Understanding and Selecting Data Masking Solutions: Creating Secure and Useful Data

Version 1.0
Released: August 10, 2012

Securosis, L.L.C. 515 E. Carefree Blvd. 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. 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, and to Rich Mogull for his Five Laws of Data Masking.

Licensed by IBM

IBM InfoSphere Optim and InfoSphere Guardium solutions for data security and privacy protect diverse data types across different locations throughout the enterprise, including the protection of structured and unstructured data in both production and non-production (development, test and training) environments. Such an approach can help focus limited resources without added processes or increased complexity. A holistic approach also helps organizations to demonstrate compliance without interrupting critical business processes or daily operations.

IBM InfoSphere solutions for data security and privacy support heterogeneous enterprise environments including all major databases, custom applications, ERP solutions and operating platforms. IBM InfoSphere Optim and InfoSphere Guardium scale to address new computing models such as cloud and big data environments.

IBM and the IBM logo are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. For more information please visit: www.ibm.com

Licensed by Informatica

Informatica Corporation is the leading independent provider of enterprise data integration software. Our mission is to help organizations increase their competitive advantage by maximizing their return on data to drive their top business imperatives and information technology initiatives. Data is one of an organization's most strategic assets, and we continue to develop solutions for nearly every type of enterprise, regardless of which technology platform, application, or database customers choose. We address the growing challenge organizations face with data fragmented across and beyond the enterprise and with data of varying quality.

Informatica solutions for data privacy address the unique challenges that IT organizations face in securing sensitive data in both production and nonproduction environments. Based on an innovative, open architecture, these comprehensive solutions are highly flexible and scalable, and provide for real-time or batch processing – dynamic or persistent. With easy, fast, and transparent deployment, plus a centralized management platform, Informatica solutions for data privacy will help your organization to protect and control access to your transactional and analytic data, reduce compliance, development and testing costs, and ensure data privacy.

For more information, visit: www.informatica.com

Copyright

This report is licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0.

http://creativecommons.org/licenses/by-nc-nd/3.0/us/

Table of Contents

Introduction 2

Use Cases 3

Defining Data Masking 6

- Data Masking Definition 6
- Five Laws of Data Masking 6
- Masks 7
- Masking Constraints 8

How Masking Works 11

- Deployment Models 11
- Data Sources 15

Technical Architecture 17

- Base Architecture 17
- Deployment and Endpoint Options 18

Platform Management 20

- Central Management 20

Advanced Features 22

Buyer's Guide 24

Conclusion 28

About the Author 29

About Securosis 30

Introduction

Very few data security technologies can simultaneously protect data while preserving its *usefulness*. Data is valuable because we use it to support business functions — its value is *in use*. The more places we can leverage data to make decisions the more valuable it is. But as we have seen over the last decade, data propagation carries serious risks. Credit card numbers, personal information, health care data, and good old-fashioned intellectual property are targets for attackers who steal and profit from other people's information. To lessen the likelihood of theft, and reduce risks to the business, it's important to eliminate both unwanted access and unnecessary copies of sensitive data. The challenge is how to accomplish these goals without disrupting business processes and applications.

Data masking technology provides data security by replacing sensitive information with a non-sensitive proxy, but doing so in such a way that the copy of data that looks — and acts — like the original. This means non-sensitive data can be used in business processes without changing the supporting applications or data storage facilities. You remove the risk without breaking the business! In the most common use case, masking limits the propagation of sensitive data within IT systems by distributing surrogate data sets for testing and analysis. In other cases, masking will dynamically provide masked content if a users request for sensitive information is deemed 'risky'. Masking architectures are designed to fit within existing data management frameworks and mitigate risks to information without sacrificing usefulness. Masking platforms act as a data Shepard, locating data, identify risks and applying protection as information moves in and out of the applications that implement business processes.

Data masking has been around for years but the technology has matured considerably beyond its simple script-based beginnings, most recently advancing rapidly to address customer demand for solutions to protect data and comply with regulations. Organizations tend to scatter sensitive data throughout their applications, databases, documents and servers to enable employees to perform their jobs better and faster. Sensitive data sets are appearing and expanding ever more rapidly. Between worsening threats to that data, and regulatory mandates to 'incentivize' customers into taking data security seriously, the masking market has ballooned in recent years — fourfold in the last three years by our estimates. Much of this has been driven by the continuing need to derive information from a sea of data while address increasing compliance requirements. Of all the technologies available to reconcile these conflicting imperatives, masking offers several unique advantages.

