Security at Different Layers of Abstractions

Application, Operating Systems, and Hardware

Threats Across Computational Abstractions

In the field of security, it is important to take a threat-centric view of the world. Threats help us understand *who* the malicious actors are, *what* these actors want, and *how* they achieve their goals. Computer systems are then evaluated in light of the perceived threats to understand the best defensive measures. As a result of recent events in the news such as the Snowden revelations [Greenwald2014] and the Equation Group [Kaspersky2015], we now have a much richer view of the threats posed by the most advanced adversaries. Meanwhile, researchers have extensively studied other adversaries over the past two decades, which helps round out or understanding of the threat space.

A threat actor will typically use the easiest path the achieve his or her goals. In some cases this requires manipulating another person, a practice known at social engineering. On the technical side, a threat actor may choose to attack the victim's hardware, operating system, or applications. Each option has its own trade-offs covering a broad spectrum of sophistication, cost, likelihood of detection, feasibility, and more.

Often application-level attacks are the easiest target for attackers. Given that applications cover a broad space across servers, desktops, laptops, and mobile devices, there is a broad range of application-level attacks. Phishing attacks [Garera2007] involve convincing a victim to visit a malicious web page that exploits a vulnerability in the user's web browser or email software. Another example is Heartbleed [Mehta2014], a recent attack on the OpenSSL cryptographic software that mostly affected web servers and could result in disclosing private information.

Operating system attacks can be more difficult to execute because modern operating systems have reasonably good default security practices. Even so, this remains an attractive position for attackers because software running at the operating system's privilege level can view data from anything on the system including all applications, files, and memory state. Typically these attacks install rootkits: malicious software designed to provide operating system level access to an attacker while hiding from the system's users and applications. Since it can be difficult to compromise operating systems remotely, rootkits are usually installed after a successful application-level attack or by compromising the supply chain.

Hardware attacks are the most advanced. These can include replacing hardware with a malicious version that appears the same to users, but allows an attacker to access the system. This can also include firmware attacks that can change programmable portions of hardware to an attacker's benefit. Physical hardware replacement requires a supply chain attack or an on-site operative. However, malicious firmware can be installed from a rootkit as a way to improve stealth and retain access across fresh installations of operating systems.

Plentiful Targets

Upon hearing "hardware, operating systems, and applications", many people will constrain their thinking to the obvious computers in their lives. Perhaps this includes a laptop and smart phone. Yet computers are everywhere. Computers called switches and routers manage network traffic on the Internet and mobile phone networks. Computers are in our automobiles [Miller2015], medical devices [USFDA2015], smart watches, TVs, and refrigerators. Computers are even inside some light bulbs [McMillan2014]. Each of these represents a different attack

option for malicious actors. And while some are useful targets by themselves, others can serve as a stepping stone to help attackers reach their ultimate goal(s).

The challenge for security professionals is to understand the broad scope of threats while still building systems that provide meaningful security. For example, if one is concerned with network traffic monitoring on the Internet, then a reasonable countermeasure is to encrypt that traffic. Similarly, if one is worried about attackers using guessed or stolen passwords to access an online service, then a reasonable countermeasure is to use two-factor authentication (e.g., send a code to the user's phone that must be provided along with the password for access). But when everything is interconnected, and the threat landscape is enormous, it is not always obvious what protections are most important to implement.

In addition to rich interconnections, modern computing systems from companies such as Facebook, Google, and Netflix operate at an extremely large scale with some managing millions of computers [Ballmer2013]. This kind of scale shifts the security challenges considerably. Securing each piece of hardware and each operating system requires dedicated teams with this specific focus. And these teams must work closely to ensure that security problems do not arise on the boundaries. Even finding the application data that requires protection can be challenging. And retrofitting security into an existing large ecosystem requires that security doesn't break reliability or performance.

Case Study: Protecting Network Traffic in the Cloud

When using a web browser, most people know to look for a lock icon indicating a secure connection before making a credit card transaction or some other sensitive operation. This security is enabled by a network protocol called transport layer security (TLS). It works because

there is a pre-established trust relationship between your computer and a third-party that has verified the authenticity of the web site you are connecting with.

In addition to protecting the network traffic between a web browser and a web site, TLS is often used to protect the internal network traffic within a cloud or data center that supports the web site's operation. In these cases the technology is the same, but the deployment strategies are often different. These differences reflect both the scale and the threats across many abstraction layers. The use of TLS in a cloud or data center mitigates the threat of malicious network switches eavesdropping on or altering network traffic. It also mitigates potential threats in the software virtualization layers used to enable cloud computing.

