DISC - Seguridad en C’omputo 98

Auditing code for improving security.

Theo de Raadt
deraadt@openbsd.org

How does one audit code for security?

- What did the OpenBSD team audit?
- Why is OpenBSD a good case study?
- The approach taken.
- Lessons learned.
- How can you improve your ability to audit?

What did the OpenBSD team audit?

- A complete 4.4BSD operating system.
  - kernel, libraries, utilities, daemons, everything.
  - even the tar command.

- About 10-12 people on our auditing team
  - Some were not Unix experts when this started, but they are now.
  - Some auditors worked for Secure Networks and CORE

- The main auditing process took about a year
  - Thousands of problems discovered and fixed
  - A problem is not necessary a hole all by itself
  - We discovered many types of security problems

What did the OpenBSD team audit? (continued)

- **Year two: We continue to audit the code, but:**
  - **Far fewer discoveries nowadays**
  - **Significantly more obscure types of bugs**
  - **More 'incomplete' holes (as I say it, 'almost a hole')**
  - **Some auditors appear to have moved on because the time between finds is stretching (ie. less rewarding)**

- **We fully disclose our new discoveries at http://www.OpenBSD.org/security.html**

Why is OpenBSD a good case study?

- **Basically, there is proof of some success.**
  - **Very few problems reported on BUGTRAQ, CERT, L0pht, SNI, ...**
  - **Recognition for security in Wired (eek!) and many other magazines**
  - **The OpenBSD mantra: "We fixed that more than a year ago" (which really annoys some people)**
  - **Many security sensitive sites run our software**
  - **Developers who praise our security efforts**
  - **'Mimicry' in the FreeBSD and Linux communities (members of their communities admitting that they are way behind OpenBSD in security, but not following the process which led to our success)**

The approach taken.

- This is what I do when I audit, and what I ask auditors to do.
- This is basically what has been said on our web page for quite some time:
  - http://www.openbsd.org/security.html
- This is a best-effort type thing.

The approach taken (continued)

- File by file, line by line -- well, almost.
- Often we can concentrate on vulnerable subsystems first
- Including things often over-looked, such as:
  - Libraries, even include files
  - Obscure system call behaviours (or non-compliancies)

- Fix as many non-security bugs as you can!
  - Most security problems are just regular bugs
  - If you see a regular bug, why not fix it? (Imagine the shame if it turns out to be a hole)

- Actually, it becomes difficult to define whether a bug is security related or not ...

The approach taken (continued)

- A good auditor has to continually backtrack, ie:
  - Start with some knowledge
  - Audit a program: fix a few bugs
  - Audit a second program: fix a few bugs, spot a new type of bug
  - Go BACK to the first program, look for that new type of bug in it
  - Add a new program to the set, repeat. (After a few cycles it is easy to get bogged down)

- Most people who do audits today are quite unfamiliar with the detailed behaviours of the system they are auditing
- Often as unfamiliar as the author
- Hence, on-going education is the key!

The approach taken (continued)

- If you find a particular bug is rampant, it may be worthwhile to do a seek-and-destroy session. (mktemp races, estimated effort 50-60 hours)
- Eventually, this can lead to "find | grep" scans of the entire source tree searching for things which rely on a suspect or often-misused API. (like readlink)
- As the auditor gains greater familiarity with the system, he should spot new bugs easily. (We do, in a 300MB source tree)

Lessons learned.

- A bunch of us have gotten really good at spotting bugs in code
- It appears to be a learned skill, not knowledge
- We don’t know how to pass the skill of ‘discovery’ on to others
- Lists of old bugs do not teach people the skill of discovery!

- At times, some of us have simply sifted through source code for 6 hours, fixed 20 tiny bugs, then suddenly discovered a serious new issue which then took us 4 days to resolve.
- At other times, one of those tiny inconsequential-looking bugs has turned out to be a huge hole (years later, on BUGTRAQ).

Lessons learned (continued)

- On the whole, code quality just plain SUCKS.
  - Not just holes -- bugs, but...
  - 20 bugs will add up to become a hole... or maybe just 3 bugs?
  - We have found serious problems in pretty much everything.

- Programmers do not understand the APIs they code to.
  - Addendum: Programmers don't care to learn the APIs they code to.
  - A good programmer will make an API error about every 30 lines.
  - A novice programmer...

How can you improve your ability to audit (or write secure code)

- Become a better programmer by..
- Start studying standards documents
- Strive to completely understand the APIs you use (complete internal source code for most APIs is now available. Read it and learn!)
- Avoid the NIH syndrome. Look at what others do!
- Limit your trust.

Sponsoring our efforts

OpenBSD is a free unix-like operating system.

Binary releases are freely available for ftp. Our group also sells CDROMS to pay for our development costs -- without such sales we would not be able to continue our efforts.

For information about OpenBSD, start at

- **http://www.OpenBSD.org**

Our most recent release is OpenBSD 2.3. CDROM sets are available for $30 USD. For CD orders see http://www.OpenBSD.org/orders.html or via FTP at http://www.OpenBSD.org/ftp.html