Trends in Data Centric Security

Version 1.0
Released: September 14, 2014

Securosis, L.L.C. 515 E. Carefree Highway Suite #766 Phoenix, AZ 85085 **T** 602-412-3051
info@securosis.com www.securosis.com

Author's Note

The content in this report was **developed independently of any sponsors or licensees**. It is based on material originally posted on the Securosis blog, but has been enhanced, reviewed, and professionally edited.

Special thanks to Chris Pepper for editing and content support.

Licensed by Intel® Services

Intel® Mashery™ API Gateway - Tokenization is a hardware or software appliance that dramatically simplifies PCI DSS & PII compliance and administration for organizations across all industry types that accept, capture, store, transmit or process credit card, debit card, or sensitive personal data. As such, it functions as a tokenization broker for any enterprise application tasked with handling clear-text primary account number (PAN) data. The data is tokenized in documents or API calls and it stores encrypted card data in a protected, secure vault. Tokenization is a great way to improve enterprise security.

To learn more, please visit services.intel.com

Contributors

The following individuals contributed significantly to this report through comments on the Securosis blog and follow-on review and conversations:

Marco Tietz Khürt Williams Scott Yoneyama

Copyright

Table of Contents

Introduction

You want to make data useful, by making it available to users and applications that can leverage that data into actionable information. You share data between applications, partners and analytics systems to derive the maximum business intelligence possible. But what do you do when you can't guarantee the security of these systems? How do you protect information data regardless of where it moves? One approach is called Data Centric Security, and it's designed to protect *data* instead of the *infrastructure*.

IT systems *exist* to provide and process data. But when we approach security we spend an inordinate amount of time and money protecting the network, servers and endpoints. With the mobile devices now more commonly used that PCs, and the growing use of 3rd party providers to deliver 'cloud' storage, we are creating more copies of data than ever before. It's storage platforms. common to stream several event feeds into 'Big Data' analytics clusters and cloud services, but we're forced to adapt old security models despite new security challenges with these new environments. This means we've simultaneously increased usefulness of data while reducing security and control.

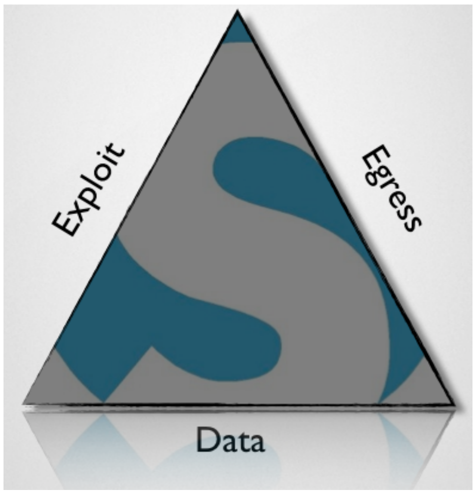
This is what Data Centric Security (DCS) does: focus security controls onto the data, rather than into servers or supporting infrastructure. This approach means data is secure wherever it moves. The challenge is to implement security controls that do not render the data inert. Put another way, you want to derive value from data without leaving it exposed. Sure, we can encrypt everything, but you cannot analyze encrypted data. Nor can you expect to be able to securely distribute key management and decryption capabilities everywhere data moves. But what you can do is allow data to be protected everywhere, without exposing sensitive information.

In response to customer inquiries about what to do when moving data to locations they do not completely trust, this research paper focuses on how to protect information regardless of where it is moved. The majority of these inquires are motivated by "big data" usage as firms move data into NoSQL clusters. The gist is that we don't know how to secure these environments, we don't really trust them, and we don't want a repeat of data leakage or compliance violations. In this research paper we will detail what DCS is, how it works, and what problems it can be expected to address.

Use Cases

Data Centric Security provides security even when the systems processing data cannot be fully trusted, which is the reasons customers are interested in this approach. We can both propagate and *use* data to derive business value while still maintaining a degree of privacy and security. Sounds like a fantasy, but it's real.

