Jingtao Wang – Security
4/22/09
Adam Kauk

The phase 4 implementation doesn't need threads because read and accept are non-blocking.

Review: Authentication vs. authorization. Authentication is verifying who a user is; authorization is allowing a user to perform an operation.

Access control matrix: Whether a user can read or write a file is controlled by the access control matrix, except that it isn't actually implemented with a a matrix, because the matrix would be so sparse that it wouldn't be worth it. Instead it is either implemented with an access control list (associated with a file) or a capability list (associated with a user).

An access control list is used more that access control list, because it makes it easier to revoke capabilities.

Challenges to access enforcement

- Abuse of privileges
- Imposter of Trojan Horse
- Listeners
- Spoilers (using up all of the resources)
- Doctored versions of standard program (this is especially easy with open source software, e.g. firefox)

Penetration: phishing, fake shells, fake software, people forgetting to log out.

Buffer overflow
Strcpy is vulnerable to buffer overflow, as are other functions that use buffers

Possible alleviations:

- Signatures – put a signature after the stack variables, so that if the signature is modified, we know that a buffer overflowed. Problem: People can avoid overwriting signatures; thus, this method is not fool-proof.
- New functions (instead of strcpy). Problem: good, but nobody uses them.
- Checking the length of buffers. Problem: same as last.
- Don't use C. Problem: this idea is 30 years too late.
- Program scanners. These programs will warn you when you might be opening your program to buffer overflows. Problem: not all warnings are actual vulnerabilities. Also, there may be hundreds of these warnings.

Famous stack overflow example: Morris Internet Worm

Timing attack: Tenex password checking: Store the guessed password in such a way that it overlaps pages in memory. How long the authentication takes will tell whether the first part of the password was correct or not. In this way, a password can be discovered one letter at a time by brute force. This problem has since been solved by making programs waiting a set amount of time before responding to a password authentication with true or false.

Once a system is penetrated, it is often impossible to resecure.

Countermeasures to password stealing:

- Call backs: When a user logs in, the server drops the call and then calls back to the number where the server is supposed to be. If it was a false user that signed it, it is thus stranded.
- Being a non-privileged user: When you are a non-privileged user, a virus that you somehow allow in also does not have privileged user status.
- Use plausibility checks on money transfers
- Get humans involved (e.g. humans have to approve money transfers)

Inference Control (i.e. how do we make sure that people can't figure out private information from public information):

- There is no good solution
- Possible solution: randomize data—i.e. introduce small error
- Possible solution: restrict queries that are allowed of the public data

Confinement problem: How do we protect users and businesses who are both suspicious of each other.

- Some services leak information
- Some services collect information over time, which can be dangerous
- Sometimes, confidential information is encoded in bills.

Viruses (they are bad)

Antivirus techniques:

- Scan for virus signatures (this may need to involve decryption).
- Run on virtual machine, then search for virus signatures again.
- There are no good solutions (is this problem NP complete? Is it harder?)