The Experience of Having Condor Assessed

Zach Miller
Condor Project
Computer Sciences Department
University of Wisconsin-Madison

The Experience of Having Condor Assessed

- What is Condor?
- Why is assessment important?
- Assessment results
  - Vulnerabilities were found and fixed
  - Our lack of clear policies became clear
  - We developed new coding practices
  - We developed a process for disclosure
  - Other benefits

What is Condor?

- Condor is High-Throughput Computing software.
- Allows users to submit arbitrary executables to run on machines in some compute cluster somewhere (sound scary?)
- Is deployed on thousands of machines worldwide

Why Is Assessment Important?

- The scale of Condor deployment makes security issues a big problem
- It should result in more secure software
- It increases administrator confidence in Condor software
- People do ask if it has been reviewed

Why Is Assessment Important?

- As developers, we are focused on developing distributed computing software
- Even though we are aware of the security implications, secure programming is not the specialty of all programmers, and even very experienced programmers can make mistakes
- Therefore, the software does need to be reviewed

Why Is Assessment Important?

- As mentioned in earlier talks, a thorough assessment must be done by an outside party
- Developers are "too familiar" with the code to see it from a different perspective
- So, Condor's involvement in the assessment was fairly minimal, other than providing the code and documentation and answering the occasional question

Assessment Results

- Fourteen vulnerabilities were found and fixed
- They were also very well documented: http://www.cs.wisc.edu/condor/security
- But most importantly…

Assessment Results

- The finding and fixing of the vulnerabilities, while important, was not as crucial as the changes in our methodology that came out of the experience
  - η Coding practices
  - η Developer review of Condor code
  - η Vulnerability disclosure policy

Assessment Results

- Coding Practices:
  - We started systematically looking for entire classes of vulnerabilities
  - Automatic tools such as Coverity help somewhat with this, particularly for static buffer overflows and misuse of strncpy() and friends
  - However, Coverity in practice did not uncover many exploitable security problems

Assessment Results

- Coding Practices:
  - We started tracking vulnerabilities in our external packages, i.e. security libraries we link with such as Globus, Kerberos, OpenSSL, etc.
  - We also try to learn from their mistakes

Assessment Results

- As an example, there was a vulnerability in the Globus toolkit related to a race condition in creating a file and setting its permissions:

```
FILE *f = fopen(filename, "w");
int ret = chmod(filename, 0600);
```

Assessment Results

- We looked in the Condor code for the same type of race condition…
- And found an instance in our own code!
- So, with Jim Kupsch, we create a wrapper for opening files so that race is avoided
- Finally, we made the use of regular open() and fopen() a compile-time error to prevent new code from having the same problem

Assessment Results

- We also started having all checkins reviewed by another developer before being committed
- This has helped catch many bugs before they even became part of the code base

Assessment Results

- Procedural Changes
  - Now that we know we have vulnerabilities, what do we do about it?
  - We discussed our policy with our downstream consumers like the VDT and RedHat
  - They need time to get the patched binaries into their packages and pushed to their users

Assessment Results

- Condor policy for announcing vulnerabilities is now this:
  - Release new binaries with the vulnerability fixed
  - At the same time, include a notice that there is a vulnerability, with high-level information about it that is not descriptive enough to actually create an exploit
  - After 30 days, publish full details on the vulnerability

Assessment Results

- That policy works well when the vulnerability is discovered "internally"
- If the vulnerability was discovered "in the wild," we assume that exploit code already exists and we release new patched binaries immediately, out of band from our regular release cycle

Assessment Results

- As a result, we needed to change our source tree management somewhat so that we could immediately release binaries with the vulnerability patched but no other code changes
- This sometimes includes backporting the patch to older stable series that are still supported

Assessment Results

- A **huge** benefit to having undergone the assessment is that all discovered problems were reported directly to us, the Condor developers, therefore allowing us to follow the same procedure as if it was discovered "internally", rather than "in the wild"

Assessment Results

- Open Source Considerations
  - How do you make a new release without disclosing a vulnerability?
  - One option: release the source anyways
    - This could allow someone to figure out what changed and how the old code was vulnerable
  - Another option: release only binaries
    - You could argue that by decompiling this as well allows someone to figure out what changed

Assessment Results

- Condor has opted to release the patched source as well, so that people who rely on compiling from source are not excluded from getting the vulnerability patched
- We still do not publish explicit exploit details for 30 days
- Obviously, this is not ideal since the code diff yields information about the vulnerability.
- This is not an easy issue to solve

Conclusions

- Many vulnerabilities were found and fixed
- This is wonderful, but not as important as how we changed our processes to mitigate future problems in new code
- Thinking about the issues involved in disclosing vulnerabilities, quickly releasing patched binaries, and open source questions has helped us make formal procedures for doing so and making them well-known
- Likely, more vulnerabilities exist, but we feel better about that now that we know how to deal with them

Conclusions

- In the same way that software is licensed via well-known licenses such as the GPL, freeBSD, or Apache licenses, we are looking to standardize a procedure for disclosing vulnerabilities and releasing binaries and source code
- We look forward to future work with the assessment team on our privilege separation work and beyond…

Questions?

- Ask me!
- Email me: zmiller@cs.wisc.edu
- Visit our homepage:
  http://www.cs.wisc.edu/condor/