the Vulnerability Life Cycle

for software testers

Presenter

- Ben S. Knowles “adric” adric@adric.net
- BBST, GCIH, GSEC, LPIC-1
- Security Analyst, System Administrator with Dell SecureWorks in Atlanta, GA
- Volunteer work has included OLPC Laptop.org, AST EduSIG
- Past work: training, integration, development
- http://adric.net

Vulnerability lifecycle?

Critical to modern security profession

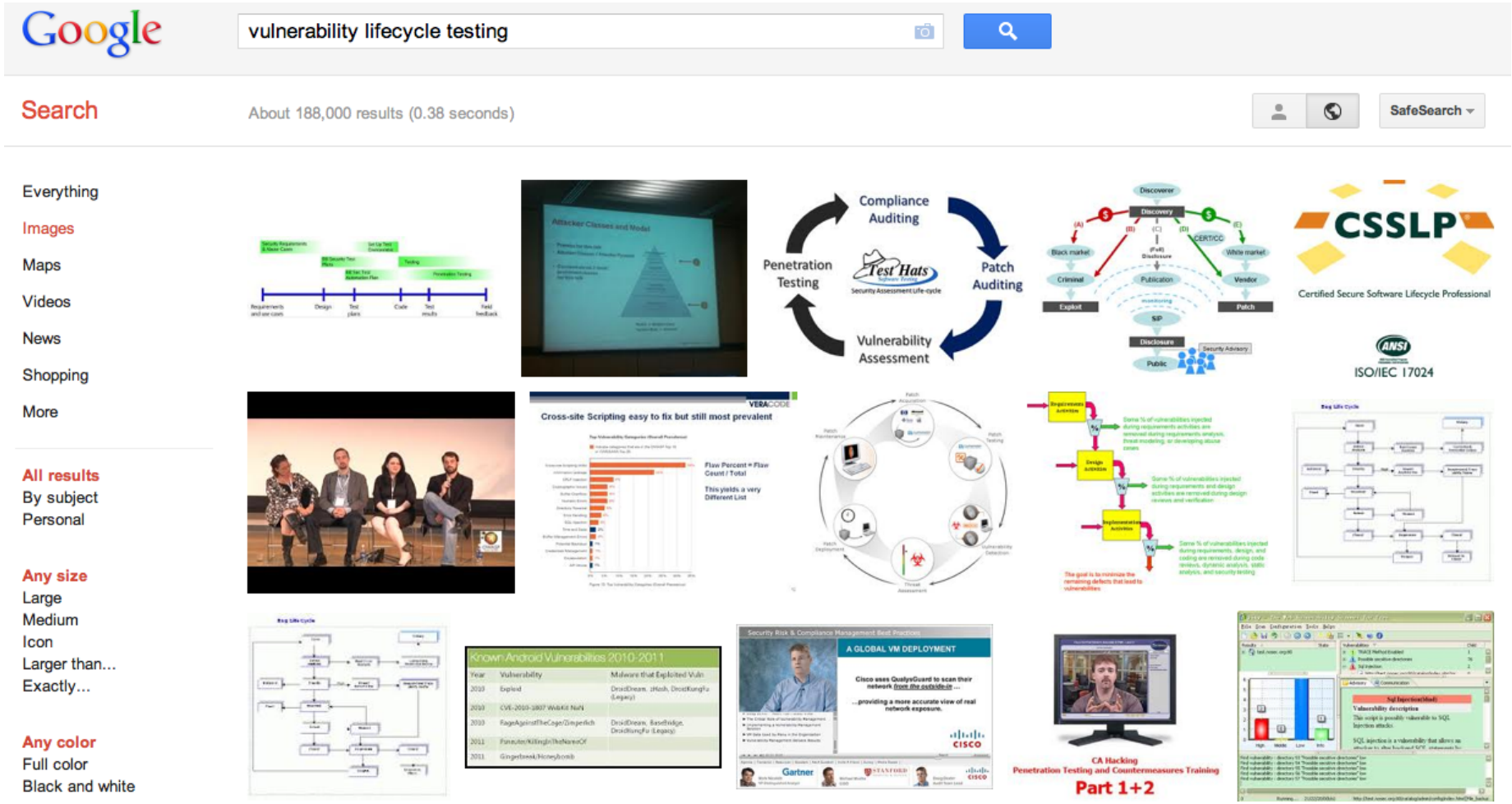
Describes how information about vulnerabilities

- Is discovered
- Circulates
- And to what purposes the information is put

Plenty of information is available including many lifecycle diagrams…

Vulnerability lifecycle, Google Image Search, last Tuesday

Where's the testing in any of these processes? And who is that lady?



