What is a Security Flaw?

Colin Percival

cperciva@freebsd.org

What is a Security Flaw?
(Questions in Security: The Bad, the Ugly, and what we can do about them.)

Colin Percival

cperciva@freebsd.org

What does a Security Officer do?

- "Find security flaws and fix them"
  - What is a security flaw?
  - Do all security flaws need to be fixed?
  - How and where should they be fixed?
- In practice, the job is...
  - 60% making policy decisions,
  - 30% administrative,
  - and 10% technical.

1. What is a security flaw?

- FreeBSD:
  - Any privilege escalation.
  - Any disclosure of potentially sensitive information.
  - Any *remote* denial of service.
  - Local denial of service attacks are not handled as security issues.
    - Any sufficiently advanced local DoS is indistinguishable from a user trying to get a lot of work done.
    - Some local denials of service are corrected as Errata.

1. What is a security flaw?

- Linux (most vendors):
  - Any privilege escalation.
  - Any disclosure of potentially sensitive information.
  - Any *remote* denial of service.
  - Any *local* denial of service.
    - Keeping kernel + world separate makes this easier.
    - Most Linux vendors correct local denial of service issues in batches, and only issue one advisory for the entire group.

1. What is a security flaw?

- OpenBSD:
  - Not even remote denial of service.
  - In January 2006, a bug was found which allows a remote attacker to send three IP packets and cause a kernel panic.
    - FreeBSD issued FreeBSD-SA-06:07.pf.
    - OpenBSD does not mention this bug on their “release errata & patch list” web page.

1. What is a security flaw?

- Intel:
  - Not even local privilege escalation.
  - “In order for this particular exploit to be launched on a system, the system has to already have been compromised” (May 2005).
- Several LKML posters:
  - There's no point fixing local privilege escalation bugs, since attackers will always be able to find more of them.

2. Would anybody really do that?

- Sometimes the conditions needed for a bug to be exploited are so bizarre that it does not need to be treated as a security issue.
- Bug discovered in sh(1): If a program is being run with two here-documents, and the first here-document includes backticks, the second here-document will be executed.
  - Data is being treated as code!
  - Who ever executes shell scripts constructed using untrusted input?
  - Why place untrusted input into a here-document instead of redirecting from a file?

2. Would anybody really do that?

- Bug discovered in qmail: If you can send a >2GB message to qmail-smtpd, you can execute arbitrary code via an integer overflow.
  - Response from DJB: “Nobody gives gigabytes of memory to each qmail-smtpd process”.
  - When DJB wrote qmail (1995), this was probably correct.
- Documenting what you rely upon users never doing is a good idea.

3. Where is the security flaw?

- Cryptographic information can leak via timing channels in the cache on processors with Hyper-Threading.
  - Cryptographic code manipulates key information in non-oblivious ways.
  - Processors leak information about memory access patterns.
- Which component is at fault?
  - Is it reasonable to expect information about memory access patterns to not be disclosed?

3. Where is the security flaw?

- Hypothetical bug in sort(1): Every time “aaaaaaaa” should be output, “/etc/master.passwd” is output instead.
  - Is this a security flaw? Not really...
- Behaviour of portsnap client:
  - Download a database file, sanity-check its contents, use text file manipulation utilities including sort(1), and use the resulting text as file names.
- Portsnap together with this pathologically buggy sort(1) is insecure, even though *neither program has a security flaw.*

What can we do about this?

- Security questions become difficult when interfaces are unclear.
- An interface is a contract.
  - Some interfaces are written in formal languages and define the exact behaviour required.
    - Formal languages are great, but very few people actually use them.
  - Most interfaces are very vague.
- Most API contracts are violated due to bugs.
  - Fortunately computer programs don't have lawyers.

Violating contracts

- What happens when real-world contracts are violated?
  - Lawyers go to court.
  - In the next version of the contract, lawyers add more fine print.
- "If you are injured while skiing at our resort, we will *try* to evacuate you safely, but *we do not guarantee* that we will do so."
- We should add fine print to interface specifications.

Fine print

- Separate the behaviour which is *expected* to be provided from the behaviour which is *guaranteed* to be provided.
- Algorithmists have done this for a long time:
  - Quicksort takes O(n log n) expected time, but $O(n^2)$ guaranteed time.
    - Don't use quicksort for sorting data provided by an untrusted source!
- Most code can break without introducing security issues.

Benefits of fine print

- Fine print makes the Security Officer's job easier.
  - It is immediately obvious whether an bug needs to be treated as a security issue.
- Fine print allows developers to indicate the state of their code.
  - Most developers don't want to say “I was drunk when I wrote this, so don't trust it for anything important”.
- Clarifying which code has security implications allows eyeballs to be concentrated on the most important code.

Conclusions

- Security is hard.
  - Especially when you have several independent components.
  - Especially when interfaces are poorly documented.
- Rather than aiming for zero bugs, we should aim for zero *security* bugs.
  - Failing that, we should document what sort of bugs will be treated as security issues, so that users of our code know what they can rely upon.

Supported by:

www.freebsdfoundation.org

www.irmacs.sfu.ca

Paper:

http://www.daemonology.net/papers/codingbycontract.pdf