In this paper, *Understanding and Selecting Data Masking Solutions,* we will discuss why customers buy masking products, how the technology works, and what to look for when selecting a masking platform. As with all Securosis research papers, our goal is to help end users get their jobs done. So we will focus on how to help you, would-be buyers, understand what to look for in a product. We will cover use cases at a fairly high level, but we'll also dig into the technology, deployment models, data flow, and management capabilities, to assist more technical buyers understand what capabilities to look for. At the end of this paper we provide a brief guide to features to look for, based on the most common use cases and deployment preferences.

Use Cases

Over the last decade we have consistently heard about three use cases behind the bulk of purchases:

Test Data Creation and Management

This is, by far, the most important reason customers gave for purchasing masking products. When polled, most customers say their #1 use for masking technologies is to produce test and analysis data. They want to make sure employees don't do something stupid with corporate data, like making private data sets public, or moving production data to unsecured test environments. This informal description is technically true as far as customer intent, but fails to capture the essence of what customers look for in masking products. In actuality, masking data for testing and sharing is almost a trivial subset of the full customer requirement; tactical production of test data is just a feature. The real need is for administrative control of the entire data security lifecycle — including locating, moving, managing, and masking data. The mature version of today's simpler use case is a set of enterprise data management capabilities which control the flow of data to and from hundreds of different databases. This capability answers many of the most basic security questions we hear customers ask, such as "Where is my sensitive data?" "Who is using it?" and "How can we effectively reduce the risks to that information?"

Companies understand that good data makes employees' jobs easier. And employees are highly motivated to procure data to help do their jobs, even if it's against the rules. If salespeople can get the entire customer database to help meet their quotas, or quality assurance personnel think they need production data to test web applications, they usually find ways to get it. The same goes for decentralized organizations where regional offices need to be self-sufficient, and organizations needing to share data with partners. The mental shift we see in enterprise environments is to stop fighting these internal user requirements, and instead find a safe way to satisfy the demand. In some cases this means automated production of test data on a regular schedule, or self-service interfaces to produce masked content on demand. Such platforms are effectively implementing a data security strategy for fast and efficient production of test data.

Compliance

Compliance is the second major driver cited by customers for masking product purchases. Unlike most of today's emerging security technologies, it's not just the Payment Card Industry's Data Security Standard (PCI-DSS) driving sales — many different regulatory controls, across various industry verticals, are creating broad interest in masking. Early customers came specifically from finance — but adoption is well distributed across different segments — with strong adoption across retail, telecomm, health care, energy, education, and government. The diversity of customer requirements makes it difficult to pinpoint any one regulatory concern that stands out from the rest. During discussions we hear about all the usual suspects — including PCI, NERC, GLBA, FERPA, HIPAA, Mass 201 CMR 17, and in some

cases multiple requirements at the same time. These days we hear about masking being deployed as a more generic control — customers cite protection of Personally Identifiable Information (PII), health records, and general customer records, among other concerns — but we no longer see every customer focused on one specific regulation or requirement. Now masking is perceived as addressing a general need to avoid unwanted data access, or to reduce exposure as part of an overall compliance posture. It's attractive across the board because it removes sensitive data and thus eliminates risk of data sets being scattered across the organization.

For compliance, masking is used to protect data with minimal modification to systems or processes which use the now masked data. Masking provides consistent coverage across files and databases with very little adjustment to existing systems. Many customers layer masking and encryption together — using encryption to secure data at rest and masking to secure data in use. Customers find masking better at maintaining relationships within databases; they also appreciate that it can be applied dynamically, causing fewer application side effects. Several customers note that masking has supplanted encryption as their principal control, while others are using the two technologies to support each another. In some cases encryption is deployed as part of the infrastructure (intrinsic to SSL, or embedded in SAN), while others employ encryption as part of the data masking process — particularly to satisfy regulations that prescribe encryption. In essence encryption and even tokenization are add-ons to their masking product. But the difference worth noting is that masking provides data lifecycle management — from discovery to archival — whereas encryption is applied in a more focused manner, often independently at multiple different points, to address specific risks. The masking platform manages the compliance controls, including which columns of data are to be protected, how they are protected, and where the data resides.