Let's underscore why this approach is important. For any data breach to occur you need three things; a way 'in' to the systems that manage data (i.e.: an exploit), data to steal, and a means to exfiltrate that data. We call this the 'Data Breach Triangle'. Remove any one of the three requirements and you've stopped the breach. But we've been trying — unsuccessfully — for a decade or more to stop attackers from getting into corporate networks. Now we are adding more access points with mobile, and more 3rd party cloud services, which move data in and out existing IT systems. But if we can substitute data with a secure surrogate we've addressed the problem, regardless of where the data is stored.



But of course there are challenges, which we will detail later in this paper. For now our focus is to illustrate the trend of why customers are interested in this approach. We will examine the specific business use cases they are support, but looking to implement security and compliance controls. These are the principle customer inquiries that prompted this research.

NoSQL / Big Data Security

The single biggest reason we are asked about data centric security models is "big data": moving information into NoSQL analytics clusters. Big data systems are a new type of database that facilitates fast analysis and lookup capabilities on much larger data sets — at dramatically lower cost — than previously possible. To take full advantage of these databases, lots of data is collected from dozens of sources. The problem is that many sources fall under one or more regulatory controls because they contain sensitive data, but big data projects are typically started and run outside regulatory or IT guidance. As custodians become aware of their responsibility for NoSQL data and services, they realize they are unable to adequately secure the cluster — or even know exactly what it contains.

NoSQL databases have proven their value by offering previously unavailable scale for analytics, making their value to many organization indispensable. Unfortunately, from a security perspective, they are often too immature for enterprises to fully trust. Most of the security controls firms currently leverage on relational platforms are deficient or completely unavailable with NoSQL databases; there simply is no embedded capability to implement the security controls enterprises rely upon.

Data centric security offers critical security for such systems which process sensitive data but cannot themselves be fully secured. By protecting data *before* moving it into a big data repository, transforming data into something non-sensitive which can be analyzed is the value proposition, you provide the value of large scale analytics without the exposure.

Cloud and Data Governance

Most countries have laws on how citizen data must be secured, outlining custodial responsibilities for companies which store and manage it. These laws differ from country to country on how data must be secured, if certain types of data can be moved outside national borders, and what is required in case of a breach. If your IT systems are all within a single data center, in a single location under your control, you only need worry about your local laws.

But cloud computing make compliance much more complex, especially in public clouds. First, cloud service providers are legally third parties, with deliberately opaque controls and limited access for tenants (customers like *you*). Second, for reliability and performance reasons, cloud data centers span multiple geographic locations with different laws. This means *multiple* — possibly conflicting — regulations apply to sensitive data, and you share responsibility with your cloud service providers.

The legal issues break down into three types: functional, jurisdictional, and contractual. Functional issues include how legal discovery is performed, what happens in the event of a subpoena or legal hold, proof of data guardianship, and legal seizure in multi-tenant environments. Jurisdictional issues require you to understand applicable legislation, under what circumstances the law applies, and how legal processes differ. Contractual issues cover access to data, data lifecycle management, audit rights, contract termination, and a whole heap of other issues including security and vulnerability management. Regardless of which issues you need to address, DCS provides different approaches to meeting those requirements.

Many firms want to leverage low-cost on-demand cloud computing resources, but hesitate at the huge burden of data governance in and across cloud providers. This is a case where data centric security can reduce compliance burdens and resolve many legal issues — hopefully meaning fewer reports, fewer controls, and less complexity to manage.

Protected Health and Personal Information

Queries on how to secure HIPAA and Protected Health Information (PHI) were almost non-existent a couple years ago, but we are now asked about them with increasing frequency. We consider health care data to me an emerging use case; one which encompasses many different kinds of sensitive data, and the surrounding issues are complex. A patient's name is sensitive data in some contexts. Medical history, medications, age, and just about every other piece of data is critical to some audiences, but too sensitive to share with others. Some patients' data can be shared in certain limited cases but not in others. And there many audiences for PHI: state and federal governments, hospitals, insurance companies, employers, organizations conducting clinical trials, pharmaceutical companies, and more. Each audience has a legitimate interest in a particular data subset, and restrictions on their access.

Today data centric security manage subsets of databases to different audiences, with surrogate data for elements which each audience is *not* allowed to possess. As data storage and management systems become cheaper, faster, and more powerful, creating — and tracking — a unique copy for each audience is now feasible. Each recipient can securely access its own copy, containing only its permitted data. Data centric security enables organizations to provide just those data elements which partners need, without exposing data they must not access. And this can all be done in real time on demand, by applying appropriate controls to transform the original data into the secured subset. Many tools and techniques developed over the last several years for test data management are now employed to generate custom data sets for individual partners on an ongoing basis.

