It's Time for Application-Centric Security

Position paper by Irdeto

Jan 20, 2014

Abstract

Existing security models are inadequate to address emerging web threat models, such as massive surveillance and ever burgeoning malware, because they fail to protect data and code on exposed consumer terminal devices of the network, which are vulnerable to attacks directly accessing their hardware and software. We present an *application-centric* approach based on dynamically renewable homomorphic transformations of executable code and data on such terminal devices, to provide greater security control to consumers and application providers, with a much better fit to user-centric business models than the existing service-centric security frameworks widely deployed today.

Introduction

Web computing is entering an exciting stage with Open Web Platform while open standards such as HTML5, SVG, CSS, JavaScript, video (via CDMs) and others are advancing together so that programmes that once worked only a native environment on desktop, tablets or phones can now work from within a browser itself.

As tablets are replacing laptops, and smartphones are replacing wired lines and fixed function devices, mobile apps now not only pervade consumers' personal life but also represent core productivity tools of the modern workforce: at the office, during travel, in the field, at the factory, and at home. Open Web standards also allow Web Apps to connect computing activities between client devices and cloud-based Web services. Therefore, using Apps (native, web or hybrid), people can easily access anything from anywhere at any time by using available devices at their convenience.

However, threats to web and mobile spaces are also rapidly evolving from typically unsophisticated attackers, through organized crime, to far more serious threats posed by some nation states (for example, massive surveillance by national intelligence agencies), terrorist organizations, and sophisticated hacktivists. Almost everything including emails and personal data can become attack targets for the less scrupulous arms of national governments, financial service organizations, retailer and on-line businesses, IP rich organizations, critical infrastructure providers, social groups and networks, who can target millions of people at once. The pervasive monitoring attack is just such a high profile case within the new attack landscape.

Threats are becoming exponentially more sophisticated and advanced. The threats often seen today are agile and dynamic, more focused for very specific goals and a narrow class of organizations and groups if necessary, more intelligent and smarter, and employ a wide range of social engineering techniques and technical exploits to gain a foothold within victim browsers and devices while avoiding detection. Some security threats and security breaches are so serious that an appropriate response requires an update to a widely used interface and/or protocol. As this implies a very long transition process, the attack life cycle is extremely long.

Traditional Security Is Seriously Broken

Obviously, new security challenges and problems reported strongly indicate that traditional security is not designed to counter today's advanced threats. Existing security measures are often:

Signature-Based: By looking for known "bad" data based upon previous identical attacks, the long-standing blacklisting approach is losing the battle against new malware. Right now, we are seeing about more about 150,000 new pieces of malware every day, and this will get much worse in the future. Obviously, signature-based defenses are grossly insufficient to defend against today's threats.

Perimeter Oriented: Perimeter oriented approaches concentrate on preventing or detecting threats entering networks of an organization, but perimeters are very porous these days. Anything with an IP address can be a launch pad for attackers. With the range of new use cases that need to be supported, from BYOD to fixed function devices, accessing legacy web apps to new cloud-based app development and services, IT is left today with the challenge of working with a varied set of non-integrated tools to accomplish a wide variety of tasks for differing constituents and business requirements, while striving to achieve regulatory compliance and security at the same time. Deadline pressure often yields working functionality with no real security, because non-functionality is easily visible whereas insecurity may take deep analysis to find.

Sandboxing Security: Sandboxing is one of oldest security techniques and has been widely using to prevent traditional man-in-the-middle attacks. It leaves us with a false sense of security: many threats cannot be addressed by sandboxing security approach. For example, it does absolutely nothing to prevent massive surveillance, because it seeks to protect the host against hostile software, not the software and its data against the potentially hostile host. Plainly, we need a more comprehensive approach.

Device-Centric Security: Over the coming few years, managed devices will increasingly employ secure execution environments as common practice. But there will still be many unmanaged devices in the market, in particular, m2m devices used in the Internet of Things (IOT) with limited hardware security. Moreover, mobile device management (MDM) controls depend on native capabilities, so policies and enforcement inevitably vary across devices. The policies we hope to enforce are inactive most of the time. To get around this limitation, some MDM vendors use location-based services to track the device so that the application remains active in the background. Such implementations must be very secure indeed, or they risk being co-opted by attackers to compromise the very devices they are supposed to protect.