Production Database Protection

The first two use cases drive the vast majority of market demand for masking. While replacement of sensitive data — specifically through ETL style deployments — is by far the most common, it is not the only way to protect data in a database. For some firms protection of the production database is the main purpose of masking, with test data secondary. But production data generally cannot be removed as it's what business applications rely upon, so it needs to be protected in place. There are many ways to do this, but newer masking variants provide a means of leveraging production applications to provide secured data.

This use case centers around protecting information with fine control over user access, with a dynamic determination of whether or not to provide access — something user roles and credentials for files or databases — are not designed to support. To accomplish this, requests for information are analyzed against a set of constraints such as user identity, calling application, location, time of day and so on. If the request does not meet established business use cases, the user gets the masked copy of the data. The masking products either alter a user requests for data to exclude sensitive information, or modify the data stream before it's sent back to the user.

Keep in mind this type of deployment model is used when a business wants to leverage production systems to provide information access, without altering business systems to provide the finer grained controls or embedding additional access and identity management systems. Dynamic and proxy based masking products supplement existing database systems, eliminating the need for expensive software development or potentially destabilizing alterations. ETL may be the most common model, but in-place and dynamic masking is where we witness the most growth. Customers appreciate the ability to dynamically detect misuse while protecting data; this ability to monitor usage and log events helps detect misuse and the mask ensures copies of data are not provided to suspect users.

Emerging Use Cases

It is worth mentioning a few other upcoming use cases. We have not seen customer adoption of the following use cases, the majority of customers we interviewed were actively investigating the following options.

One model that has gotten considerable attention over the last year couple years is masking data for 'the cloud', or pushing data into multi-tenant environments. The reasons behind customer interest in the cloud varied, but leveraging cost-effective on demand resources or simple an easier was to share data with customers or partners without opening holes into in-house IT systems. It was common amongst those surveyed to include traditional relational databases, data warehouse applications, or even NoSQL style platforms like Hadoop. When asked about this, several companies complained that the common security strategy of "walling off" data warehouses through network segmentation, firewalls, and access control systems, worked well for in-house systems but was unsuitable for data 'outside governance controls'. These customers are looking for better ways to secure *data* instead of relying on infrastructure and (old) topology. Research into masking to enforce policy in multi-tenant environments was atop customer interest.

We were surprised by one use case: masking to secure streams of data — specifically digitized call records or XML streams of — which came up a number of times, but not enough yet to constitute a trend. Finally, we expected far more customers to mention HIPAA as a core requirement in order to mask in order to secure complex data sets. Only one firm citied HIPAA driving masking. While dozens of firms mentioned patient health records as a strong area of interest, it's not yet driving sales.

Defining Data Masking

'Masking' has become a 'catch-all' phrase used to describe masks, the process of masking, and a generic description of different obfuscation techniques such as tokenization or hashing. We will offer a narrower definition of masking, but first, a couple basic terms with their traditional definitions:

- **Mask:** Similar to the traditional definition, meaning a *facade* or a *method of concealment*, a data mask is a function that transforms data into something new but similar. Masks *must* conceal the original data and *should not* be reversible.
- **Masking:** The conversion of data into a masked form — typically sensitive data into non-sensitive data of the same type and format.
- **Obfuscation:** Hiding the original value of data.

Data Masking Definition

All data masking platforms replace data elements with similar values, optionally moving masked data to a new location. Masking creates a proxy data substitute which retains *part* of the value of the original. The point is to provide data that looks and acts like the original data, but which lacks sensitivity and doesn't pose a risk of exposure, enabling use of reduced security controls for masked data repositories. This in turn reduces the scope and complexity of IT security efforts. Masking must work with common data repositories, such as files and databases, without *breaking* the repository. The mask should make it impossible or impractical to reverse engineer masked values back to the original data without special *additional* information, such as a shared secret or encryption key.