Payment Card Security

The widespread theft of credit card data is a huge problem for merchants and banks, resulting in billions of dollars in fraud losses each year. Despite the rollout of the Payment Card Industry's Data Security Standard Today (PCI-DSS) theft of credit card data has contented to increase in the US. Attackers have been able to take over entire data centers, installing malware that exfiltrates transaction data on everything from point-of-sale devices to payment servers. In fact, at one time or another, every single IT component governed by PCI-DSS has been exploited in a breach.

While PCI-DSS security controls are a competent set of *minimum security requirements*, it's still not enough. The fundamental problem is we continue to pass credit card information through complex, difficult to secure environments, where a single security failure results in a catastrophic breach. A large percentage of merchants have adopted tokenization — one of the key data centric security approaches — to reduce their compliance requirements. But for many who no longer process or store credit card data, the data centric security approach has eliminated the problem entirely.

Tools

The three basic data centric security tools are tokenization, masking, and data element encryption. Each has a role in data security and each is part of robust data security architecture. As mentioned in the foregoing sections, traditional security measures are challenged to keep pace with persistent and escalating threats. Data centric security tools not only help the challenges of data growth and proliferation, but also provide systemic support for protecting data at rest and when accessed by applications; regardless if the data is used for production or non-production (test, support, training) purposes. These tools can be used as a principle security control, or supplement existing defenses for external compromise or from insider threats. Here we will discuss what they are, how they work, and which security challenges they address best.

Tokenization

You can think of tokenization in terms of a subway or arcade token: it has no cash value but can be used to ride a train or play a game. In data centric security a token is provided in lieu of sensitive data. The most common use today is in credit card processing systems, as a substitute for credit card numbers. A token is basically just a random number — that's it. The token can be made to look just like the original data type — for example credit card tokens are typically 16 digits long, preserving the last four digits from the original card number, and can even be generated to pass the LUHN validation check. But aside from the constraints a token credit card number is random, with no mathematical relationship to the original, with the original value stored in another (more secure) database.

Adoption of tokenization was fueled by two key facets of the technology; it preserved application functionality while offering better security. For example, many relational databases were keyed off a Social Security number as it was a convenient — and unique — customer identifier. But you could neither encrypt, nor remove, the SSN without causing the database to crash. Enter tokenization, where a data-type and format preserving token replaced real SSNs. Tokenization systems are typically embedded in-line with transactional systems, substituting data as it moves through the system. Some tokenization solutions replace data in databases or files, but most work on data streams — such as replacing unique cell identification number as data packets move across their networks. This enables both simple and complex data to be tokenized, at rest or in motion — and tokens can look like anything you want.

Tokenization has not seen wide adoption outside of credit card processing and government safeguards for PII. The technology does not easily lend itself to complex data types, nor does it come with the data-management features need to create different types of tokens to dynamically serve multiple audiences. For example, several products on the market can create tokens for complex data types such JSON & XML files, but this is a static relationship, and requires the original values be stored in a 'token vault' or other secure repository. And while token provide secure surrogate, most retain little value from the original. While a credit card may retain the last four digits, tokens do retain aggregate value of the *data set*, a principle requirement for many 'big data' and health care projects.

Masking

Masking has been around for many years, but is emerging as a popular option for protecting data elements while retaining aggregate values of data sets. For example, masking can provide a simple substitution of an individual's Social Security number with a random number (like tokenization), or more complex substitutions. One example might be of a persons name — randomly selected from a phone directory — while retaining gender. We might replace date of birth with a random value within X days of the original value to effectively preserve age. This way the original sensitive value is removed without randomizing the value out of the aggregate data set, to support later analysis.

Masking is the most popular way to create *useful* new values without exposing the originals. It is ideally suited for creating data sets which can be used for analysis — or application testing — without exposing the original data. This is important when you lack sufficient resources to secure every system within your enterprise, or don't fully trust the environment where the data is being stored. Different masks can be applied to the same data fields, to produce different masked data for different use cases. This flexibility can be used to expose much of the value of the original data with minimal risk. Masking is very commonly used to secure data for PII, PHI, and PCI compliance requirements, as well as promote test data management and NoSQL analytics without increasing risk of exposure.

