

RADICALLY SIMPLIFYING CYBERSECURITY

Dan Kruger
CEO, Absio Corporation

SYNOPSIS

Cybersecurity professionals are publicly acknowledging^{1,2} that the traditional approaches to securing information fail because the threat environment has become impossibly complex. This paper focuses on a comprehensive approach to securing information that can radically simplify cybersecurity.

THE DEFINITION OF THE PROBLEM IS THE PROBLEM

Cybersecurity has become unmanageably complex because the definitions of security do not match the operational environment—and they haven't for a long time. These approaches to securing information made a lot of sense in the mainframe era:

- Perimeters: Information is protected when you control access to it by establishing a perimeter. Defining the perimeter was simple—control physical access to the building and the terminals. Early in my career I made the information systems director of a major oil company nearly apoplectic by introducing personal computers to his company “because once you let them in, we'll never have control of our data again.” He was right.
- Processing Capacity: For information to be readily useful it needs to be unprotected—stored “in the clear.” Protecting information takes too much processor capacity and makes it harder to use.
- Training: You keep users from leaking information by training them to understand and follow the organization's security policies.

Perimeters

These assumptions are no longer valid, starting with the notion of simply definable perimeters. It's worthwhile to look at just how hard it is to merely define the perimeter—much less defend it. Start with a mainframe and terminals in a building you control. Then add PCs, each with their own processors, local storage, i/o ports and local applications; each PC has a perimeter of its own. Then add portable computers designed to go outside the physical perimeter. Then add local area networking; perimeter elements you must defend now include physical network wires because people can tap into them to eavesdrop. Then add wide area networks; now your perimeter elements include somebody else's wires and switches.

These definitions describe distributed computing in the mid '80s—the last time anybody was able to do a marginally credible job of defining a perimeter—and this is where the notions of cybersecurity have been stuck. People in the cybersecurity field constantly talk about protecting networks as if that were synonymous with protecting information. Network protection, even if perfect, is a partial security solution at best.

Now move to the mid '90s and add the commercialized Internet. Then add ubiquitous email and the World Wide Web, both enabling the unconstrained distribution of information by unauthenticated users. Next add high-density portable drives and make some of them smaller than a postage stamp. Then add radios in multiple flavors—cellular, Wi-Fi and Bluetooth, with more on the way. Include applications, too—peer-to-peer networking, streaming media and social networking.

Can any organization actually define its perimeter? If it were possible, would you be able to defend it? Let's look at the complexity of defending a small workgroup.

Every digital device, operating system (OS), application, transmission path, file and human being has multiple attack *vectors* (paths to the target) and attack *surfaces* (that which can be attacked). When devices are booted, users log in, launch applications, open emails, browse websites, manipulate files or transmit information, those actions open vectors and expose surfaces, creating attack opportunities.

Let's do some rough math: 10 users x 3 devices each x 20 applications each x 5 attack vectors x 100 interactions daily. That's 3,000 perimeter elements to defend and 300,000 threat opportunities a day. (Argue the math if you want—it's actually worse than the example.) If that perimeter could be precisely defined, who would have the time, money and expertise to close all of the holes and keep them closed?

In addition to sheer numbers, threats are unpredictable and dynamically complex. Addressing one can produce unintended consequences in others, and nifty new technologies compound the problem. It's only a matter of time before we hear something like this: "A major breach was traced back to an exploit that used the smart refrigerator interface in the CIO's home network to infect his tablet, which then invaded his company's network."

Even if you could define all of the perimeters, they are too complex to defend. This is not a call to abandon the perimeter; the better the perimeter defense, the more sophisticated the attacker needs to be. But it is clear that the perimeter is only the outer layer of defense.^{3,4,5} Defense in depth is required. For additional information, see Absio's white paper on "[Why Perimeter Security is Inadequate](#)".

Processing Capacity

If information protection is properly engineered, there is more than enough processor capacity to protect information everywhere it's at rest or in motion—if you include the processor on edge devices. It's critical to include edge device capacity in any approach to cybersecurity because increasingly sophisticated edge devices:

- Are where most information lives
- Have i/o ports for exporting information
- Are often portable and easily stolen or captured
- Are what users are using and will continue to use

That brings us to the cloud. The drive toward clouds and the drive for more powerful mobile devices are, in many ways, at odds with each other. Cloud computing assumes that processing is done in the cloud and that edge devices host what are essentially prettified dumb terminal emulations that always have a good connection. We have recreated the mainframe model in an attempt to define the perimeter.

