for the operating system swap file. The cost of suitably sized flash devices has come down greatly over recent months and even eight Gigabytes is very reasonably priced – especially when considered against the price of a laptop.

The second consideration relates to the lifetime of the memory stick, as flash can only be re-written a finite number of times. The number of erase cycles that a flash device can sustain is in proportion to the quality of memory stick.

Flash memory is a type of EEPROM memory. There are two types of flash memory: NOR gate and the more modern, cheaper NAND gate types. Modern USB memory sticks are all based on NAND memory. Both of these technologies allow essentially arbitrary programming, but can only be erased on a block level. An empty block can be written to any value, but to rewrite it requires emptying the entire block. An average quality flash memory device can sustain in the region of 100 000 cycles per block. Any changes that are written to a block require the use of a whole erase and reprogram cycle.

If data is written to the same blocks over and over, then the drive will fail more quickly than if the whole memory were used proportionally. This is particularly important when using flash to store operating system data, as the areas of the drive that hold swap file and disk caches may wear out prematurely. Some more expensive flash memory devices include firmware that intelligently utilises different areas of the disk to decrease wear. Bad block marking can also be used to improve the life of flash memory by forcing the driver to write to a fresh location if a bad block is detected.

Other strategies can be used to decrease the impact of excessive writes on the flash media. Some virtualisation software uses an area of the host operating system disk as a 'scratch area'. Disk writes are cached on the hard disk meaning that changes need only be written to the flash media on operating system shutdown. This can be advantageous in two ways by both improving the life of the flash media and potentially improving performance by using local hard disk writes over flash memory on a slow USB bus.

The speed of the flash memory and the USB bus device is also important. This can be improved by allowing the host operating system to perform read-ahead and write caching in memory. If the host operating system is Windows Vista, SuperFetch pre-fetch caching may be of particular assistance. The implementation of these speed improvements would be subject to the operating system drivers and the features of the memory drive itself.

There are USB devices on the market that combine fast-access, high-quality flash memory with physical security features. Hardware encryption of the entire flash memory is possible, using military grade algorithms and components. In addition, some devices include tamper resistance features, water proofing and tough metal cases. The virtual desktop environment works in exactly the same way on these devices. While these devices are more expensive than even a decent quality 'normal' USB storage device, they may make the adoption of the virtual desktop more palatable to even the most diehard cynical IT security officer.

About the author

Tom Rowan is the principal security consultant in the UK for IT solutions provider Magirus, and has over 11 years of security experience, In his spare time, Tom is also a RAF Reserve Officer with the Air Training Corps.

It's the software, stupid

Ed Ray, information risk strategist, Getronics

According to the 2005 US Federal Bureau of Investigation (FBI) Computer Crime Survey[1] (see Figure 1), 98% of enterprises in the United States deploy antivirus software on their systems and 91% also use perimeter firewalls. The survey also indicated that 84% of enterprises were attacked by worms or viruses, and 79% experienced spyware penetration. These statistics show that attacks persist despite employing the best protection measures.

Compliance standards such as the Payment Card Industry (PCI) Data Security Standard have sought to mandate that certain security measures be taken to ensure data protection.[2] Yet, the latest exposure of millions of credit and debit card numbers by Hannaford Bros., a grocery chain with 271 locations in New England and Florida, shows that the value of PCI compliance may not mean much when it comes to securing computer networks.[3] Approximately 4.2 million credit card numbers were exposed, and the fraud cases associated with this breach have reached 1 800 thus far. Even a bank as large as France's Société Générale, with presumed security controls in place to prevent such an occurrence, allowed a rogue trader to lose $7 billion through fraudulent trading.[4]

Although these examples illustrate a wide variety of security issues, the solution is not going to come from new security products. In the manufacturing industry, Motorola developed a set of practices known as Six Sigma,

which measures quality of a product.[5] Six Sigma's implicit goal is to improve all processes to defect levels below 3.4 defects per (one) million opportunities (DPMO). This is roughly equivalent to saying that 99.999999% of products produced are good or free of defect. In security products almost the reverse of the Six Sigma effect has been reached; 99.999999% of security products are of little use in securing networks from compromise.

