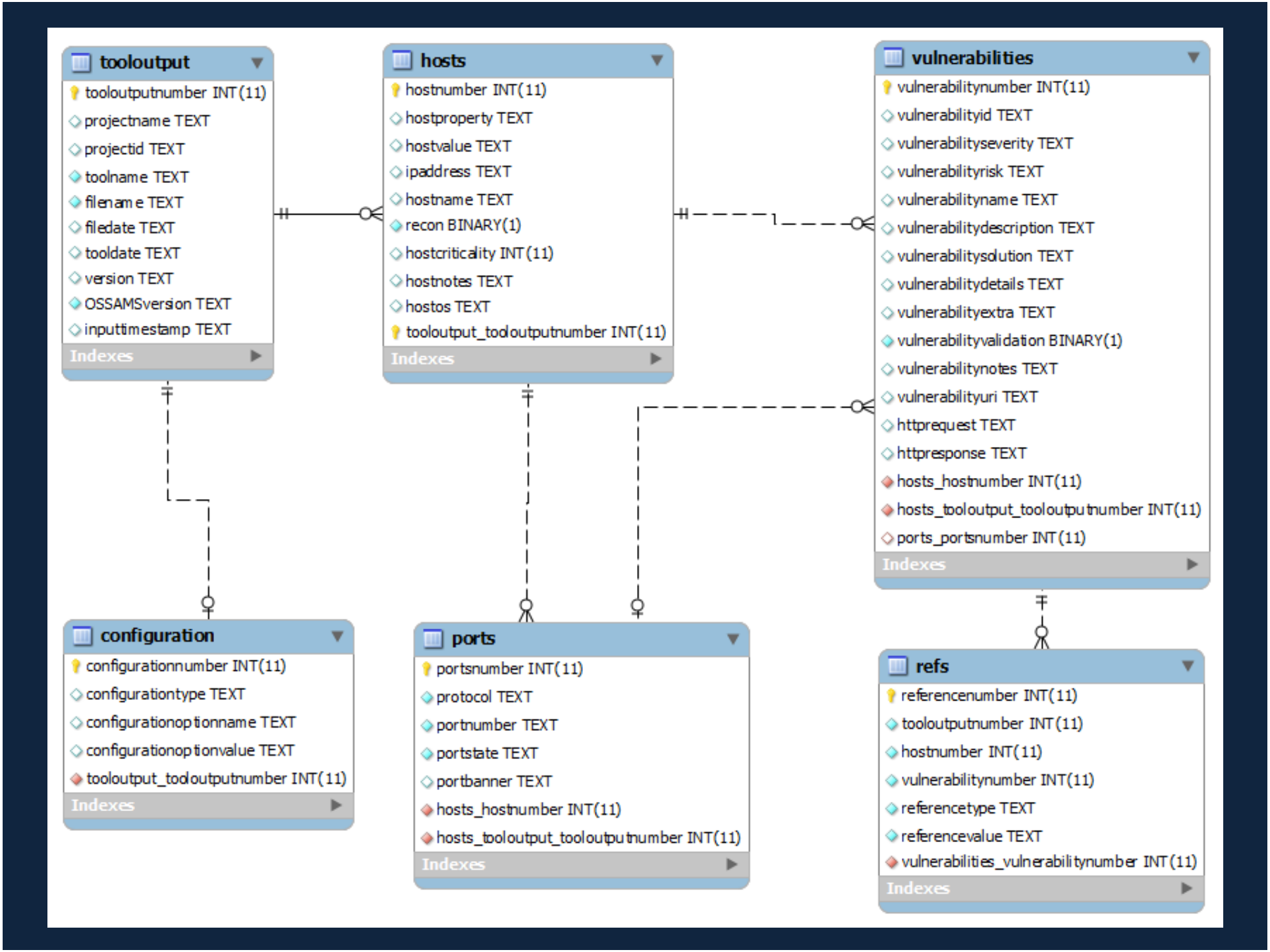
OSSAMS

- Open Source Security Assessment Management System

  www.ossams.com
- A framework for security assessors to correlate and analyze risk to information systems.
- Streamlines the assessment reporting process.
- A modular process that builds on past assessments.

Database Design

- One of the key aspects of OSSAMS is the database design.
- It is capable of having any number of tool outputs as an input.
- Currently using MySQL on Linux with Python, PowerShell, or Perl scripts to parse outputs.
- A front-end will be designed in addition to CLI.
- It is flexible, extensible, and Open Source.