Deploying TLS at scale for a cloud application presents many challenges. Primary to these is the need to create trust relationships -- like those used to make credit card transactions with a web browser -- that are reliable and maintainable. This kind of trust relationship is called a public key infrastructure (PKI). Unfortunately, in practice, it can be very difficult to deploy and maintain a PKI. One of the challenges is that critical security credentials may sometimes be lost or compromised. Traditionally this problem has been addressed using revocation lists, but these lists are hard to maintain and do not scale well.

The use of short-lived credentials [Topalovic2012] in lieu of revocation lists has been growing in popularity. The idea is that a compromised credential is less valuable because it will only work for a very limited timespan. Using short-lived credentials can increase software complexity due to the automation required to create and deploy the updated credentials [Clark2014]. However, it also raises the bar for an attacker and results in a more maintainable PKI.

Establishing a PKI and using TLS to protect network traffic mitigates certain classes of attacks, as described above. At the same time, if a malicious actor can access other levels in our computing ecosystem then he or she could compromise the very data that TLS is designed to protect. Security practitioners must estimate if a particular threat actor is sufficiently capable and motivated to break the assumptions and compromise the system. This game is why security is often considered to be both an art and a science.

Challenges and Future Directions

In practice, hardware-level security is a domain that only nation state actors can afford to engage in both offensively and defensively. For many, the best strategy is to assume that the hardware is secure and leverage some of its security features. Building from this foundation, there are many opportunities to improve security at the operating system and application layers.

Services running in the cloud today generally need to take a leap of faith and assume that the underlying cloud infrastructure is secure. Often this is a very reasonable assumption because the major cloud providers have thus far demonstrated that they are better at infrastructure security than all but the largest data center operators. However, rather than trusting by faith, **it would be much nicer to trust and then verify that cloud providers are in fact secure**.

Another major challenge in security today is simply that it is too hard. We have created security primitives that are so flexible and general purpose that the average developer can't possibly be expected to use them correctly. We must reduce the complexity at all levels. This includes **reducing software engineering complexity**, making security **more science and less art**, and providing **secure building blocks for creating complex software**.

These technical challenges are huge, but critically important. The only way we will be able to solve them is with many talented computer security professionals. Unfortunately, the community has a **massive talent shortage**. As with the other challenges above, solving this will require close partnerships between academia and industry.

References (in order of citation)

[Greenwald2014] Glenn Greenwald. No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State. Metropolitan Books, May 2014.

[Kaspersky2015] Kaspersky Lab. Equation Group: Questions And Answers (Version 1.5). February 2015: https://securelist.com/files/2015/02/Equation_group_questions_and_answers.pdf

[Garera2007] Sujata Garera, Niels Provos, Monica Chew, Aviel D. Rubin. A framework for detection and measurement of phishing attacks. *Proceedings of the ACM workshop on Recurring malcode (WORM)*, November 2007.

[Mehta2014] Neel Mehta (Google) and Codenomicon. The Heartbleed Bug. April 2014: http://heartbleed.com

[Miller2015] Charlie Miller and Chris Valasek. *Remote Exploitation of an Unaltered Passenger Vehicle*. Black Hat USA. August 2015.

[USFDA2015] U.S. Food and Drug Administration. *Symbiq Infusion System by Hospira: FDA Safety Communication - Cybersecurity Vulnerabilities*. July 2015: http://www.fda.gov/Safety/MedWatch/SafetyInformation/SafetyAlertsforHumanMedicalProducts/ucm456832.htm

[McMillan2014] Paul McMillan. *Attacking the Internet of Things Using Time*. DEF CON 22. August 2014: https://www.youtube.com/watch?v=uzGXxWuDwxc

[Ballmer2013] Steve Ballmer. Keynote Speech. Microsoft Worldwide Partner Conference. July 2013: http://news.microsoft.com/2013/07/08/steve-ballmer-worldwide-partner-conference-2013-keynote/

[Topalovic2012] Emin Topalovic, Brennan Saeta, Lin-Shung Huang, Collin Jackson and Dan Boneh. Towards Short-Lived Certificates. *Proceedings of the Web 2.0 Security & Privacy Workshop (W2SP)*, May 2012.

[Clark2014] Robert Clark. SSL Everywhere with Ephemeral PKI. OpenStack Summit. November 2014: https://www.openstack.org/summit/openstack-paris-summit-2014/session-videos/presentation/ssl-everywhere-with-ephemeral-pki