Five Laws of Data Masking

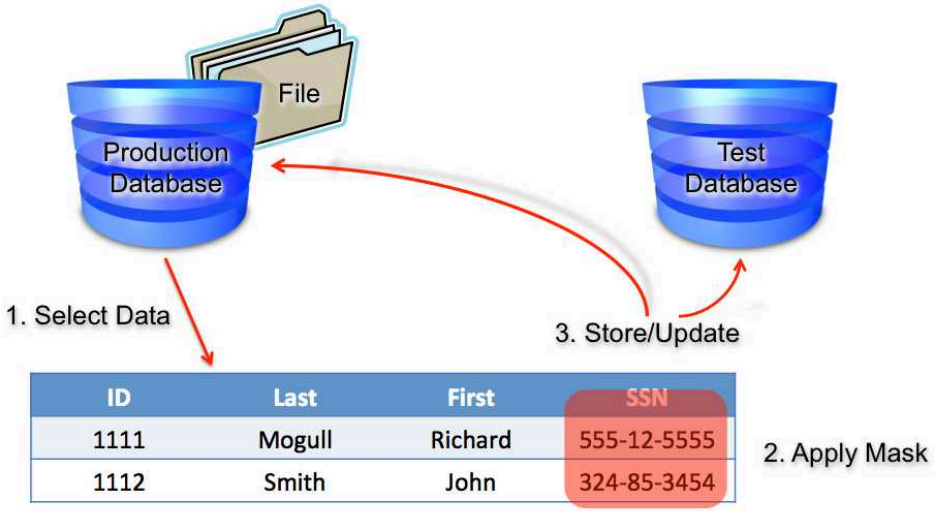
1. *Masking must not be reversible*. However you mask your data, it should *never* be possible to use it to retrieve the original sensitive data.
2. *The results must be representative of the source data*. The reason to mask data instead of just generating random data is to provide non-sensitive information that still resembles production data for development and testing purposes. This could include geographic distributions, credit card distributions (perhaps leaving the first 4 numbers unchanged, but scrambling the rest), or maintaining human readability of (fake) names and addresses.
3. *Referential integrity must be maintained*. Masking solutions must not disrupt referential integrity — if a credit card number is a primary key, and scrambled as part of masking, then all instances of that number linked through key pairs must be scrambled identically.

> The five laws help capture the essence of data masking, and how it helps address security and compliance.

4. *Only mask non-sensitive data if it can be used to recreate sensitive data.* It isn't necessary to mask everything in your database, just those parts that you deem sensitive. But some non-sensitive data can be used to either recreate or tie back to sensitive data. For example, if you scramble a medical ID but the treatment codes for a record could only map back to one record, you also need to scramble those codes. This attack is called inference analysis, and your masking solution should protect against it.

5. *Masking must be a repeatable process.* One-off masking is not only nearly impossible to maintain, but it's fairly ineffective. Development and test data need to represent constantly changing production data as closely as possible. Analytical data may need to be generated daily or even hourly. If masking isn't an automated process it's inefficient, expensive, and ineffective.

The following graphic illustrates a typical masking process. We select data from a file or database, mask the sensitive data, and then either update the original file/database, or store the masked data to a new location.



Masks

'Masking' is a generic term which encompasses several process variations. In a broad sense data masking — just 'masking' for the remainder of this paper — encompasses the collection of data, obfuscation of data, storage of data, and often movement of the masked information. But 'mask' is also used in reference to the masking operation itself; it's *how* we change the original data into something else. There are many different ways to obfuscate data depending on its type, each embodied by a different function, and each suitable for different use cases. It might be helpful to think of masking in terms of Halloween masks: the level of complexity and degree of concealment both vary, depending on the effect desired by the wearer.

The following is a list of common data masks used to obfuscate data. These are the standard options masking products offer, with the particular value of each option:

- **Substitution:** Substitution is simply replacing one value with another. For example, the mask might substitute a person's first and last names with names from a random phone book entry. The resulting data still constitutes a name, but has no logical relationship with the original *real* name unless you have access to the original substitution table.

- **Redaction/Nulling:** This substitution simply replaces sensitive data with a generic value, such as '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, but the output provides little or no value.

- **Shuffling:** Shuffling is a method of randomizing existing values *vertically* across a data set. For example, shuffling individual values in a salary column from a table of employee data would make the table useless for learning what any particular employee earns without changing aggregate or average values for the table. Shuffling is a common randomization technique for disassociating sensitive data relationships (*e.g.,* Bob makes $X per year) while preserving aggregate values.

