"Bruce"
A Java-based
Security Auditing Framework

Alec Muffett
Sun Professional Services
Alec.Muffett@UK.Sun.COM

Statement of Problem

- *Network vulnerability scanners* popular (SATAN, Nessus, Nmap, etc. . .)
- . . . difficult to scale to WAN-size, and/or distribute
- . . . only diagnose problems visible to network
- . . . views network as seen from one point only
- . . . painful to check some details (eg: promiscuous interface checking)
- So why can't we just look *inside?*

Statement of Problem

Looking inside:

- *Host vulnerability scanners* common.
  (COPS, fixperms, Tripwire, etc. . .)
- very powerful for analysing host weaknesses
- . . . but tend to stagnate quickly
- . . . and difficult to upgrade for each new bug

What is Bruce?

A *networked, host vulnerability scanner.*

- run daemon on every machine
- link daemons into tree-hierarchy (cf: DNS)
- use tree to distribute/execute security-checking code
- retrieve/centrally collate results
- view output with web browser (Netscape, Lynx, . . .)
- strongly tamper-resistant (ACLs, anti-spoof, anti-forge, anti-replay; secrecy & authentication via plug-ins)

Inside Bruce

Command the daemon to launch a security task (*pollet*) which in turn invokes *other* pollets:

- "AuditDispatch" ⇒ Audit + Dispatch
- "Audit" ⇒
  SystemCheck + UserCheck + NetworkCheck + . . . etc
- "SystemCheck" ⇒
  PatchRevCheck + FilePermsCheck + . . . etc
- "UserCheck" ⇒
  BadUserPermsCheck + RhostsCheck + . . . etc
- "Dispatch" ⇒ *punt request to "children"*

Inside Bruce

- The pollet maps to an implementing *package* using a table of OS, OS-revision, architecture and hostname:

```
Foo sunos 5.* sparc *.sun.com ⇒ foo-sol.jar
Foo sunos 5.* i86pc *.sun.com ⇒ foo-solx86.jar
Foo linux * i?86 * ⇒ foo-linux-x86.jar
Foo win 4.* * * ⇒ foo-nt.jar
```

Inside Bruce

- pollet generates output as object which contains HTML, text, images, etc. . .
- output percolates back up tree to parent – browsable via HTTP on localhost
- launch request propagates to "children" via recursive `Dispatch` pollet
- new/updated packages are automatically distributed along with launch request

Bruce Security

- packages have version numbers
- packages are digitally signed
- pollet ⇒ package mappings are digitally signed
- launch requests are digitally signed
- launch requests have non-replayable 64-bit random serial number

Bruce Security

- comms protected by strong authentication and optional crypto secrecy pluggable modules
- comms protected by TCP access control generated from above config files
- comms always initiated parent $\Rightarrow$ child (simpler, more secure, less DoS)

Benefits of Bruce?

- centralise security checking/auditing
- role-based delegation through certificates
- find and fix bugs in your network before "the bad guys" do
- write/use your own pollets in anything Java, Perl, C, Bourne-shell, X86 . . .
- AND. . .

Benefits of Bruce?

- free license for download and use
  (personal, academic, research, company internal)
- free license for source access

Java Development

Why use Java to implement Bruce?

Java Benefits for Bruce

- rapid prototype/development cycle
  (not as fast as Perl, but almost)
- good internal security
- easy access to crypto and networking
- ability to plumb ACLs and crypto into *all* comms
- zero network-buffer-overflow opportunities

Java Downsides for Bruce

- learning a new language from scratch
  (thank heavens for LISP experience!)
- language still developing and filling-out
  (moving target hard to hit)
- platform independent nature poor for probing
  platform dependent bugs (eg: SUID)
- Bruce by nature must run in a standalone JVM
  on most platforms

Java Solutions

- Speed: was a non-issue
- Platform Independence: use Java for "glue"
  use platform tools for platform tasks:
  perl, awk, find, df, . . .
- US Govt Export Regulations: write program to generic pluggable interfaces, use runtime-loaded code (cf: DLL's)

Java Issues

Mostly to do with sundry Java Virtual Machine (JVM) implementations:

- JVMs of variable quality
- JVMs hugely different memory footprints
- JVM implementations are out of [my] control

Future Directions

- further reducing code memory footprint
- bugfixes and functionality enhancement
- security refinements and extensions
- supported platforms: Solaris, Linux, NT, other...

Future Directions

- HTTPS browser interface (US Govt Export Control)
- crypto transport modules (Ditto)
- XML output for diffing against previous runs
- more pollets – both free *and* commercial

Download/Contact

http://www.sun.com/software/communitysource/senss/

bruce-feedback@sun.com