Vulnerability lifecycle testing, Google Image Search, last Tuesday

Well, there's the word some, amidst a lot of products? And who are these people?

is Made of Software Testing!

Even after the product is released into the world:

- New Vuln research and development
- Vulns, tools, signatures all get QA
- Pen **Test**ing
- Vulnerability Assessment
- Verification and compliance scans
- even Attacking a system…

All use common software testing techniques!

Let’s get the jargon out of the way

COMMON TERMINOLOGY

Vulnerability

Vuln(erability): a weakness or flaw in a system that makes attack easier

- Bugs, but not just bugs in the software …
- Typically specific to a platform e.g.:

MS12-004 : for Win Media Player, on Windows

- But not always:

HashDOS vuln was in most every web platform: PHP, Tomcat, ASP, Perl, Ruby … all now patchable

And some flaws just keep getting re-implemented, like **Little Bobby Tables**

WMP Remote : MS12-004

Vulnerabilities in Windows Media Could Allow Remote Code Execution from MS12-004

- Make WMP do things it's not supposed to
- Chain with another vuln/sploit to increase privs
- Or just use WMP to attack another system (pivot)

Affects every modern Windows version

Patches were out Tuesday 10 Jan 2012 as Critical.

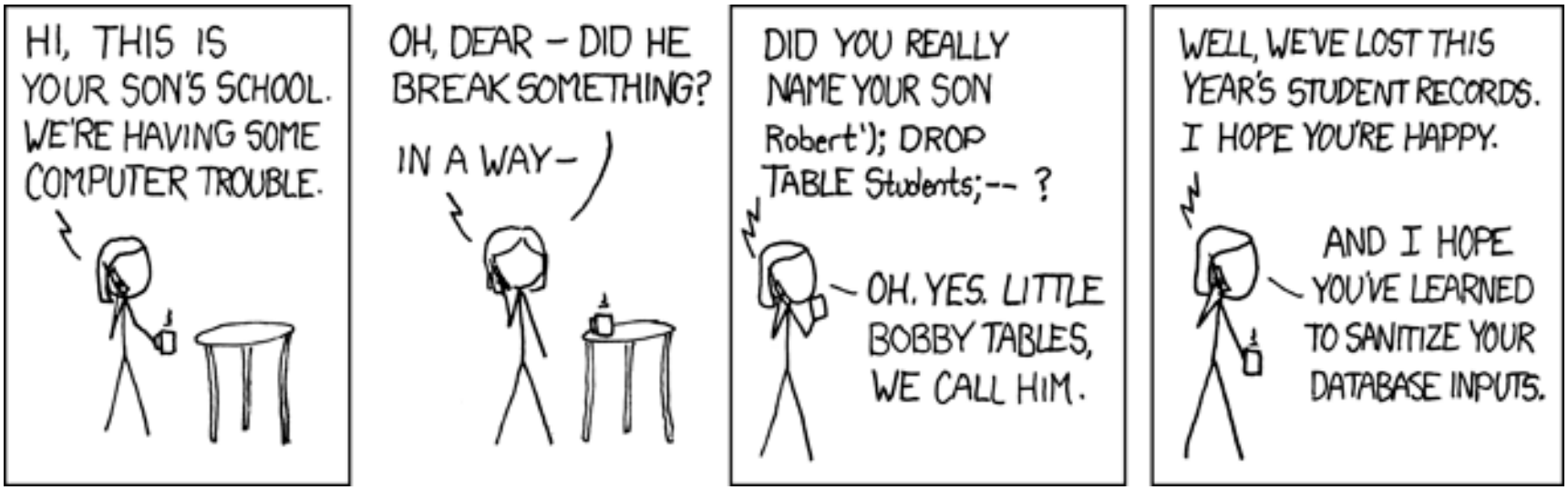
Internet Storm Center said "Patch Now!"

