Computer Security
Fundamental at Every Level

Ian Davis

The University of Texas at Austin
Department of Computer Science

CS370: Undergraduate Reading and Research, Spring 2016
Supervising Instructor: Alison N. Norman, PhD

Introduction

Modern society is powered by technology, and securing that technology is crucially important. As technology continues to increase in both prominence and user dependence, a lack of security features may cause disastrous results. Security must be implemented end to end, from user-facing facets such as the software that allows access to the Internet, to the systems and libraries that support user applications, and the hardware that makes everything possible. The technological layers at risk of exploitation, attack, and misuse can be divided into three main categories: 1) user-level products and services, 2) system control mechanisms, and 3) and physical hardware.

Complex software may utilize multiple categories to fulfill their functions, and so may be even more difficult to secure. For instance, the Internet utilizes all three categories: it has applications that require users to be able to input and receive data, requires system software to control communication, and hardware to create the physical transport of the data. The Internet that the modern world has come to know, admire, and depend was not initially designed to be what it is today. Originally, it was designed as "a group of mutually trusting users attached to a transparent network" [1] with users confined in closed networks, such as those in research institutes or governments. Today, the state of the Internet is far from that initial conception, as it is used for widespread communication over open networks containing unrelated users.

Many examples of technology have followed the same trend. Many of the other tools prevalent within society were never built to support the role they currently have. Unfortunately, security risks and other hazards are more likely when a product is used in a way that differs from the original design. This research explains implementations that suffered from unexpected vulnerabilities as well as the updates that resolved critical issues with prior implementations. Examples from the user, system control, and hardware categories will be discussed in this paper, as well as complex software systems and the vulnerabilities that plague them.

User-Level Hazards

As users browse the Internet, a constant flow of advertisements are displayed in their browsers, some of which are targeted, or tailored to specific subsets

of the population, and are chosen based on trends in a user's browsing habits. Targeted advertisements are a recent development in online advertising, and many users are unaware that the websites they visit track user requests in order to better advertise [8].

The websites track user activities by setting an identifying marker in the user's browser when a page is requested, commonly referred to as a *cookie*. The service uses the cookie data to track which pages are requested, and over time data can be compiled and subsequently analyzed, giving insight on a user's possible interests. Cookies were implemented without a specification or complete design, leading to a long line of vulnerabilities and insecurities. For example, cookies can be set to read-only mode, but many current implementations allow the the cookies to be modified anyway. Nowhere is the integrity of the data protected [9]. Additionally, cookies are frequently used to implement features such as persistent logins, where the website allows secure pages to be loaded without reauthentication by verifying cookie data. In that case, using vulnerabilities such as copying or overwriting secure cookies could be exploited to gain access to restricted data.

When a widely used constructs such as cookies are found to be insecure, it is common to assume that the next step is to fix the implementations and roll out an update. However, the situation is more complex than that. While this solution is simple in developed society, where a majority of devices support newer functionality, developing countries often depend on devices that cannot support newer standards. Thus, updating the core functionality of a service might strand users in these developing countries without that service. For instance, SHA-1 is a once-popular algorithm for certifying online identities, in which certificates are used to verify the identity of a website operator. SHA-1 was found to be breakable, resulting in insecure certificates. A new standard, SHA-2, has been developed. However, disabling support for the SHA-1 algorithm would cut off millions of users in developing countries from the Internet. Even with the risk of cutting off nearly 40 million people [2], many mainstream browsers such as Mozilla Firefox and Microsoft's browsers plan to disable SHA-1 support by late 2016 to ensure secure operation of their browsers.

System Control Protection and Exploitation

Shifting to the system control layer, security measures become more prominent. A recent system level security feature is the idea of provenance, or the history through time of an object. After a security incident, the object provenance could be reviewed, aiding in investigations. Since any changes to marked system data is recorded, a provenance-aware system would be able to detect abnormal changes in an application or set of data based on historical evidence using heuristics, allowing for efficient repair of system state by reversing the history written by the actions of the malicious entity. Keeping track of history for protected data such as system configurations, user data, and access records would add minimal overhead, with tests averaging around 2.7% of system load in the study released by the University of Florida and MIT [3]. In that implementation, SELinux, the set of security enforcement policies governing modern Linux systems, is used as a manager to the provenance system. The framework uses kernel modules likely already present in most secure systems, allowing for easy implementation into existing systems. Monitoring system functionality such as network connectivity uses common kernel modules such as IPsec and SSL.

Networks and Internet connectivity open a system to a large number of vulnerabilities and exploits. Peer to peer networks, such as the one present in the Bitcoin cryptocurrency network, use connections to others across the world to determine the status of the network and complete transactions. Unlike server-based infrastructures, there is not a single point of control or failure, resulting in a resilient and distributed network. Instead, the network is managed and operated by the *peer swarm*, users currently connected to the network. If a new peer connects to the network, it must receive the data it needs to function from other peers already connected to the network. Unfortunately, requiring data from unknown sources introduces possibilities for exploitation and attack. For example, until recently, the Bitcoin peer network implementation suffered from a vulnerability that made it possible to launch an "eclipse attack", defined as overwhelming a new peer with malformed information [4]. Under normal conditions, a node machine does not connect and communicate with a single peer – instead it uses the collection of peers to aggregate the needed information. In this exploit, an attacker overwhelms the connection tables of the victim, forcing the victim to use only the information provided by that attacker. This attack can lead the victim to complete computations or other tasks for the attacker.