Data masking is deployed in two ways: Persistent data masking where stored data is changed/masked, and dynamic masking which provides altered data on demand. For non-production purposes, such as application testing and analytics, persistent masking allows organizations to de-identify and de-sensitize data for testing, training, or even sharing outside the organization. Dynamic data masking applies masks to data as it is requested, in real-time, altering what is delivered — not what is stored. For example, based upon a users role or credentials, a database query may return the real data stored within the database, or it may return a de-sensitized copy of that data. This allows organizations to implement a risk-based mask, ensuring "least privilege' and "need to know" data access across multiple databases and user types.

That said, masking requires careful consideration before implementation. De-identifying records requires understanding of how various fields could be used to statistically re-identify information and thus leak information. It also requires that the user understand which masks are suitable for various regulations and laws. Some data masks (*e.g.: replacement with a random value)* are completely secure and produce surrogates with no mathematical relationship to the original value. Some masking techniques have a mathematical relationship to the original, and must be used cautiously. Harder still is understanding when anonymized attributes in a data set — for example a person's health record (age, gender, zip code, race, DoB, etc.) — may provide sufficient information to reverse engineer the masked data back to the original record. Masking can be very secure, given carefully chosen masking tools and a well-reasoned masking function, to achieve security and privacy goals while supporting desired analytics.

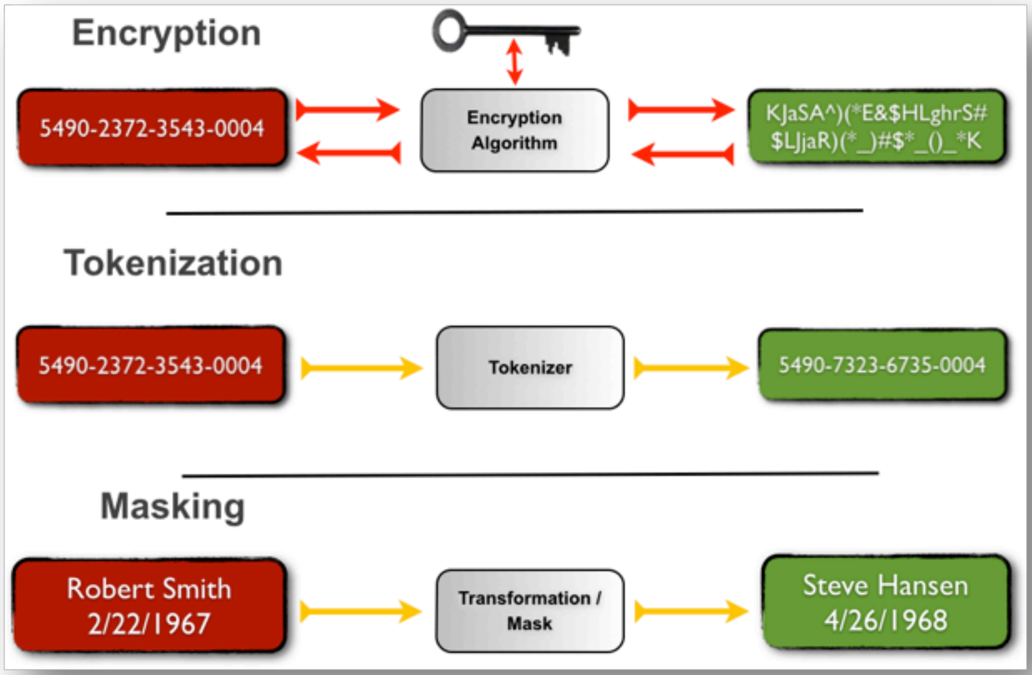
Element/Data Field Encryption / Format Preserving Encryption (FPE):

Encryption is the go-to security tool for the majority of IT and data security challenges today. Properly implemented encryption provides obfuscated data that *cannot* be reversed into the original data value without an encryption key. That said, most encryption ciphers were designed for files and disk volumes, and were not suitable for data replacement in databases and files. Encrypted data was in binary format, and of an indeterminate length, which applications could not process without first decrypting the data. It's only with recent advancements in format and data-type preserving techniques has encryption been a viable option for use in applications and databases, and really address the data centrical model. Now encryption can be applied to any type of data — such as first and last names stored in a file — and still be processes by the application without error. And encryption keys can be provided to select users, keeping data secret from those not entrusted with keys.