HashDoS (PHP)

PHP before 5.3.9 computes hash values for form parameters without restricting the ability to trigger hash collisions predictably, which allows remote attackers to cause a denial of service (CPU consumption) by sending many crafted parameters. from CVE-011-4885

DoS : attacks that prevent others from using a service

- Crash the service or the whole system
- Use up enough of something that no one else can work



Exploits of a Mom, everyone's favorite input checking use case

http://xkcd.com/327/

Exploit

Exploits ('**sploits**') are tools crafted to use a vuln

- Frequently built on or distributed with a framework (eg Metasploit) these days
- Can also be a standalone code module
- Published freely by some researchers
- Kept secret and hoarded by skilled attackers

"0day" refers to unpublished vulns/sploits

Exposure

Exposure: how much a vuln applies to your protected assets , and how that modifies the risk

e.g.: that Windows Media Player/Windows 7 vuln does not strengthen attacks on Amazon MP3/ Android …

But most corporations have a lot of exposure to Office or Acrobat vulns

In Risk Management this is calculated:

Risk = Vulnerability x Exposure

Minimal or zero exposure greatly reduces Risk!

Attacker

attacker: adversary, black hats (sometimes), Mallory, the antagonist in our story … them

Be careful characterizing attackers further:

- You may never know who/what they are.
- You may work in the next office over (I do).
- Poor strategy to make assumptions about the opposition without evidence.

Without further ado…

VULNERABILITY LIFE CYCLE

Your product released!

SDLC is mostly done

- QA is over
- Product ships
- Celebrate!

But keep in mind:

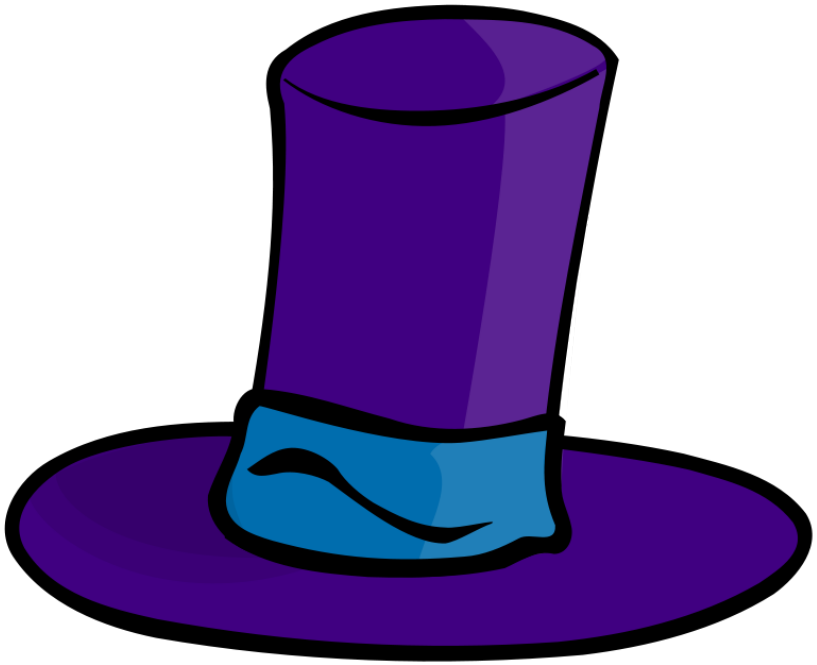
All software has bugs…

Vuln Research: who?

Someone will perform an investigation of your product (the SUT) to find its weaknesses:

- black hats
- grey hats
- scrooge hats …

Motive doesn't matter to their techniques, which should look familiar to BBST students…



Vuln Research: Test Design

People are going to look for vulns in your product using software testing techniques!

Information objectives, **resources**, and what they do with their **findings** vary a lot by their goals:

- sell the vulns? There's a market…
- develop vulns into exploits -> build attack tools
- develop protection -> build defense tools, signatures

and some involve more testing than others.

