**Learning Outcomes**

|  |
| --- |
| * Demonstrate understanding of the factors that put computer systems at risk of failure * Identify suitable solutions to common threats and weakness faced by modern computer systems |

**Summary**

|  |
| --- |
| There are two sources of threats to a system’s stability. Firstly viruses. These can be accidently introduced through the internet or removable storage. Secondly, exploits of the system architecture. Attackers seek to take advantage of hardware behaviours and interactions.  Virus checkers and firewalls can help protect a system but come at a performance cost.  Data can be made resilient to faults and errors through encryption or validation, whereas a system can be made resilient through redundancy and UPS systems. |

**Lesson 1: Threats and Weaknesses**

|  |  |  |
| --- | --- | --- |
| **Viruses and Exploits**     * **Viruses** * **Virus Checkers** * **Virus Examples** * **Architectural Exploits** |  | Security threats to systems can only happen when unsafe entities are introduced. It should be almost impossible for a system that is never connected to the outside world to be exploited.  Networks and removable storage introduce a security risk to the system. Threats could include   * Data or identity theft * Corruption of data * Denial of access to data, service, or system * Misuse of resources   Viruses are programs written to maliciously access or alter the behaviour of a system.   * As programs, they require execution to perform their task * This usually happens when a user is deceived in some way * Alternatively, security flaws in programs may allow execution without asking permission * A **boot-sector virus** will execute the code on start-up, without a chance to intervene   Restricting these areas reduces threats, however the internet provides a continued source of threats.  Anti-virus software attempt to inspect all information being transferred by IO devices. They try to match data with known virus data patterns and intervene before they can be executed.   * This is done in a streamlined fashion: not byte for byte * Instead, inspecting for patterns speeds up the task but still impacts transfer rates * New viruses are written to circumvent such protections * Polymorphic viruses even mutate their code structure to create passable duplicates   Sophisticated viral impacts include ransomware, keyloggers, and creation of botnets.  These range from annoying to catastrophic, depending on the nature of the machine infected.  Historically notable viruses include:   * CyptoLocker – ransomware that encrypts files * ILOVEYOU – spread by email and overwrites system files and user data * MyDoom – created botnets to inflict distributed denial of service attacks * Storm Worm – spread by email and sends spam emails * Sasser – spread by scanning remote network ports * Stuxnet – disrupted operating parameters of Iranian uranium enrichment facilities   It is possible to attack a system through information leaked from a computer system during its normal operation. These **side-channel attacks** are hard to detect and can use hardware or OS behaviours.   * TLBleed sought to exploit the relationship between memory and the Translation Look-Aside Buffer (TLB), though only to prove the vulnerability exists * Spectre targeted the timing of events and behaviours during speculative execution to uncover information that should have been private to the process * DMA side-channel attack involves a device initiating memory data transfers at a hardware level, bypassing the OS and software level memory privacy controls   The complexity of modern systems makes it difficult to provide absolute security of systems, sometimes leading to features, such as hyperthreading, being disabled while fixes are devised. |
| **Firewalls** |  | Firewalls are filters built into systems and network infrastructure components to block certain types of data and limit or block access to resources. They attempt to block unauthorised third-party access.  Service attacks, like denial of service attacks, seek to overload a server or network connection through randomly generated requests. This results in slow performance for normal users or complete service failure.   * In-house servers can only really be protected by blocking the IP addresses with a firewall * However, many attacks circumvent this by automatically switching IP addresses * Cloud computing services can activate extra servers to meet increased demand * This allows service to continue normally while a solution is found |