Encryption also offers a — potentially — significant advantage for compliance, namely some U.S. compliance mandates offer blanket approval for encryption: when data is encrypted it's automatically compliant. And encryption vendors typically offer encryption for complete files and disk volumes, providing added versatility. Encryption is also widely integrated with cloud infrastructures, most application platforms, and is increasingly common with packaged NoSQL solutions to protect data files on disk. Properly selected and deployed, encryption protects data very effectively, in diverse scenarios and use cases.

That said, encryption does not provide de-identifying or de-sensitizing of data for analytical or test purposes. Several EU data protection mandates still do not permit encrypted personal information to pass across national boundaries, while masking and tokenization are. Worse still, long-hidden vulnerabilities are sometimes discovered in ubiquitous encryption tools, exposing user data. It is critical that encryption cipher's be carefully selected to suit the use case, that platforms should be obtained from a trusted source, and special attention be placed on key management and key security — otherwise the entire model fails.

It is also worth mentioning homomorphic encryption, a topic which gets considerable media coverage despite not yet being very useful. Homomorphic encryption is concept, like masking, where encrypted data could still provide value to analytics. For example, homomorphic encryption would allow summation and averages to be calculated on encrypted data columns. It's a great idea, but the reality is still little more than a research concept. Real-world homomorphic systems either require vast computing resources which make them uneconomical or are based on compromised variants of standard encryption protocols. They enable basic analytics on encrypted data, but security is compromised, failing on the principal requirement. Scientific advances may make homomorphic encryption a viable alternative within our lifetimes, but today it is simply impractical — or worse, snake oil.



Redaction

This method of substitution simply replaces sensitive data with a generic value, such as the digit 5 with 'X'. For example, we could replace a phone number with "(XXX)XXX-XXXX", or a Social Security Number (SSN) with XXX-XX-XXXX. This is the simplest and fastest form of masking. It's very effective at preserving data privacy, but the output provides little or no value for analysis — it's just a blind placeholder. It's commonly used when performance is critical, such as in cases where data is replaced in real time as it moves across a network, without any chance for intelligence substitution.

Discovery

Most data centric security solutions include a discovery module. You need to locate sensitive data to know what needs to be secured, and you must identify the type of sensitive information to understand how to protect it.

Data discovery tools are typically geared toward sifting through files or relational databases. *File* discovery tools first look at storage volumes to see what types of files they contain, then scan their contents for sensitive data. File scanners need credentials for access to the files to scan — typically a user account with read-only access. Some database scanners can identify databases on the network by scanning IP addresses but most need to be pointed specifically at databases to scan. Like file scanners, they need credentials to read relational tables.

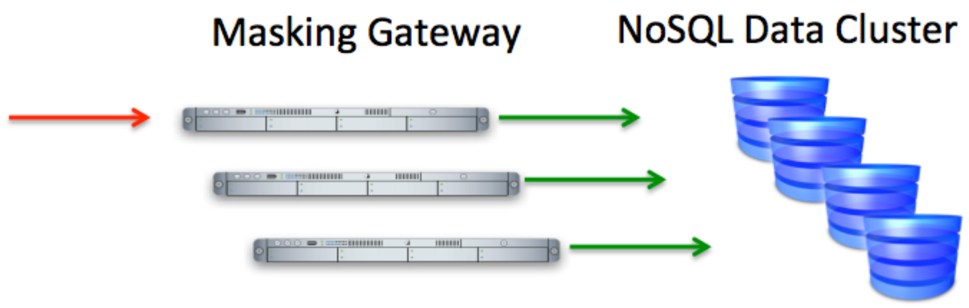
Discovery tools commonly employ two methods to discover content: metadata and regular expression checks. Metadata checks are common for database scanners: rather than examine every data element in the database, they look at the column length, column name, and structure of column. A column of 16-digit numbers with 'C' appearing twice in the column name is probably a credit card field. Regular expression checks are also common — they examine the data itself rather than the database schema. You can try to pattern-match the contents of a file or database column — 16 digits is the most popular pattern — and then check whatever matches. More advanced scanners employ heuristics to speed up scanning, or tricks such as positional checks (*e.g.,* the third element in a set of associated values is often a Social Security number). There are many types of discovery but they all seek to locate sensitive data and identify its type, and often to map each data type to a security control. We will provide more examples of discovery below.