Authentication and secure communication are important pieces of Internet communication. OpenSSL, a protocol that secures most of the modern Internet, was recently found to have a memory management vulnerabilty that made many of the world's secure systems vulnerable to attack. While the mistake was easily corrected once caught, the memory management exploit exemplified weaknesses in the development pipeline: the OpenSSL development team was not being paid for their work, despite the importance of the library. Since the exploit, companies have funded the team to continue development on the software. Such heightened support is common after a security event, but it would be better to support the fundamental libraries from the start to help avoid such incidents.

Hardware/Infrastructure

The hardware level represents the lowest level of a system. Designing secure hardware is a complicated process, as changing physical hardware requires months of planning, testing, and production. An example of a hardware security device is the Trusted Platform Module (TPM). A TPM is a hardware chip that is sectioned off from the rest of the hardware, and it requires strict permissions to change configuration settings. For example, through the use of a TPM, a system can boot and check the current state of the system against the TPM, aborting boot if an abnormality is found. TPMs are used for many different functions throughout the system, such as storing private keys for encryption.

As in the system control layer, the network is a major source of attacks. With the advent of Cloud infrastructure, where system hardware might be shared between multiple separate users, sectioning off network devices becomes critical. If multiple virtual machines are sharing a single interface, it would be possible for one VM to see or change another machine's packets. A solution developed by the Israel Institute of Technology [5] describes the creation of network hardware that is also virtualized and assigned individually. Securing virtual devices rather than subdividing access to a single shared physical device gives more fine-grained control over the device. Allocating a virtualized part of the device removes any access to a system without access to that subsystem.

Complex Attacks

A set of system security risks do not fall easily into either hardware or system level control software. An example of such a risk is a covert channel. Covert channels allow unwanted data transfer to occur without breaking the defined security policies. Covert channels come in many forms. A study published by ETH Zurich focused on a thermal covert channel, where differences in temperature are used to transfer information. The study utilized a system with dedicated cores of the processor assigned to certain processes. By observing the change in temperature of a CPU under various states of load, a malicious application can use trends of execution to receive data from the other cores. Data can be transferred at a rate of 12.5 bits per second, allowing three characters to be passed in two seconds. While that may seem slow, an amount equivalent to a few characters falling into the wrong hands in a highly secure system could still lead to unwanted results. With more powerful hardware than the entry level server platform that was used in testing, that rate could be much faster [6]. If the channel is not detected and shutdown in an efficient manner, large amounts of data could be transfered with little recourse.

Encryption is a process that obfuscates data so that it is not easily readable. The encryption and the subsequent decryption of data are computationally expensive. In modern devices, the hardware emits a detectable amount of electromagnetic radiation while the CPU is executing, and during heavy use such as when encryption or decryption algorithms are executing, the emission of radiation increases. A study by Tel Aviv University [7] was able to extract both partial and full Elliptic Curve Digital Signature Algorithm (ECDSA) cryptographic keys by interpreting the amount of radiation being emitted. These keys are the basis of the cryptographic algorithm. A compromised key renders the encryption useless, allowing easy decryption of the data. Due to the severity of the vulnerability, the device vendors have released both software and hardware updates to help patch the devices. The vulnerability was first found on Apple's iOS devices, but was later successful on other mobile devices such as devices running the Android operating system.

Conclusion

The security vulnerabilities presented in this paper span in subject from web exploits on persistent data in the browser to the radiation emission of a device during execution. These examples highlight that security is an end-to-end concern; securing a single portion of the spectrum is not sufficient. Despite that, the development cycle continues to consider security as an afterthought, only gaining focus after an event occurs. That mindset is not working now, nor will now work in the future. As complexity of technology continues to increase, security flaws will become more prominent and their effects will carry a larger impact. Security needs to be a fundamental part of the development process to reduce the occurrence and effects of security incidents, which will in turn lead to more efficient and reliable systems.

References

[1] Benjamin M. Compaine and Shane M. Greenstein. *Communications Policy in Transition: The Internet and Beyond*. MIT, 2001.

[2] Lucian Constantin. Sha-1 cutoff could block millions of users from encrypted websites. CSO Online. Updated 12/10/2015.

[3] Adam Bates, Dave Tian, and Thomas Moyer. Trustworthy whole-system provenance for the linux kernel. *24th USENIX Security Symposium*, 2015.

[4] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. *24th USENIX Security Symposium*, 2015.

[5] Igor Smolyar, Muli Ben-Yehuda, and Dan Tsafrir. Securing self-virtualizing ethernet devices. *24th USENIX Security Symposium*, 2015.

[6] Ramya Jayaram Masti, Devendra Rai, Aanjhan Ranganathan, Christian Muller, Lothar Thiele, and Srdjan Capkun. Thermal covert channels on multi-core platforms. *24th USENIX Security Symposium*, 2015.

[7] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. Ecdsa key extraction from mobile devices via nonintrusive physical side channels. *Tel Aviv University*, 2016.

[8] Farah Chanchary and Sonia Chiasson. User perceptions of sharing, advertising, and tracking. *Symposium On Usable Privacy and Security 2015*, 2015.

[9] Xiaofeng Zheng, Jian Jiang, Jinjin Liang, Haixin Duan, Shuo Chen, Tao Wan, and Nicholas Weaver. Cookies lack integrity: Real-world implications. *24th USENIX Security Symposium*, 2015.