Platform Based Security: Platform based security seeks to prevent data leakage such as personal emails, social media data, or business data in cloud storage. It relies on OS native security or security support on the system level, rather than in the application itself. However, devices can easily be lost or stolen, exposing them to attacks which may well break such platform based security. Moreover, security implementations vary widely across platforms. The commodity security offered by SoC manufactures will continue to be behind software and hardware hackers.

Fixed Security: A typical approach to security is to assume that the initial design will remain secure over time: it treats security as a fixed target, assuming that innovative attacks will not arise after deployment. Most security designs and implementations follow such a static deployment model, especially for hardware-based security. Once cracked, hardware security can't be recovered quickly or cheaply. In reality, anything, including clever hardware, can be hacked given enough time and effort. Therefore, new dynamic security approaches treat security as evolving and assume that security must be continually renewed, whether as part of ongoing policy or reactively.

Compliance Driven Security: Compliance meets the requirements of auditors, or specific government mandates, rather than addressing the biggest current threats. The danger is that we may mistake compliance with security standards for actual security. They are two very different things, and some of them may deal with past threats.

As we move to the mobility era, the constant among the hundreds of computing devices, with which a user may interact in a given day, is the consumer himself/herself. Therefore, many businesses are making necessary changes to their services and solutions to move to user-centric business models.

Underlying technology, however, is still service-centric, and users of such services have little control and limited rights regarding security. Without fundamental technological adjustment, this revolution may founder due to the mismatch between what is needed functionally and what security requires. Rights between services creators/owners/providers and consumers *cannot* be balanced to suit all users. Security lacks interoperability and comparability, and we are unlikely to get competing providers to co-operate to the extent needed to solve this. Therefore, we need a solution which depends less on the rival device and platform providers.

Application-Centric Security and Protection

Tim Berners-Lee, creator of WWW and director of W3C, has recently written the following section:

"*The W3C community is currently exploring Web technology that will strike a balance between the rights of creators and the rights of consumers. In this space in particular, W3C seeks to lower the overall proprietary footprint and increase overall interoperability, currently lacking in this area.*"

"So we put the user first, but different users have different preferences. Putting the user first doesn't help us to satisfy users' possibly incompatible wants: some Web users like to watch big-budget movies at home, some Web users like to experiment with code. The best solution will be one that satisfies all of them, and we're still looking for that. If we can't find that, we're looking for the solutions that do least harm to these and other expressed wants from users, authors, implementers, and others in the ecosystem."

In a rapidly changing world where information relies on application software for its creation, storage, distribution, consumption and processing, we can't expect any fixed protection to provide ongoing security. This is true even when hardware is employed to harden platforms against attacks. Hence, it's becoming a fundamental requirement that we protect digital information by continuously upgrading the protections in its associated software.

Increasingly, companies rely on security technologies to protect their business model and assets, while users expect their assets to remain protected. Application security must be dynamically developed, deployed, maintained, and updated. We have to make security agile and rapidly deployable, and to employ dynamically and flexibly renewable protection technologies. Software defenses can go far toward meeting these requirements. Software defenses must achieve two objectives: resist outcomes that provide the information or capabilities the attacker desires, and resist the disabling of protections. That is, they must obscure both data and computation, and must resist tampering by making its effects as useless to the attacker as possible. (For example, attacks designed to cause a specific change in billing procedure fails if the tampered version of the application simply fails to function at all.) In the software world, this can be achieved by compilation tools and utility libraries that operate on obscured data to resist information capture, with subtle interdependencies to produce chaotic results which are useless to the attacker.

We favour a new security perspective based on an application-centric protection approach and solutions that can inter-work with many existing security features, by addressing the vulnerabilities which existing security approaches fail to protect.

Application-centric security has many advantages. The security of an web app is not only supported by security features from the environment, in which the app is running, but also is built into its code by applying application-centric protection to the web app software. Such security becomes part of execution behaviour of the protected app. E.g., it can apply app-level data protection so that: a) different apps, or different user using the same app, can have different schemes to secure their own data at rest and in the motion; b) different rights and controls to access app-level data can be managed by a web service provider and/or app users; c) we can provide cloud-based access between mobile devices and corporate resources, while securing app-level data and document without losing transparent network access; d) we can extend collaboration to selected partners and customers while maintaining information confidentiality.