Deployment Models

So far we have talked about the need for data centric security, what it is, and which tools fit the model. Now it is time for specifics on how to implement and deploy data centric security, so here are some concrete examples of how these tools are deployed to support a data centric model.

Gateways

A gateway is typically an appliance that sits in-line with traffic and applies security as data passes. Data packets are inspected in real time and sensitive data is replaced or obfuscated before packets are passed on.



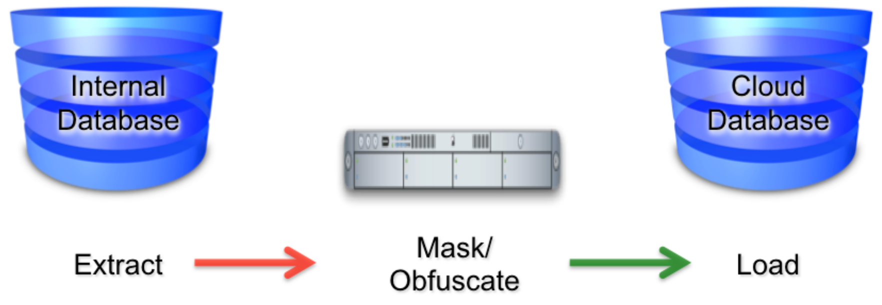
Gateways are commonly used by enterprises to filter data moving off-premise, such as up to the cloud or to another third-party service provider. The gateway sits inside the corporate firewall, at the 'edge' of the infrastructure, discovering and filtering out sensitive data. For example some firms encrypt data before it is moved into cloud storage for backups. Others filter web-based transactions inline, replacing credit card data with tokens without disrupting web servers or commerce applications. Gateways offer high-performance substitution for data in motion — but they must be able to parse the data stream to encrypt, tokenize, or mask sensitive data.

Another gateway deployment model puts appliances in front of big data (NoSQL databases such as Hadoop), replacing data before insertion into the cluster. But support for high "input velocity" is a key feature of big data platforms. To avoid crippling performance at the security bottleneck, gateways must be able to perform data replacement while keeping up with the big data platform's ingestion rate. It is not uncommon to see a cluster of appliances feeding a single NoSQL repository, or even to spin up hundreds of cloud servers on demand to mask or tokenize data. These service must secure data very quickly so they don't provide deep analysis. Gateways may even need to be told the location of sensitive data within the stream to support substitution.

Hub and Spoke

In this model a central obfuscation 'hub' pulls data from one or more sources, applies security to the data, and then moves it out to one or more data destinations. In this model DCS is a form of data management, and requires you to apply security polices, transform the data, and orchestrate distribution through policies on the central hub.

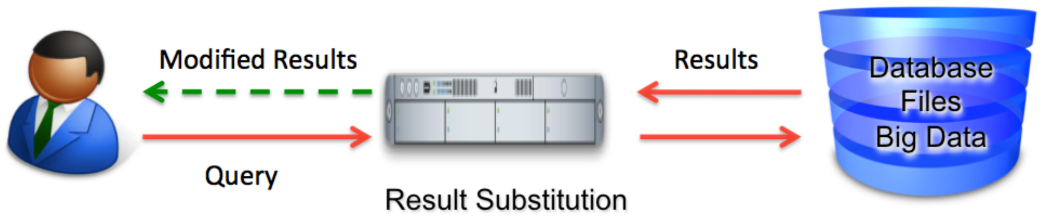
ETL (Extract, Transform, and Load) has been around almost as long as relational databases. It describes a process for extracting data from one database, transforming it, then loading the de-sensitized data into another database. Hub and spoke is simply an advanced deployment model for ETL, adding in data management and automation capabilities. Over the last several years we have seen a huge resurgence of ETL as firms look to populate test databases with non-sensitive data that still provides a reliable test bed for quality assurance. A masking 'hub' orchestrates data movement and implements security. A common approach to sharing PHI and PII (Personally Identifiable Information) across multiple locations with inconsistent or unknown security. The model is used to create one or more data sets to be stored remotely, rather than securing streams of data.