Solutions: people and processes

While this may be a gross exaggeration, the truth is that all the security products released in the last seven to 10 years have done little to decrease the frequency or severity of security breaches. The solutions lie, therefore, not in products but in processes and people, a conclusion that presents further considerations. According to the IT Governance Global Status Report 2008, (complimentary copies may be downloaded from the IT Governance Institute (ITGI) Web site at www.itgi.org), 58% of respondents noted that their organisations had an insufficient number of staff, compared to 35% in 2005; and 38% pointed to problems relating to staff with inadequate skills.[6]

"There is a growing consensus that the software industry needs actively to address root causes of exploitable vulnerabilities"

How can the risk of future compromise be reduced? The answer begins with identifying the root of the problem: insecure software. While organisations have become increasingly dependent on software, targeted attacks against software are on the rise, causing harm to the infrastructure and disrupting business operations. There is a growing consensus that the software industry needs actively to address root causes of exploitable vulnerabilities. It also needs to implement methods that improve software resilience to attacks at their onset, thereby enhancing software trustworthiness.

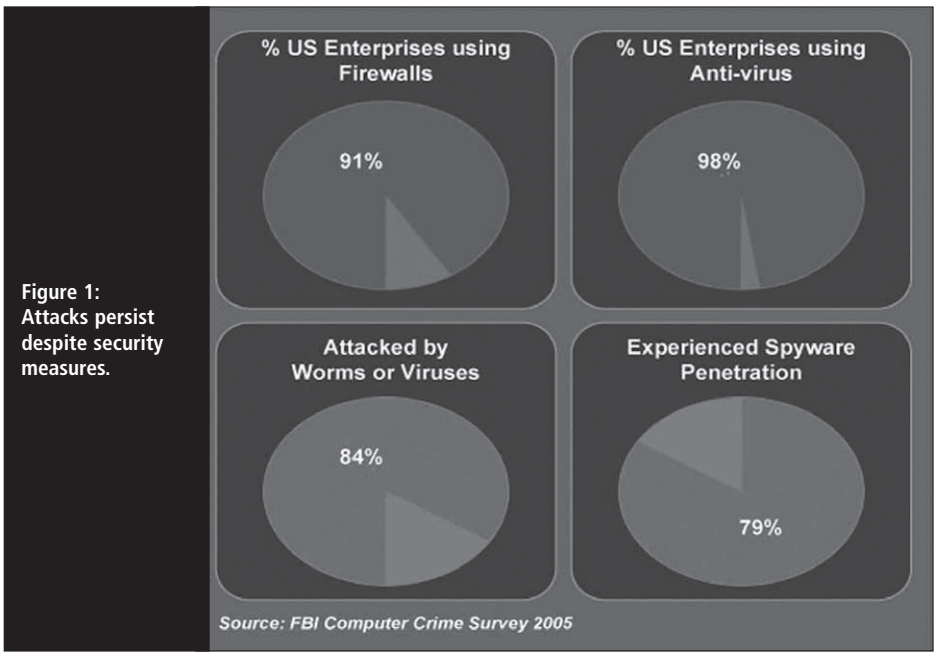

Figure 1: Attacks persist despite security measures.

Trustworthy software

Unfortunately, it is becoming increasingly difficult to establish or verify whether or not software is sufficiently trustworthy, due to a variety of factors:

- **Complexity**. The size and complexity of software systems and the components from which they are built are growing.
- **Interconnectivity**. Software implementations are becoming more interconnected via ever-larger networks, distributing control of maintenance and providing more opportunities for attacks.
- **Wireless**. Wireless technologies are being adopted more and more, potentially introducing rogue elements into a network and making the traditional designations of 'internal' networks even less relevant.
- **Net-centricity**. Migration to net-centric environments and service-oriented architectures is increasing and it is necessary to ensure that software components in these environments can interact securely without supervision.
- **Globalisation**. Modern software development and supply chain support are being more widely distributed worldwide, often with the focus on functionality for the lowest possible price, and without considering resistance to attack.
- **OSS**. There is expanding interest in and use of open source software (OSS). OSS is often developed quite differently from proprietary software and, therefore, ensuring the trustworthiness of contributing developers (i.e. the pedigree of software) becomes complicated.
- **Hybridisation**. Interconnections are increasing between different software components that were developed using differing methodologies and under varying constraints. Systems are built on hybrid networks of commercial-off-the-shelf (COTS) and custom software, open source and proprietary software, all using a variety of technologies.
- **Pace of evolution**. Software evolves rapidly as new technologies and supported feature sets introduce ever greater levels of complexity to an already complex system of networked and autonomous software components.
- **Lack of knowledge**. Most software developers generally do not know how to develop software that resists attack, a skill that is rarely taught in formal settings.