For years, security experts have urged layers of security controls across endpoints, networks, and the cloud, supported by multi-vector threat intelligence. Application protection and security control is fundamental to success because a network is useless without communicating to its end-points such as commodity user devices. A protected web app can employ built-in security intelligence to change security strategy dynamically, and can apply different protection approaches and methods at dynamic software-renewal time in response to the monitoring and analysis of attacks/threats over time. Application-centric protection permits a balanced security framework integrating current security with new protections which current security methods fail to address. A software-protected application is a user-centric point at which to glue together user-selectable end-point data- and behavior-protection, protection against potentially hostile hosts or device loss or theft, and existing security techniques, to provide a comprehensive set of protections.

White-Box Security and Protection

Most of the analysis in IT systems security assumes that attackers are remote and attacks of interest are network-based, or the attacker is a user with limited privileges; thus the attacks of interest center on escalation of privilege and its follow-on activities. In traditional system security, getting a root access is sufficient to accomplish further steps in the attack and gain benefits. In this traditional security domain, the system under attack is a "Black Box", which means the attacker is not able to see into the system, but can only try to manipulate it externally. Examples of such attacks are SQL injection, Cross-Site Scripting, Malware, etc.

Such areas of computer security are characterized as man-in-the-middle attacks and addressed by well-known technologies such as firewalls, cryptography, intrusion detection, authentication (such as VPN, SSL), secure coding, vulnerability scanners, browser sandboxes, malware detection, anti-virus programs, trusted computing and the like. This 'classic' security domain has been thoroughly described.

In contrast to the traditional Black-Box security, in a different and more challenging domain, the attackers can also be the system owners, and therefore, they already have root access (or equivalent) to the system.

This security context is also sometimes called the White-Box attack context. A white-box system can be inspected, disassembled, restarted, reverse-engineered, debugged, or modified, in ways limited only by the intelligence of the attacker and the inherent security of whatever software is inside the white box. Attacks within such a white-box context are characterized as man-at-the-end attacks.

Suppose that application software does not contain any black-box vulnerability. It may well still be vulnerable to white-box attacks. Without applying necessary protections to it, many obvious and direct attack points exist. E.g., the attacker can obtain valuable secrets, such as cryptographic keys, by inspecting or lifting data from memory and registers, modifying control flow by jamming branches, modifying program logic by tampering with individual instruction code and data values, or stealing intellectual property by lifting or reverse-engineering the code. In general, every part of an application is in clear form and exposed to white-box attack methods. How can software be made systematically resistant to such attacks?

Protecting software in a white-box context is a hard problem. Indeed, some observers would argue that it is an inherently unsolvable problem. From an academic point of view, this is an understandable position. However, in real-world applications, such as mass-market commercial software and digital media, this is not a profitable business position. Instead, the industry has an urgent need for software protection technology.

Irdeto cloaking technology is recognized as an advanced and widely used method of protecting software, and has been adopted and deployed in more than two billion devices worldwide. It addresses the truly daunting problems of maintaining software security in untrusted environments. The importance of such security technology is greater than ever because software is increasingly being deployed in untrusted environments throughout the digital world, from home networks to consumer devices, to the public Internet, to the cloud, and to the Internet of Things, where traditional computer and network security are inadequate. (We hasten to add that other companies and organizations have related technologies for protecting both executable code and data in exposed environments: this is an active area of research in both industry and academia. But Irdeto has considerable experience and success in this area to draw upon.)

Irdeto has developed technology to make software applications "White box" attack resistant, and applies such protection mechanisms on a wide range of software on both native and non-native execution environments. Extended variants of such tools will also have the ability to protect web apps statically and dynamically, which is an important element in the HTML5 standard where it provides the ability to include functional logic in web contents. Irdeto's software protection technology can co-work with other security features from underlying systems and platform including hardware security, and can protect software layers from low level to high level within applications.

Data Security and Homomorphic Transformations

According to the 2012 Verizon Data Breach Investigations report, 99 percent of breaches led to data compromise within "days" or less, whereas 85 percent of breaches took "weeks" or more to discover. This presents a significant challenge to security technology and response teams as it grants attackers extended periods of time within a victim's environment. More "time" spent for deploying a countermeasure leads to more stolen data and more digital damage.