The graphic above shows ETL in its most basic form, but the old platforms have evolved into much more sophisticated data management systems. They can now discover data stored in files and databases, morph together multiple sources to create new data sets, apply different masks for different audiences, and relocate the results — as files, as JSON streams, or even inserted into a data repository. This is a form of data orchestration, moving information automatically according to policy. Plummeting compute and storage costs have made it feasible to produce and propagate multiple copies of secure data, each tailored to a specific audience.

Dynamic Masking / Reverse Proxy

As with the gateways described above, in the reverse-proxy model an appliance — virtual or physical — is inserted inline into the data flow. Unlike the gateways mentioned above, reverse proxies are placed between users and a single database. Offering much more than simple positional substitution, proxies can alter what they return to users by recipient and specifics of the request. They work by intercepting and masking query results on the fly, transparently substituting masked results for the user. For example if a user queries too many credit card numbers, or if a query originates from an unapproved location, the returned data might be redacted. The proxy effectively intelligently — *dynamically* — applies a suitable mask to data based upon the user making the request. A proxy may be an application running on the database or an appliance deployed inline between users and data to capture all communications. The huge advantage of proxies is that they enable data protection without needing to alter the database — avoiding additional programming and quality assurance validation processes.



This model is suitable for PCI/PII/PHI data, when data can be managed from a central location but external users may need access. Some firms implement tokenization this way, but masking and redaction are more common. The principal use case is to protect data dynamically, based on user identity, location and the request itself.

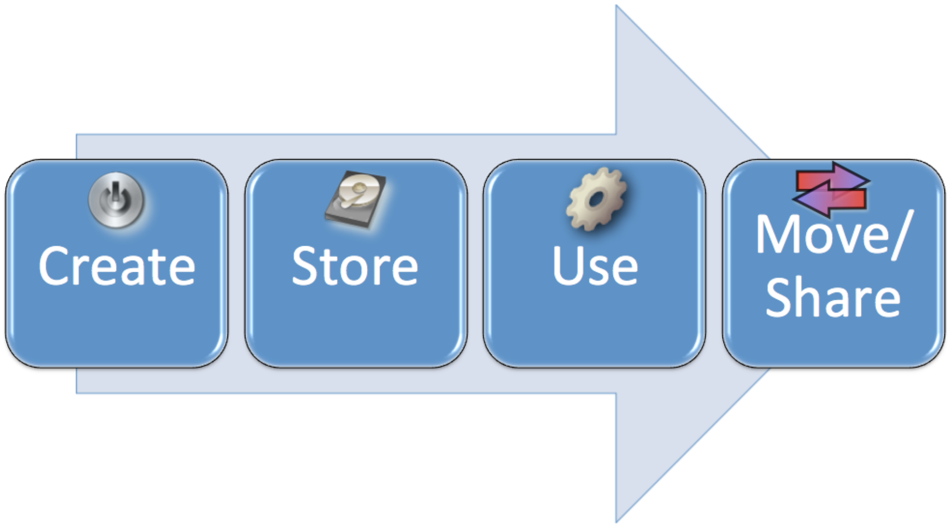
Recommendations

So where should you start? Data centric security requires a change in mindset and approach. It is difficult to move away from a network or firewall centric security model because the vast majority of tools are geared to those models. You need to bake DCS into your data management model. A good starting place is to consider the types of data in use in comparison with the security tools at your disposal. The following overview is how customers describe their use of these tools with a data-centric deployment model.

	Static Masking	Dynamic Masking	Tokenization	FPE
Financial Data			X	X
Application Data	X	X	X	
Test Data	X			
Data Warehouse / Big Data	X			
PII		X	X	X
PHI	X			X

It is also helpful to consider the data lifecycle — where data is, where it moves, and how it is used — and then figure out the tools and deployment model you need to secure it. Once you know what you *do* with your data you have a much better perspective on how to protect it. That requires learning where data is moving and what sensitive information you have, so that knowledge can guide your technology selection and deployment model. A simple example is storing data in the cloud: encryption is effective protection for data at rest. But if you are sharing sensitive data between multiple parties with different levels of security access, a combination of redaction and masking might be the right approach.