**Online**:Section 7.3-7.4, Computer Architecture and Operating Systems, University of York

**Print**:Chapter 14.1-14.5, Computer Architecture and Operating Systems, Crispin-Bailey

Chapter 3.10.3-4, Computer Systems: A Programmer’s Perspective, Bryant et al

**Lesson 2: The Resilient System**

|  |  |  |
| --- | --- | --- |
| **Encryption and Validation** |  | Data theft can be protected against using encryption.   * Data sent by browser can be encrypted to a point that attempting to decode takes too long * All algorithms can be broken, but time and data value make some unviable targets * **Quantum cryptography** may provide this but is not yet deployable for everyday use * Example encryption standards include AES, RSA, DES, and SHA   As computing power increases, encryption standards become weaker over time.   * This means they become breakable with less expensive computers. * Thus, standards are strengthened over time, eg from 64 bit to 128 bit * However, more complex encryption requires more CPU effort and delays data transfer * This is compounded by the need for constant checking of the data for viruses |
| **Data Resilience**     * **Parity** |  | A key aspect of data resilience concerns detecting and ideally correcting errors at the bit-level.  While errors may be handled by systems and hidden from the user, it is rare for no errors to occur.   * Context informs error acceptability * Momentary disruption of communication over satellite phones may be acceptable * Similar disruption to digital stock trading may not be   Error detection and correction comes at a cost.: The more resilience needed; the more effort required.   * One suitable method is to **handshake** important data exchanges * When a signal is sent, it is returned to the original sender to ensure data is the same * However, methods that allow the receiver to act independently are also needed * Parity checks and cyclic Redundancy Checks (CRC) are often built directly into DRAM * These can also be applied in data storage, data transmission, and parts of the OSI model   Parity checks involve a parity bit being added to the data transmitted:     * The simplest method involves adding a parity bit to each byte * **Even parity** adds a 0 or 1 bit to make the sum of bits in a row or column even * This will detect single errors in a row, but multiple errors may cancel each other out * Thus, **horizontal parity checks** are resilient for single bit errors, the most common * This method can be extended to a whole data block for **vertical parity checks** to occur * At least 4 bit errors in the same rows and columns are needed for an undetectable error   No system is 100% error resilient and there is always a cost.   * The example above there are 40 bits, and dual parity checking takes this to 54 bits * This means 35% more bits need to be transmitted and stored * Therefore, consideration is needed between data size and severity of an error occurring |
| **Resilient Systems**     * **UPS Systems** * **Resilience and Safety** |  | All system components can fail and the **Mean-Time Between Failure** (MBTF) indicates how likely this is.     * An HDD may have an MTBF of 50,000 hours of operation * This is an average not a guarantee: it could fail in 30,000 or continue past 60,000 * Quality manufacturing usually means the majority fail within a narrow band as above * The probabilistic curve carries on indefinitely in theory   The probabilities of failure in different scenarios can be calculated as follows:     * In a system, the various components will have their own **independent failure rate** * A **critical failure** occurs when a component failure causes the system to fail * This can be reduced by introducing redundancy, such as paired redundancy above * In the example above, both drives need to fail in the same period for a critical failure * However, the chance of any component failing increases as there are more components * Avoiding double failures by replacing single failures promptly lowers the risk to effectively 0   A system will also fail without electricity, so mechanisms need to be put into place to safeguard against this.  An **Uninterruptible Power Supply** (UPS) is a short-term reserve of power can be provided to systems.   * A UPS will detect a power outage and indicates the system should begin shutting down * It gives a system time to send memory to disk, close files, and stop critical processes * In professional systems or server farms, UPS can be complicated and expensive * However, they can potentially allow limited critical systems to operate for hours   While critical failure can be managed, no system can remain fault free forever.  When **safety critical systems** do fail, they need to do so in a way that ensures a safe outcome. This is required in areas such as the industrial, medical, automotive, and aerospace industries.   * In a self-driving car, the processor interpreting sensor data may be a critical point of failure * **Triple redundancy** could be used, decisions being a consensus among three processors * If a processor disagrees, maybe due to a fault, the car could be configured to pull over * The chance of both remaining processors failing before the car pulls over is negligible * The intended behaviour of safely pulling over to shut down is called **graceful degradation**   Some processors support the concept of lock modes. This pairs two cores together to execute instructions together step by step. If the output of one varies, this indicates a fault. |

**Online**:Section 7.5-7.5.1, Computer Architecture and Operating Systems, University of York

**Print**:Chapter 14.6-14.11, Computer Architecture and Operating Systems, Crispin-Bailey