- **Blurring:** Taking an existing value and alter it so that the value falls randomly within a defined range.

- **Averaging:** Averaging is an obfuscation technique where individual numbers are replaced by a random value, but across the entire field, the average of these values remains consistent. In our salary example above, we could substitute individual salaries with the average across a group or corporate division to hide individual salary values while retaining an aggregate relationship to the real data.

- **De-identification:** A generic term for to any process that strips identifying information, such as who produced the data set, or personal identities within the data set. De-identification is important for dealing with complex, multi-column data sets that provide sufficient clues to reverse engineer masked data back into individual identities.

- **Tokenization:** Tokenization is substitution of data elements with *random* placeholder values, although vendors overuse the term 'tokenization' for a variety of other techniques. Tokens are non-reversible because the token bears no logical relationship to the original value.

- **Format Preserving Encryption:** Encryption is the process of transforming data into an unreadable state. Unlike the other methods listed, the original value can be determined from the encrypted value, but can only be reversed with special knowledge (the key). While most encryption algorithms produce strings of arbitrary length, format preserving encryption transforms the data into an unreadable state while retaining the format (overall appearance) of the original values. Technically encryption violates the first law of data masking, but some customers we interviewed use encryption and masking side-by-side, we wanted to capture this variation.

Masking Constraints

Each type of mask excels in some use cases, offering concealment of the original while preserving some critical value or aspect of the original. Each incurs a certain amount of computational overhead. For example, it's much easier to replace a phone number with a series of 'X's than it is to encrypt it. In general, applying masks (running data through a masking function) is straightforward. However, various constraints make generating quality data masks much more complicated than it might seem at first glance. Retention of some original value makes things much more difficult and requires considerable intelligence from masking products. The data often originates from, or is inserted into, a relational database; so care is required with format and data types, and to ensure that the mask output satisfies integrity checks. Typical customer constraints include:

> Masking data is easy. Preserving value and maintaining integrity while protecting data is difficult; you must verify that masks provide security and maintain value.

•**Format Preservation:** The mask must produce data with the same structure as the original data. This means that if the original data is 2-30 characters long, the mask should produce data 2-30 characters long. A common example is dates, which must produce numbers in the correct ranges for day, month, and year, likely in a particular text format. This means a masking algorithm must identify the 'meaning' of source data such as "31.03.2012", "March 31, 2012", and "03.31.2012", and generate a suitable date in the same format.

•**Data Type Preservation:** With relational data storage it is essential to maintain data types when masking data from one database to another. Relational databases require formal definition of data columns and do not tolerate text in number or date fields. In most cases format-preserving masks implicitly preserve data type, but that is not always the case. In certain cases data can be 'cast' from a specific data type into a generic data type (*e.g.,* `varchar`), but it is essential to verify consistency.

- **Gender preservation:** When substituting names, male names are only substituted with other male names, and similarly female with only female names. This is of special importance amongst some cultures.

- **Semantic Integrity:** Databases often place additional constraints on data they contain such as a LUHN check for credit card numbers, or a maximum value on employee salaries. In this way you ensure both the format and data type integrity, but the stored value makes sense in a business context as well.

- **Referential Integrity:** An attribute in one table or file may refer to another element in a different table or file, in which case the reference must be consistently maintained. The reference augments or modifies the meaning of each element, and is in fact part of the value of the data. Relational databases optimize data storage by allowing one set of data elements to 'relate', or refer, to another. Shuffling or substituting key data values can destroy these references (relationships). Masking technologies must maintain referential integrity when data is moved between relational databases, or cascading values from one table to another. This ensures that loading the new data works without errors, and avoids breaking applications which rely on these relationships.

- **Aggregate Value:** The total and average values of a masked column of data should be retained, either closely or precisely.

- **Frequency Distribution:** In some cases users require random frequency distribution, while in others logical groupings of values must be maintained or the masked data is not usable. For example, if the original data describes geographic locations of cancer patients by zip code, random zip codes would discard valuable geographical information. The ability to mask data while maintaining certain types of patterns is critical for maintaining the value of masked data for analytics.