The conceptual security advantages of clouds are obvious: They have a definable perimeter. But the perimeter itself raises a fundamental security problem.⁶

If users can take the product of cloud computing and freely distribute it, then cloud security stops at the perimeter of the cloud (more precisely, at the datacenter's interface to the Internet). Information security

solutions must comprehend information security from cloud to cloud, cloud to edge device, and edge device to edge device. Information security must *persist*.

Disable-your-hardware solutions for securing cloud connections have not and are not going to be fully implemented. People are not going to disable their devices' i/o ports. Users will store and share information on their edge devices regardless of security policies.

People and Training

The third assumption was always more hope than reality. Do we expect users to follow security policies that make their jobs harder—especially when they know the policies make little difference? Do we expect that good training will stop all social engineering? Thwart suborned or malicious insiders? Stop outsiders who masquerade as insiders? Do we expect that sufficient training will ensure that every user in the distribution chain will make the correct security decisions about that piece of information?⁷

The perimeter has been breached. The network has been invaded. We cannot stop it. What can we do?

Change the game.

INFORMATION THAT PROTECTS ITSELF

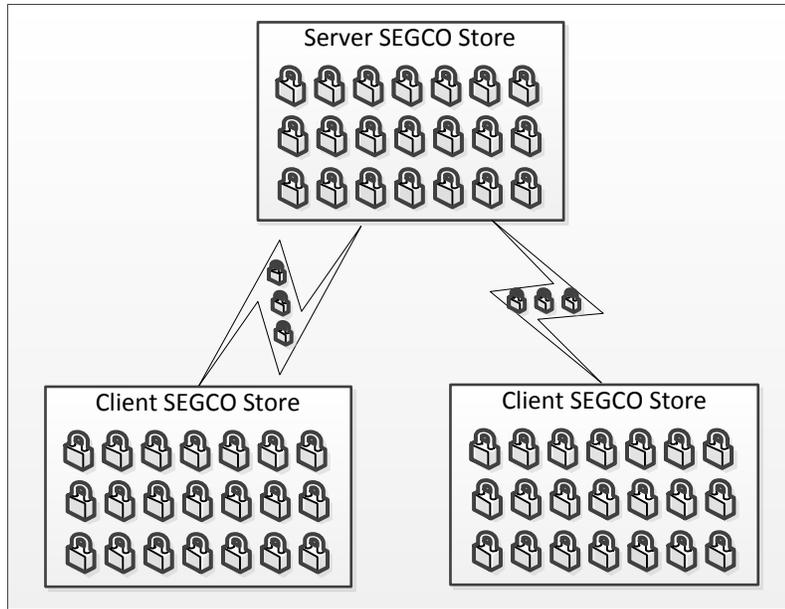
Cybersecurity is radically simplified if we move primary information security **into the information itself**. With today's processors, storage density and the right engineering, it is possible to make every piece of information a "hard target" that protects itself and is still easy to use. The focus of cybersecurity can shift from the utterly impossible (defending the undefendable perimeter and persuading the unpersuadable user) to the merely difficult (moving information out of the clear).

If information can protect itself, the assumptions behind cybersecurity are very different:

Current Assumptions	New Assumptions
Information is protected when you control access to it by establishing a perimeter.	Information is protected when it is bound to persistent distribution controls and is never in the clear except when actually being used.
Information needs to be stored in the clear to be easy to use.	Information does not need to be stored in the clear to be easy to use.
You keep users from leaking information by establishing rules and hoping they follow them.	You keep users from leaking information by establishing rules that their applications enforce.

ESTABLISHING PERSISTENT DISTRIBUTION CONTROL AT THE OBJECT LEVEL

An individually protected object is called a SEGCO—a Secure Extensible Global Content Object. The application architecture that supports the creation, storage, transportation and authentication of SEGCOs, along with the application programming interface (API) that makes it available to developers, is called the Persistent Distribution Control System (PDCS). PDCS enables developers to build persistent distribution control into their applications.



PDCS-enabled applications encapsulate information in SEGCOs as new information is created.

PDCS-enabled applications store and transmit only SEGCOs.

Information at rest and in motion is automatically and always secured.