The threat of a breach may or may not provide sufficient impetus for firms to seriously re-examine security. And even when companies do re-examine their security approach, they do not necessarily adopt a data centric security model. Focusing on data is logical, but it is still an unusual way for firms



to consider security. More often they ask "How do attackers get in, and how can I stop them?" If the threat *du jour* is phishing and malware, customers tend to respond by trying to stop phishing and malware. But if the threat is SQL injection, cross-site scripting, or weak passwords... security deployments follow threats. These risks demands a larger awareness, and sufficient bravery to focus on the data in the heart of the data center, rather than on the perimeter. Securing data first, to provide "security from the inside out", requires a different mindset than the ever-popular never-ending threat vs. patch ping-pong. For some use cases firms are realizing that traditional approaches won't work, and seeking out data protection options which can work in their environments. Compliance with data security and privacy laws are typically the driver for investment in security technologies, and a data centric security provides a simple approach that remains effective regardless of the environment or threat model.

If you have any questions on this topic, or want to discuss your situation specifically, feel free to send us a note at info@securosis.com or ask via the Securosis Nexus <http://nexus.securosis.com/>.

About the Analyst

Adrian Lane, Analyst and CTO

Adrian Lane is a Senior Security Strategist with 25 years of industry experience. He brings over a decade of C-level executive expertise to the Securosis team. Mr. Lane specializes in database architecture, data security and secure code development. With extensive experience as a member of the vendor community (including positions at Ingres and Oracle), in addition to time as an IT customer in the CIO role, Adrian brings a business-oriented perspective to security implementations. Prior to joining Securosis, Adrian was CTO at database security firm IPLocks, Vice President of Engineering at Touchpoint, and CTO of the secure payment and digital rights management firm Transactor/Brodia. Adrian also blogs for Dark Reading and is a regular contributor to Information Security Magazine. Mr. Lane is a Computer Science graduate of the University of California at Berkeley with post-graduate work in operating systems at Stanford University.

About Securosis

Securosis, LLC is an independent research and analysis firm dedicated to thought leadership, objectivity, and transparency. Our analysts have all held executive level positions and are dedicated to providing high-value, pragmatic advisory services. Our services include:

- **The Securosis Nexus:** The Securosis Nexus is an online environment to help you get your job done better and faster. It provides pragmatic research on security topics that tells you exactly what you need to know, backed with industry-leading expert advice to answer your questions. The Nexus was designed to be fast and easy to use, and to get you the information you need as quickly as possible. Access it at <https://nexus.securosis.com/>.

- **Primary research publishing:** We currently release the vast majority of our research for free through our blog, and archive it in our Research Library. Most of these research documents can be sponsored for distribution on an annual basis. All published materials and presentations meet our strict objectivity requirements and conform to our Totally Transparent Research policy.

- **Research products and strategic advisory services for end users:** Securosis will be introducing a line of research products and inquiry-based subscription services designed to assist end user organizations in accelerating project and program success. Additional advisory projects are also available, including product selection assistance, technology and architecture strategy, education, security management evaluations, and risk assessment.

- **Retainer services for vendors:** Although we will accept briefings from anyone, some vendors opt for a tighter, ongoing relationship. We offer a number of flexible retainer packages. Services available as part of a retainer package include market and product analysis and strategy, technology guidance, product evaluation, and merger and acquisition assessment. Even with paid clients, we maintain our strict objectivity and confidentiality requirements. More information on our retainer services (PDF) is available.

- **External speaking and editorial:** Securosis analysts frequently speak at industry events, give online presentations, and write and/or speak for a variety of publications and media.

- **Other expert services:** Securosis analysts are available for other services as well, including Strategic Advisory Days, Strategy Consulting engagements, and Investor Services. These tend to be customized to meet a client's particular requirements.

Our clients range from stealth startups to some of the best known technology vendors and end users. Clients include large financial institutions, institutional investors, mid-sized enterprises, and major security vendors.

Additionally, Securosis partners with security testing labs to provide unique product evaluations that combine in-depth technical analysis with high-level product, architecture, and market analysis. For more information about Securosis, visit our website: <http://securosis.com/>.