Achieving a breakthrough in software assurance requires the collaboration of multiple stakeholders, including organisations that produce major software components, system and software

integrators, acquisition organisations, certification agencies, insurance companies and regulators. Today's organisations rely on COTS/reuse/OSS products, developers from foreign and non-vetted domestic suppliers, integrated systems from mergers and acquisition activities, and more. Within this landscape, there is a growing need to either prolong the lifespan of existing legacy applications or improve existing applications to accommodate ever-changing market requirements and governmental regulations.

Most software development efforts depend on integration of off-the-shelf software; they do not solely develop custom code. Therefore mechanisms for managing assurance requirements of this code must exist. As these software systems grow in scope and complexity, they introduce greater variability in design at the level of individual software components. As a result, component designs may conflict with one another or obscure functionality through complicated interfaces, subsequently hindering comprehension and reducing the ability to effectively 'evolve' aging software components or respond to bugs. Ultimately, this may result in the fielding of a growing number of potentially unstable and vulnerable applications in operational environments.

Future challenges

The challenge for software providers and security professionals is to ultimately prevent the fielding of unstable or vulnerable applications whenever possible, to effectively manage security situations if they arise, and to reduce the risk to users and consumers that the software may pose in advance of deployment. The challenge for users and consumers is to understand the risks involved in deploying particular applications in their respective environments. The challenge for regulators, auditors and insurers is to communicate regulations effectively to both software providers and software consumers, effectively enforce regulations, and identify incidents where regulations and agreements are violated. The challenge for system integrators and acquirers is to assess the trustworthiness of components used to build the software.

To address these challenges there is a need for agreement on collective vision of the necessary component-level software properties that constitute secure software, system-level properties that increase the trustworthiness of a system running less trustworthy components, and properties for methods and tools that might be used to evaluate the trustworthiness of software and systems. Once firmly established, these properties are expected to serve as rudimentary design criteria. They will be used by software providers and system integrators as they engineer products that will meet the expectations for product-level assurance in advance of the acquisition and deployment of software. Software-intensive organisations and system integrators would be able to make arguments and verifiable claims to their customers that their software systems are sufficiently trustworthy, and customers would have methods to verify those arguments and claims through collected and presented evidences.

"Software assurance must be standardised whenever justifiable and when sufficient trustworthiness can be formally measured"

In addition, there needs to be a consensus on common terminology related to arguments, claims and evidences; on common, structured and repeatable techniques to exchange data and information related to arguments, claims and evidences; and on interpretation of data and information related to arguments, claims and evidences.

Software assurance must be standardised whenever justifiable and when sufficient trustworthiness can be formally measured. The measurement is achieved through a software assurance common framework and delivered in the form of a software assessment. The main benefits of these efforts are: secure and reliable software in its environment for the user's intended use, informed users and consumers demanding secure software assets, systems development of tools to help build more secure and reliable software, and better trained and educated software developers who utilise community-approved processes and tools to produce secure software.

Software assurance baseline model (SABM)

The common software assurance framework will be built on prior experiences and best practices and is expressed in terms of a software assurance baseline model (SABM). The SABM is intended to facilitate free exchange of information among software assurance community participants – notably between software consumers and providers. The community will collaborate to define the structure of the SABM, which will allow for the establishment of a common repository structure that can be used to universally represent information regarding vendor claims, arguments, and evidence related to software assets and their operational environments. More specifically, the SABM provides the ability to document:

- Existing software asset profiles, including associated data about the operational environments in which the software operates, and any relevant vendor-issued assurance claims.
- Industry-standardised reference models, including protocols and engineering requirements. These models explain intended use and requirements – enabling suppliers and acquirers to communicate and rigorously validate agreements – and the difference between the two.