- **Uniqueness:** Masked values must be unique. As an example, duplicate SSNs are not allowed when uniqueness is a required integrity constraint. This is critical for referential integrity, as the columns used to link tables must contain unique values.

There are many other constraints, but these are the principal integrity considerations when masking data. Most vendors include masking functions for all the types described above in the base product, with the ability to specify different data integrity options for each use of a mask. And most include built-in bundles of appropriate masks for regulatory requirements to streamline compliance. We will go into more detail when we get to use cases and compliance.

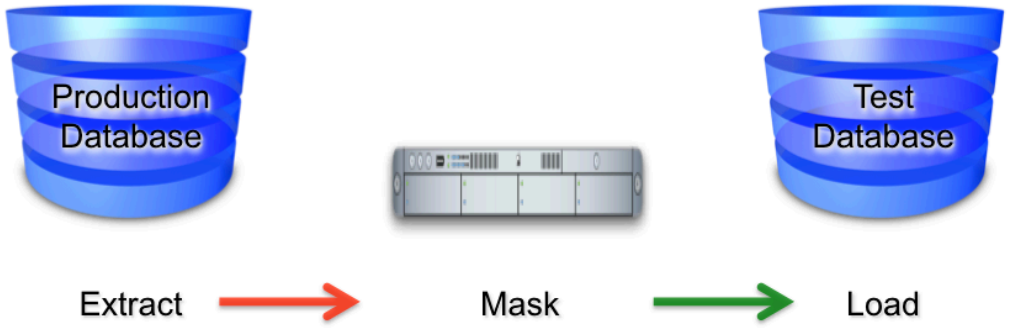
How Masking Works

In this section we will describe how masking works, focusing on data movement and manipulation. We'll start by describing the different masking models, how data flows through various topologies, and advantages and disadvantages of each. Then we will illustrate movement of data between different sources and destinations to explain orchestration managed by the masking platform and the data management capabilities embedded into masking systems. Masks can be used with many different types of data repositories, including relational and non-relational databases, files, applications, and data streams, but for simplicity we will stick to relational databases.

Deployment Models

ETL

When most people think about masking they think about ETL, short for Extract, Transform, and Load — a concise description of the classic (and still most common) masking process. Also called 'static' masking, ETL works on an export from the source repository. Each phase of ETL is typically performed on separate servers: A source data repository, a masking server that orchestrates the transformation, and a destination database. The masking server connects to the source, retrieves a copy of the data, applies the mask to specified columns of data, and then loads the result onto the target server. This process may be fully automated, completely manual, or partially automated.



Let's examine the steps in greater detail:

1. **Extract:** The first step is to extract the data from a storage repository — most often a database. The extraction process may select the entire database, or gather a sub-set of the data based upon some selection criteria. The extracted data is often formatted to make it easier to apply the mask. For example, extraction can be performed with a simple `SELECT` query against a database, filtering out unwanted rows and formatting columns. This form of

filtering — usually called sub-setting — is an important aspect of data management. Results may be streamed directly to the masking application for processing or dumped into a file — such as a comma-separated `.csv` or tab-separated `.tsv` file. The extracted data is then *securely* transferred, as an encrypted file or over an encrypted SSL connection, to the masking platform.

2. **Transform:** The second step is to apply the data mask, transforming sensitive production data into a safe approximation of the original content. Masks are almost always applied to what database geeks call "columnar data" — which simply means data of the same type is grouped together. For example, a database may contain a 'customer' table, where each customer entry includes a Social Security Number (SSN). These values are grouped together into a single column, in files and databases alike, making it easier for a masking application to identify which data to mask. The masking application parses through the data, and for each column of data to be masked, it replaces each entry in the column with a masked value.

3. **Load:** In the last step masked data is loaded into a destination database. The masked data is copied to one or more destination databases, where it is loaded back into tables. The destination database does not contain sensitive data, so its security and audit requirements are lower than those of the original database with unmasked sensitive data.

