

IY5512: Coursework 4:

Worked solutions

1. *Outline the 'penetrate and patch' approach to developing secure software, and describe three shortcomings of this approach.*

'Penetrate and patch' is an approach to managing software vulnerabilities taken by some major software vendors. It involves releasing software into the commercial market that is poorly designed from a security perspective. Patches for downloading by users are released when a security flaw is discovered.

There are a number of significant problems with this approach:

- vulnerabilities must first be found;
- hastily written patches may introduce new vulnerabilities;
- there is no guarantee that users will install patches;
- this is a lazy approach that essentially depends on users doing the testing after release;
- it is damaging to a vendor's reputation with users.

2. *Briefly describe two principles which can be used to aid in the development of secure software.*

Principles described in the course include the following:

- **Secure software from the outset.** To comply with the software security principle 'secure software from the outset' it is necessary to commence any software development process with a thorough analysis of the security requirements of the intended application. Adding features to an existing design in order to correct vulnerabilities does not address the underlying problem; either the security requirements were incorrectly specified or the implementation did not meet the security requirements. Capturing security requirements is essentially a risk assessment exercise.
- **Secure Software by least exposure:** ensure that software is not exposed to more risks than necessary; isolate code for security critical operations in separate modules or libraries, making it easier to analyse and control; enforce the principle of least privilege: grant the minimum access necessary to perform an operation, and only for the time needed to complete it; minimise the possible 'attack surface' by turning off all unnecessary functionality and services.
- **Secure software by default:** software should fail securely, i.e. failure (e.g. as a result of inappropriate input or a lost connection) should not expose the system to security vulnerabilities. Software that generates a 'world readable' core dump after a crash is problematic, because the core dump

may contain security critical information such as password entries. A firewall that continues to operate when the disk on which it stores its log becomes full, could lead to a failure to detect an important security critical event.

3. *What is a buffer overflow and how can it give rise to vulnerabilities in software?*

The allocation of memory resources is probably the most common area where software security vulnerabilities arise. The majority of these problems arise from buffer overflows. Buffers are sections of memory allocated for data storage. Storage of a value exceeding the size of the buffer may result in other memory locations being overwritten. A buffer overflow occurs when a program writes data beyond the bounds of the allocated buffer.

A buffer overflow can give rise to many different types of security problem, e.g.: execution could continue, and the overwritten memory will contain a changed variable value; the code might perform an unintended task; or the code might crash.

4. *Outline two ways in which the risks from buffer overflows can be reduced.*

Occurrences can be reduced by:

- careful coding, avoiding dangerous calls;
- software scanning tools can be used to find and remove buffer overflow weaknesses;
- programming in a type safe language can reduce vulnerabilities;
- apply the principle of least privilege when designing software, so any buffer overflows that do occur will result in minimal damage.

5. *What is type safety, and how can its use help to make software more secure?*

Types specify that the data held by variables must have certain properties: e.g. variable *Count* will always hold an integer, or Array *A* will never store more than 10 items. Type checking verifies these properties. A language is said to be **type safe** if the properties are guaranteed to hold at run-time; if there is no such guarantee then the language is said to be **unsafe**.

A safe language can guarantee memory safety by preventing errors arising from misuse of types. By contrast, static checking (at compile time) can make some buffer overflows impossible but cannot help with mobile code from other sources, and dynamic checking (at load time) could be used to give some basic checking of mobile code.

6. *Give an example of a piece of malware which exploits a failure to validate data input. Describe briefly how it operates.*

The Microsoft Remote Desktop Protocol (RDP) service contained an input validation error that can be exploited to cause a denial-of-service condition. A remote attacker

may be able to exploit this vulnerability by sending a system running the RDP service a specially crafted message on port 3389/tcp. For further details see:

<http://www.kb.cert.org/vuls/id/490628>

Helpful discussions on input validation can be found here:

<http://cwe.mitre.org/data/definitions/20.html>

and here:

http://www.owasp.org/index.php/Data_Validation

7. *Give an example of a piece of malware which exploits a data type error. Describe briefly how it operates.*

Old versions of ssh had an integer overflow problem which could be exploited remotely. The exploit caused the ssh daemon to create a hash table of size zero and overwrite memory when it tried to store values in the table. More details on the ssh integer overflow are available here:

<http://www.kb.cert.org/vuls/id/945216>

A nice discussion of integer overflows can be found here:

http://www.owasp.org/index.php/Integer_overflow

8. *Give an example of a piece of malware which exploits a buffer overflow. Describe briefly how it operates.*

The Morris worm spread in part by exploiting a stack buffer overflow in the Unix finger server. The Witty worm spread by exploiting a stack buffer overflow in the Internet Security Systems BlackICE Desktop Agent. The Slammer worm spread by exploiting a stack buffer overflow in Microsoft's SQL server. The Blaster worm spread by exploiting a stack buffer overflow in Microsoft DCOM service.

Helpful discussions of buffer overflows can be found here:

http://www.owasp.org/index.php/Buffer_Overflow

here:

http://en.wikipedia.org/wiki/Stack_buffer_overflow

and here:

<http://projects.webappsec.org/w/page/13246916/Buffer-Overflow>

and in many other places on the web.

9. *In the context of computer malware, what is the main difference between a virus and a worm?*

Both viruses and worms propagate themselves, but in different ways. A virus is a program fragment that infects other programs by modifying them to include a copy of the virus code, which can then go on to infect other programs. A worm is a replicating but non-infecting program that uses network connections to spread from system to system.

10. *Give real-life examples of a worm and a virus, and briefly explain how these particular examples propagate themselves.*

Concept was the first widely disseminated macro virus for Microsoft Word. It spread via infected word documents, e.g. sent as email attachments. Infected files were 'regular' Word documents that contained a special macro containing the virus. When it was originally circulated it took advantage of the fact that the macro system in Word had essentially unlimited powers and users were not warned about macros present in files that they opened. As a result of this virus (and other related viruses), Microsoft put much stricter controls on the operation of macros.

The 'original' computer worm was (perhaps accidentally) unleashed on the Internet by Robert Tappan Morris in 1988. The Internet Worm used sendmail, fingerd, and rsh/rexec to spread itself across the Internet.