What do attackers get if they breach a client device, server or signal?

Only SEGCOs.

Key SEGCO/PDCS attributes include the following:

- SEGCOs are individually encrypted, each with a distinct key. That radically reduces the number of useful attack vectors and offers the most resistant of all attack surfaces. Attacks designed to copy and illicitly export data fail to deliver information attackers can use.
- PDCS stores SEGCOs uniformly. SEGCOs are indistinguishable from each other. This makes getting the right information object analogous to finding a particular grain of sand on a beach of identical grains of sand. The more grains of sand, the harder the problem for attackers.
- SEGCOs are transmitted in an encrypted tunnel. A breached tunnel yields only SEGCO's.
- PDCS and SEGCOs make it possible to build tools and applications that work across almost any kind of hardware and OS. Information that is secured in a SEGCO from the moment it is created will be safe in motion and at rest wherever it exists.
- SEGCOs and PDCS make it possible to build mechanisms to authenticate users, devices and applications prior to decrypting the information (chain-of-trust fingerprinting).
- The consistency of SEGCO metadata and the audit function of PDCS make it possible to implement auditing systems that monitor the movement and use of information in near real time. Those systems may be able to flag the misuse of information objects fast enough to keep the attack from succeeding.
- The comprehensiveness and immutability of the audit trail can establish a legally admissible chain of custody to aid in prosecution.
- SEGCOs and PDCS will provide an array of capabilities that application developers could use to create new solutions, including:
 - Fine-grained control of secondary distribution such as forward, export, copy/paste, or print
 - Secure documents that redact themselves as they move through distribution
 - Cross-domain secure collaboration without interdomain access
 - Copyright protection that does not unduly restrict the user's access to content
 - Commercial authentication without risk of exposing personally identifiable information

SEGCO-PDCS REQUIREMENTS AND ARCHITECTURE

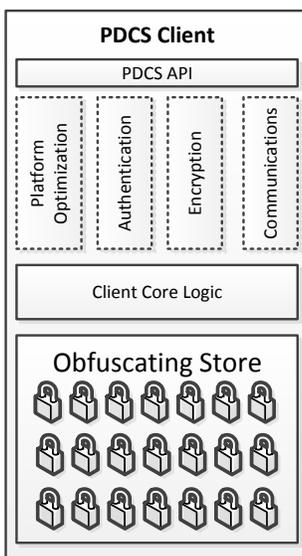
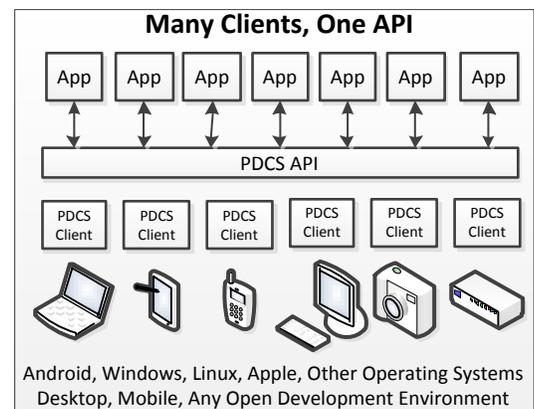
Basically, a SEGCO must:

- Be content-agnostic (support any data type)
- Be distinctly encrypted (provide a different key)
- Remain encrypted in motion and at rest
- Contain distribution control information inside the encrypted envelope
- Contain audit information inside the encrypted envelope
- Not indicate the type of content it contains
- Be randomly and/or nonsensically named



PDCS must be implemented using a client-server architecture to:

- Disperse the encrypt/decrypt processor load across millions of edge processors
- Provide a common multiplatform security API
(This is critical. Developers rarely have the knowledge required to build information security into their applications^{8,9} and it would likely do more harm than good if their security solutions were all different. Methods and developer tools for securing information objects must be easy to use, standardized, broadly applicable and inexpensive.)
- Support disconnected use
- Enable secure storage on edge devices and servers
- Enable a hierarchical server architecture (place servers where needed)
- Enable secure transport and interim storage for applications that require large amounts of data in the clear (cloud to cloud, cloud to edge)
- Enable intelligent prioritization of bandwidth



PDCS clients must be OS-optimized, authentication method agnostic, encryption method agnostic and communications method agnostic (support any Internet protocol transmission path) so that clients can run efficiently on any platform.