tooloutput
tooloutputnumber INT(11)
projectname TEXT
projectid TEXT
toolname TEXT
filename TEXT
filedate TEXT
tooldate TEXT
version TEXT
OSSAMSversion TEXT
inputtimestamp TEXT
Indexes
hosts
hostnumber INT(11)
hostproperty TEXT
hostvalue TEXT
ipaddress TEXT
hostname TEXT
recon BINARY(1)
hostcriticality INT(11)
hostnotes TEXT
hostos TEXT
tooloutput_tooloutputnumber INT(11)
Indexes
vulnerabilities
vulnerabilitynumber INT(11)
vulnerabilityid TEXT
vulnerabilityseverity TEXT
vulnerabilityrisk TEXT
vulnerabilityname TEXT
vulnerabilitydescription TEXT
vulnerabilitysolution TEXT
vulnerabilitydetails TEXT
vulnerabilityextra TEXT
vulnerabilityvalidation BINARY(1)
vulnerabilitynotes TEXT
vulnerabilityuri TEXT
httprequest TEXT
httpresponse TEXT
hosts_hostnumber INT(11)
hosts_tooloutput_tooloutputnumber INT(11)
ports_portsnumber INT(11)
Indexes
configuration
configurationnumber INT(11)
configurationtype TEXT
configurationoptionname TEXT
configurationoptionvalue TEXT
tooloutput_tooloutputnumber INT(11)
Indexes
ports
portsnumber INT(11)
protocol TEXT
portnumber TEXT
portstate TEXT
portbanner TEXT
hosts_hostnumber INT(11)
hosts_tooloutput_tooloutputnumber INT(11)
Indexes
refs
referencenumber INT(11)
tooloutputnumber INT(11)
hostnumber INT(11)
vulnerabilitynumber INT(11)
referencetype TEXT
referencevalue TEXT
vulnerabilities_vulnerabilitynumber INT(11)
Indexes

Parsing Scripts

- Main function
- Read configuration function
- Database access function
- Read a list of files
- Read a directory of files
- Parsing XML, HML, or text file function
- Insert function
- Return

Tooloutput

- For every tool there are outputs. An output file, typically an XML file, will describe what the tool has discovered from the target domain, subnet, system, host, or application).
- Tooloutputnumber - Primary Key, auto-increment. Projectname, Projectid, Toolname, Filename, Filedate, Tooldate , Version, OSSAMSversion, Scanner , Inputtimestamp.

Configuration

- For every TOOLOUTPUT it may contain configuration information about the tool. Its primary key is configurationnumber, which is an auto-increment.
- Configurationtype, Configurationoptionname, Configurationoptionvalue

Hosts

- A toolout may describe none, one, or more hosts (computers or network devices). Its primary key is hostnumber, which is an auto-increment.
- Hostproperty, Hostvalue, ipv4, ipv6, Hostname, Hostptr, Whois, Recon, Reconreason, Hostcriticality, Macaddress, Macvendor, Hostnotes, Hostos, Osgen, Osfamily.

Ports

- A host may have none, one, or more ports open. This table contains information about ports (open, filtered, or closed). Its primary key is portsnumber, which is an auto-increment.
- Protocol, Portnumber, Portstate, Reason, Portbanner, Portversion, Portname, Service, Method, Confidence, Portvalue.

Vulnerabilities

- A host, port, or application may have none, one, or more vulnerabilities associated with it. Its primary key is vulnerabilitynumber, which is an auto-increment.
- Vulnerabilityid, Vulnerabilityseverity, Vulnerabilityrisk, Vulnerabilityconf, Falsepositive, Vulnerabilityname, Vulnerabilitydescription, Vulnerabilitysolution, Vulnerabilitydetails, Vulnerabilityextra, Vulnerabilityvalidation, Vulnerabilitynotes, Vulnerabilityattribute, Vulnerabilityvalue, Vulnerabilityuri, Httprequest, Httpresponse, Httpparam.

Refs

- A vulnerability may have none, one, or more references associated with it. A reference can be a link to a web site, a database entry (such as SecurityFous bid, OSVDB, Secunia, CVE, CCE, CWE, …).
- Referencetype - Type reference (URI, OSVDB, CVE…)
- Referencevalue – Value of the reference.

Other Objects

- Particularly for internal and/or authenticated scanning.
- Subnets: a host may be part of a subnet.
- Domains: a host may be part of a domain.
- Groups: a host or domain may group objects.
- Users: a host or domain may contain users.

Supported tools

- Completed:
  - acunetix, burp, grendel, nessus, netsparker, nexpose community, nikto, nmap, ratproxy, retina community, skipfish, sslscan, w3af, wapiti, watcher, websecurify, zap.
- Roadmap:
  - arachni, core impact, fierce, httprint, iss, languard, metasploit, ncircle, nexpose, n-stalker, ntospider, openvas, proxystrike, retina, saint, sandcat, webcruiser, webinspect, wsfuzzer…

Demo

- A brief demo of the parsing script and database use.
- Also briefly discuss the roadmap for OSSAMS:
  - Finalize the database design and parser scripts.
  - Reporting templates.
  - Query database for module tool input.
  - OSSTMM RAVs.
  - OWASP.
  - Other methodologies/frameworks.
  - Work on tool data interchange format.
  - Get more people involved!!

Conclusions

- The key is not running the scanners, but analysis, methodology, correlation, documentation, and problem solving.
- Organizations can automate security testing and reporting processes, particularly consultants and enterprises.
- The key is analysis and database utilization.
- These can be built using Free / Open Source Software tools and/or commercial offerings.
- Should be done with proper planning, tools, methodology, processes, and expertise.

QUESTIONS?

ADRIEN@INTRU-SHUN.CA

THANK YOU!