In fact, data frequently exists in various classes of storage at runtime, for example, in registers, on the stack, on the heap, on disk, or other forms of secondary storage. When data is processed by a program, the layout of data in memory is well known and once the attacker finds a data asset, that asset is no longer confidential. If data is stored using conventional formats, the attacker will know how to discern the object's true value, thus defeating the security objective of the application (keeping the data secret).

Many tools are widely available for inspecting, debugging, disassembling and tracing binaries and data within the binaries or processed by the binaries. Attacks can be automated in a script thus enabling unskilled attackers to execute them successfully. Data assets within the program as well as those being computed by the program are subject to a confidentiality requirement. Valuable data assets in memory are extremely vulnerable to perusal.

Irdeto introduced homomorphic data transformation technology as a key solution to address the data and computation protection problems. By applying mathematical transformations to protect original data flow, constants, operations and original variables and computing values by transforming them homomorphically, we can achieve the following properties:

- The computation of the original data flow is functionally preserved and program operations are applied on the protected data without decoding it to easily visible form.
- A large increase in program data interdependency makes reverse engineering and tampering a very much more complex undertaking.

Transforming a variable in a program converts it from one representation to another, making it harder for an attacker to analyze. Any operation on that variable must now be performed on the new representation. In the other words, protected data can be operated on within protected space without exposing original values. Mappings between original and transformed code are many-to-many. It has been proven that reversing the transformed code to original code is an NP-complete fragment recognition problem.

The instances of natural data types are replaced with instances of equivalent securely encoded data types. A secure encoded data type has three properties which makes it less vulnerable to perusal even if an attacker successfully locates it. These properties are:

- It is a homomorphism of a natural data type and has the same semantic behavior of the natural type, but is implemented differently.
- Two instances of encoded data types can be combined in a computation even if they have different representations, without either value reverting to the natural data type.
- The implementation is *ambiguous*, in the sense that it is difficult to map any particular transformed computation to a specific natural one, since many seemingly equally valid alternatives exist.

In principle, we transform the computation by transforming its data flow (i.e., both its stored and operating data and the computations on the data). Our homomorphic transformation protections continue to increase in scope as its development continues, expanding our application-centric based data security capabilities. We believe that such solutions are of great advantage for addressing pervasive monitoring and other current web insecurities.

Conclusion

We have explained why we believe existing security models cannot address emerging web threats (massive surveillance, ever growing malware), because they fail to protect the network periphery, where consumer devices are vulnerable to direct attacks on their software and/or hardware. The *application-centric* approach we espouse, based on dynamically renewable homomorphic transformations of executable code and data on such terminal devices, can provide greater security control to both consumers and application providers, with a much more natural fit to user-centric business models than today's existing service-centric security frameworks.

References

[1] S. Chow, P. Eisen, H. Johnson and P.C. van Oorschot: White-box Cryptography and an AES Implementation, Selected Areas in Cryptography (SAC 2002), LNCS 2595.

[2] S. Chow, P. Eisen, H. Johnson, and P. van Oorschot: A White-Box DES Implementation for DRM Applications. In Proceedings of 2nd ACM Workshop on Digital Right Management, Nov 18, 2002.

[3] S. Chow, Y. Gu, H. Johnson, V. Zakharov: An approach to the obfuscation of control-flow of sequential computer programs. In G. Davida and Y. Frankel, Information Security, ISC 2001, vol 2200 of Lectures Notes in Computer Science, Springer Verlag, 2001. 68.

[4] Y. Zhou, A. Main, Y. Gu, H. Johnson: Information Hiding in Software with Mixed Boolean-Arithmetic Transforms, 8th International Workshop on Information Security Applications (WISA 2007), pp. 61-75, Springer LNCS 4867, 2008

[5] Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection, C. Collberg et al, Addison-Wesley, 2009.

[6] C. Liem, Y. Gu, H. Johnson: A Compiler-Based Infrastructure for Software-Protection, Programming Languages and Analysis for Security (PLAS'08), Tuscon, AZ, June, 2009, pp. 33-44.

[7] Y. Gu, B. Wyseur, B. Preneel: Software-Based Protection Is Moving to the Mainstream, IEEE Software, March, 2011, pp. 56-59.

[8] C. Collberg, J. Davidson, R. Giacobazzi, Y. Gu, A. Herzberg, F. Wang: Toward Digital Asset Protection, IEEE Intelligent Systems, Nov. 2011, pp. 8-13.