The PDCS client architecture must provide:

- An API with common calls across platforms
- Platform-specific performance optimization
- Platform-specific module for authentication
- Platform-specific module for encryption
- Platform-specific module for communications
- Common core logic across platforms
- A SEGCO store that makes SEGCOs uniform

PDCS content servers should be as naïve and lightweight as possible—and they can be since client devices are performing application processing and “heavy lifting” such as encrypt/decrypt. The key functions of PDCS servers will be to:

- Route, store and forward SEGCOs
- Limit access to authenticated users and devices (*no* support for system level anonymity)
- Provide mesh network management functions
- Provide resilience and disaster recovery functions

OUTCOMES OF SEGCOs AND PERSISTENT DISTRIBUTION CONTROL

What current hard problems would be simplified if applications stored and moved information in SEGCOs managed through a PDCS application architecture? These are some we have identified:

- Insider threat: If Bradley copies 200,000 SEGCOs to a Lady Gaga CD and gives them to Julian, Julian will have a useless CD—and nothing to publish.
- Malware-based storage attacks: If malware is able to convey only SEGCOs to outsiders, the malware is of no value to the malware writer or purveyor.
- Signal intercepts (man-in-the-middle attack): If a signal is intercepted and the signal contains only a stream of SEGCOs, the intercept is of no value to the eavesdropper.
- Device capture: If a device containing thousands or more SEGCOs is captured, the device has no value to the thief.
- Personal identity protection: Applications can be built that enable the complete separation of personally identifiable information from the information objects that represent the person.
- Cross-domain information sharing without cross-domain access: Applications can be built that support software-based secure intermediate logical networks.
- Copyright preservation without undue restrictions on usability: Applications can be built that ensure copyright holders can track and get paid for their content while allowing buyers to access their content on whatever device they are using at the moment.
- Intelligent traffic prioritization: Applications can be built that support client-based prioritization of traffic—a critical need anywhere bandwidth is scarce and urgency is high.

REMAINING VULNERABILITIES

Persistent distribution control does not eliminate information security problems. It shrinks the problem considerably by undermining the value of storage and eavesdropping exploits, reducing the requirement for users to follow security policies and increasing the risk, time, and effort of attempting exploits.

However, full information security will require that we close the holes “at the bottom of the stack” with solutions such as separation kernels, secure Operating Systems, encrypted memory, effective micro virtual machines, robust chain-of-trust solutions, and secure hardware and application supply chains.

By shrinking the size and scope of the threat universe, PDCS allows cybersecurity professionals and the industry to focus on the small number of truly sophisticated attacks and attackers. If that occurs, attackers will need to have more expertise, expend more capital and take on more risk. Attacking will become much more expensive and have a much smaller chance of success. Information that protects itself ruins the economics of hacking.

ENDNOTES

- ¹ Devlin Barrett, "[U.S. Outgunned in Hacker War](#)", Wall Street Journal, March 28, 2012.
- ² ThreatPost, Kaspersky Labs: [Experts Tell Senate: Government Networks Owned, Resistance Is Futile](#)
- ³ Axel Buecker, Per Andreas and Scott Paisley, "[Understanding IT Perimeter Security](#)," IBM Corporation, 2008.
- ⁴ Marcia Savage, "[Perimeter Defenses Deemed Ineffective Against Modern Security Threats](#)," *Information Security*, June 30, 2010.
- ⁵ Simson Garfinkel, "[The Deperimeter Problem](#)," CSO Online, November 1, 2005.
- ⁶ Anh Nguyen, "[Infosec: Cloud Computing 'Explodes' the Security Perimeter](#)," CSO Online, April 25, 2011.
- ⁷ "[Top Cause of Data Breaches? Negligent Insiders](#)," Help Net Security, Ponemon Institute, March 22, 2012.
- ⁸ Adam Cummings and Ron Bendes "[Information Security in Application Development Projects](#)," cmu95752, Carnegie Mellon University, March 7, 2012.
- ⁹ "[Risk Across the Phases of Application Security](#)," Help Net Security, Ponemon Institute, March 21, 2012.

Android is a trademark of Google Inc.

Apple is a trademark of Apple Inc.

Bluetooth is a registered trademark of Bluetooth SIG, Inc.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Windows is a registered trademark of Microsoft Corporation.