The baseline model will also enable this information to be exchanged among different tools, which will enable vendors that specialise in certain languages, platforms or particular types of software systems to deliver solutions in conjunction with other vendors. The baseline model is not restricted to any particular implementation language, platform or specialised system and is designed to be flexible enough to correlate against all variations of software components at any level of functional abstraction.

The SABM-based models will likely contain large amounts of information, which will be difficult to manually manage for the essentially infinite variance among potential software component extensions. To overcome such a roadblock, models will support the capability to aggregate (summarise) information to different levels of abstraction. This requires the SABM to be scalable. In addition, the SABM will represent both primary and aggregate information. Primary information is assumed to be automatically extracted from the system itself through source/binary code analysis, reference models and/or manually entered by analysts or experts. Aggregate information is extrapolated from primary information.

Aside from scalability, the models should be both composable and actionable. This way the information can be accessed at progressively deeper levels that reflect the varying mission or business-level objectives that need to be achieved. Mechanisms for dealing with proprietary data, including those for releasing aggregate data while protecting primary data in some cases, will need to be identified. During development of this framework, a few specific and justifiable aggregate metrics will be identified and standardised, along with specific mechanisms to measure them. These specific aggregate standard metrics, which need to be useful for decision making, will demonstrate the utility of the overall framework.

As an example, a baseline model to mitigate software security risks should, at a minimum, include the following information:

Document software assets in their current configuration in their operational environments:

- Software assets knowledge – static analysis techniques will be used to automatically extract required information from software.
- Knowledge discovery – software artefacts related to structure, architecture, behaviour and data.
- Analysis performed on extracted knowledge – findings related to reliability, robustness and security. Detailed information on the breadth and depth of input validation, strength of defence-in-depth and defensive programming measures, security test coverage, or fuzz testing depth.
- Metrics and trends extracted from the system – design, security, complexity, sustainability, and risk metrics.

External information about software assets:

- System requirements: security, architecture, functional
- Environment (description) under this system runs, including the accepted boundaries of the system (e.g., resources and protocols required by the system, so that users can automatically enforce these boundaries knowing that the software will never exceed them in normal operation)
- Process (description) under which this system was developed

Documented industry standardised reference models relevant to this system and operational environments that they will run under:

- Security reference model
- Architectural reference model
- Quality reference model
- Functional requirements
- Environment where system will be deployed (description)

Software asset assessment results – compares two to define model of software trustworthiness based on the following information:

- Found faults that matter using a variety of robust tools and processes
- Risks (including probability and consequences) that these faults present
- Solutions, if any, that could be deployed to mitigate given risks
- Design robustness, validation of design and resiliency to corruption
- Process integrity and configuration control
- Understanding of environmental dependencies and relationships

The SABM represents a construct by which organisations can both evaluate and design more secure software. Designers of software for productivity enhancements must be held to design in terms of security as diligently as they do in terms of productivity. Otherwise data breaches such as those that happened at Hannaford will become more commonplace and costly to corporations and individuals.

About the author

Ed Ray is an information risk strategist for Getronics. He has extensive experience in threat and vulnerability analysis, vulnerability management, information infrastructure architecture and design, incident management, and forensics. He has written articles and technical papers on security and presented to audiences worldwide on computer and network security.

References

1. "CSI/FBI Computer Crime and Security Survey." 19 May 2008 <www.cpppe.umd.edu/Bookstore/Documents/2005CSISurvey.pdf>
2. "Payment Card Industry (PCI) Data Security Standard. " PCI Security Standards Council. 19 May 2008 <https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf>
3. Claburn, Thomas. "Thieves Steal 4.2 Million Credit And Debit Card Numbers From Supermarket Servers." InformationWeek. 19 May 2008. < www.informationweek.com/news/security/showArticle.jhtml?articleID=206904437>
4. Clark, Nicola. "Société Générale posts record loss on trading scandal, subprime exposure." International Herald Tribune. 19 May 2008. <www.iht.com/articles/2008/02/21/business/socgen.php>
5. "Six Sigma Articles." Motorola University. 19 May 2008. <www.motorola.com/content.jsp?globalObjectId=3070-5788>
6. "IT Governance Global Status Report 2008." IT Governance Institute. 19 May 2008. <www.itgi.org/>