Vuln Research: Techniques

Black Box

- Fuzzing …

A *heartless* combination of input testing with endurance testing

- Every kind of quick test
  - Boundaries
  - Configuration changes
  - Timing and Races
- input and output analysis

White Box

- Reverse Engineering (disassembly, debugging, etc) ,
- static and dynamic code analysis
- virtual environment simulations

Vuln Distribution

Vulns (and sploits) are disseminated a variety of ways, including:

- directly to peers as research findings (such as in journals or at conference)
- publicly on the WWW or mailing lists (CVE, BugTraq, SecurityFocus, FD lists)
- privately through closed lists and commercial services (CTU, LANDesk, Secunia, Sourcefire VRT, X-Force)

Vuln Distribution

and in all cases this leads to more testing, by other researchers to reproduce, validate:

- private citizens and lone basement hackers
- IT security teams at organizations
- and attackers !

all with the same set of **information objectives**:

- how much trouble will this cause me?
- or can I cause with this?

Vulns get tested more

attack and defense tools both get tested though methodologies vary:

attackers may:

- beta their attack tools on unsuspecting targets,

as is the case with many new botnet and virus outbreaks

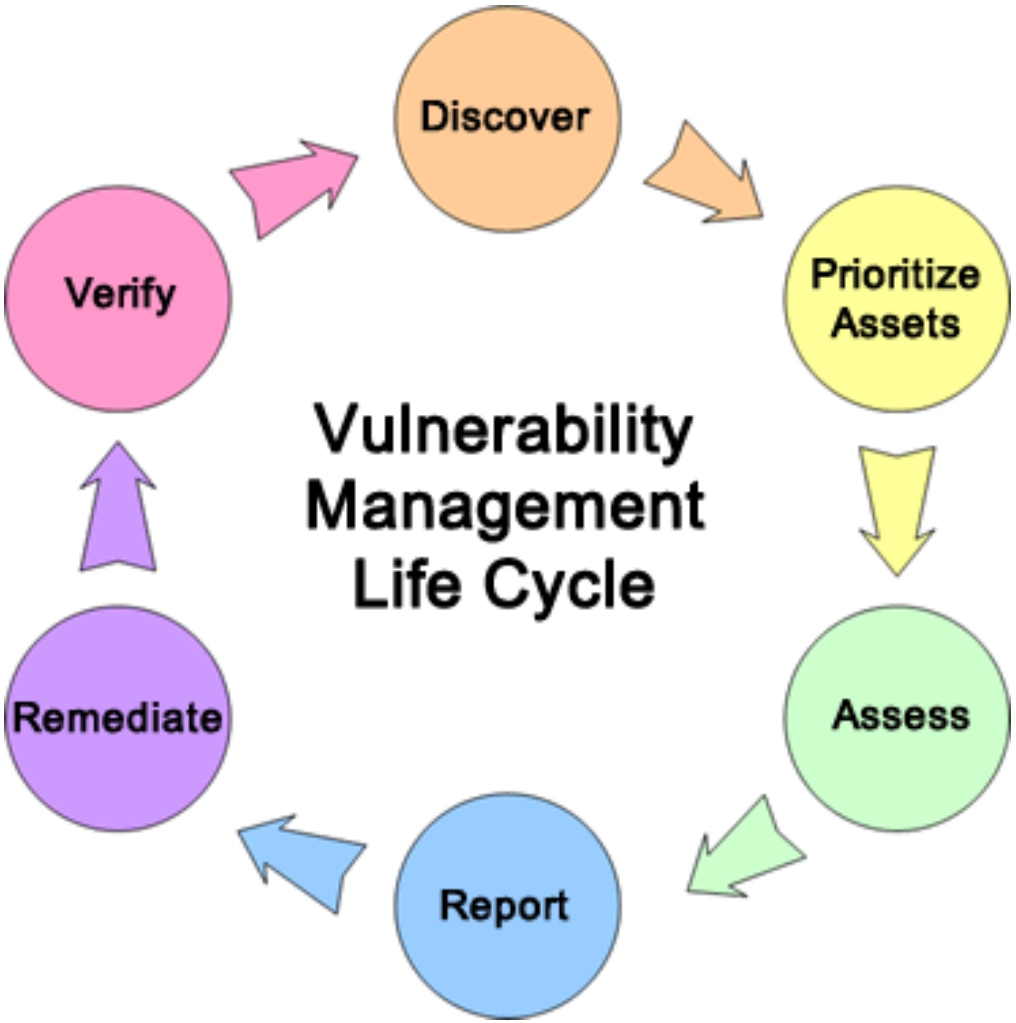
Or they may also choose to

- put their tools through a formal QA process

defenders may

put their tools through formal QA process including:

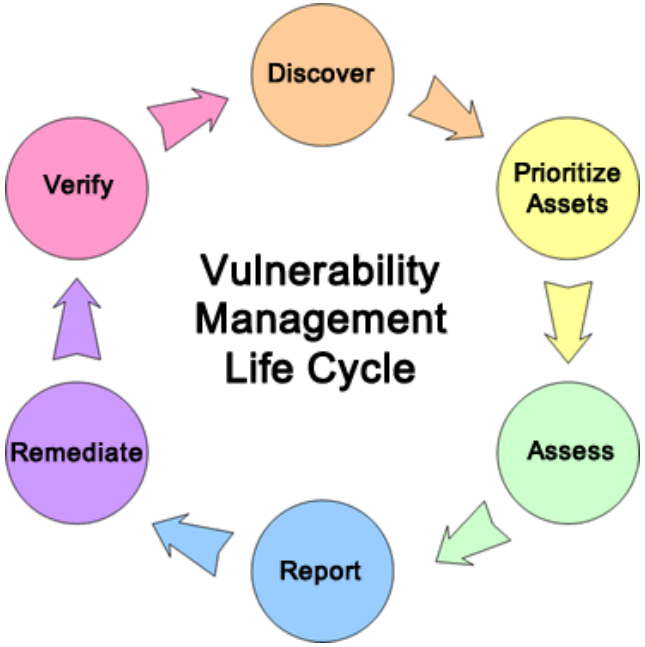
- regression suites
- configuration analysis (coverage)
- simulated attacks in a mock environment (scenario testing)



Vuln Assessment and Management

This is where the popular cycles (and products) start, such as this nice example from the CDC.

Vulnerability Assessment



Depending on the org scale, threat modeling, regulatory requirements, and IT staffing ...

Most people/places run wide-scale testing against their systems to evaluate their exposure to the new vuln as part of planning their response.

Vulns become the oracle

VA is (in tester terms) running a handful of tests

- on a large body of subject SUTs
- against the expected results oracle of the vuln (or tool) you are evaluating.

Commonly done with specialized free or commercial scanner software like: EEye, Nessus, OpenVAS, Qualys, Rapid7, w3af

but you can get pretty far with simple tools and skill : cURL, LWP, netcat, nmap

VA examples

Safe, easy

Typical safe vuln scan will look for the versions of a package known to be vulnerable. If Vuln says everything older than 7 is bad then 6.7 is bad and 7.1 is okay (even though available version 6.7.2 is fixed for this vuln).

Risky, good

A less typical and less safe but more accurate test would try and exploit the vuln (possibly with an attack tool) to see if they could achieve the dire outcome the researchers have trumpeted. (exploratory risk-focused testing)

QED: If I can bring up an root (system administrator's) console through your web site shopping cart application you have a problem no matter what your VA product reports.

Authenticated Scans?

scan : authn scan :: blackbox test: whitebox test

... well, almost

Unauth scan looks of ways to get in and outdated software (inside or outside)

Auth scan has a valid login and looks for way to exceed privileges, disrupt service, copy data you shouldn't be able to

Both are necessary for complete VA!

Remediation

Big word, means : **Fix it!**

- “Patch” software: install updates
- Update configuration
- Rebuild with better architecture
- Firewall it off from everything else

Sometimes it’s best to just:

- Turn it off and scrap it!

Remediation testing

- Many will opt to test the proposed changes on a lab or pre-production system.
- Mix of acceptance (smoke) and regression testing looked for unexpected unpleasantness in the “fix”.
- **Verification**: after someone says they are done "fixing it" you need to re-run the tests … and maybe run a few more just to be sure :)

Life Cycle

- Regression, compliance, verification tests for a particular vuln become routine.
- Meanwhile somewhere new vulns are being created and discovered.
- And the life cycle and testing begins anew!

Questions?

Q & A

Links

Black Box Software Testing courses:

- ***Foundations*** and ***Test Design***
- http://testingeducation.org/BBST/

Basic definitions and links to more:

- http://en.wikipedia.org/wiki/Application_security

SANS Institute:

- Internet Storm Center: http://isc.sans.org/
- GIAC Certifications: http://www.giac.org/