ETL is the most generic and flexible of masking approaches. The *logical* ETL process flow is implemented in dedicated masking platforms, data management tools with integrated masking and encryption libraries, embedded database tools — all the way down to home-grown scripts. We see all these used in production environments, with the level of skill and labor required increasing as you work down the chain. In more complex environments, such as with Oracle ERP systems, masking may interfere with application functionality. Still in other test environments, where the target database is fantastically complex, the process is fragile as the mask may break relational constraints. Because there are many different environments that masking serves, and different customer requirements on data quality and performance, ETL is not always the right approach. Here are some alternative masking models and processes.

In-place Masking

In some cases you need to create a masked copy *within* a database. For example when production data is discovered on a test system, the data may be masked *in place* without being moved at all. There are two basic approaches for in place masking: Copying an entire *database* and then masking, or copying all of the *data* into a new database prior to masking. In the first case a copy of the original production database — basically a copy of the original database data files — is moved then and then masked before that database is available for use. In the other case the production data is moved unchanged (securely!) into another system, and *then* masked at the destination. This is usually done by creating an 'export' file from the production server and then load that into the destination database server. Both these variations are called "in-place masking" because they skip one of the movement steps in E-T-L. The masks are applied as before, but inside the database.

In-place Masking

There are very good reasons to mask in-place. The first is to take advantage of databases' facility with management and manipulation of data. They are incredibly adept at data transformation, offering *very* high masking performance. Leveraging built-in functions and stored procedures can speed up the masking process because the database has already parsed the data. Masking data in-place can also be used for partial — or iterative — updates to existing data files, further speeding up the masking process.

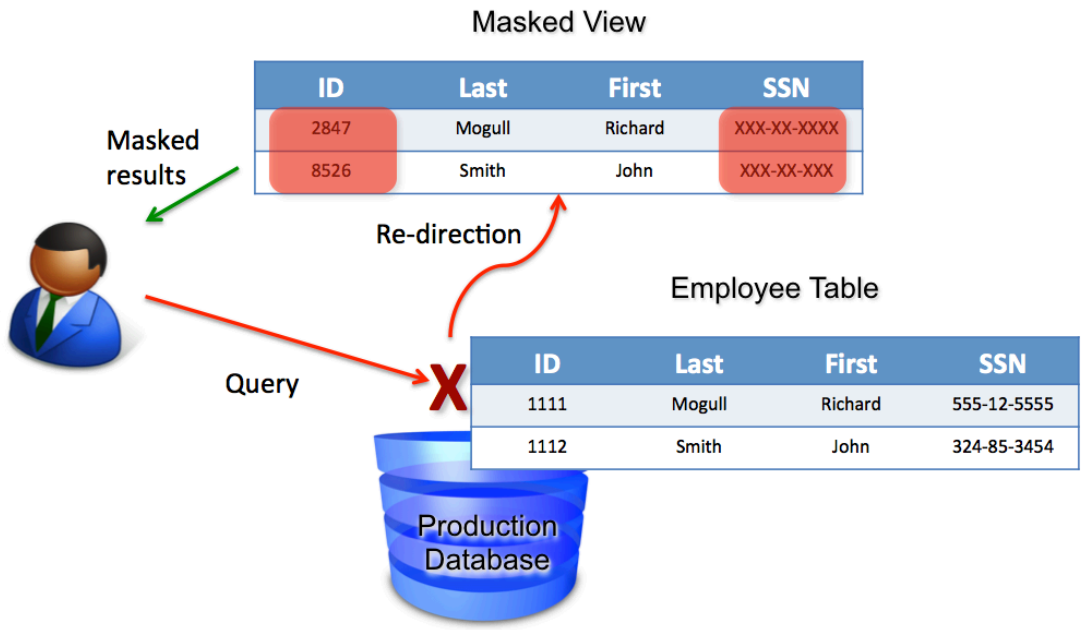
Another principle reason for in-place masking is it lessens the likelihood that complex relationships, functions or stored procedures failed to import properly, thus retaining full integrity of the production database platform. Being able to ensure an accurate copy — in terms of data, structure and functionality — is key to ensure a test environment properly mimics the production environment.

However, there is a risk with this model that sensitive data will still reside in the destination copy — such as in logs, transaction or temporary files — that may expose sensitive data. In addition, as most test environments have poor security, masking *must* be done *before* the destination repository is available for use. This means keeping the database 'offline' until all traces of sensitive data are overwritten. Finally, there is also the risk that the sensitive data — or the database files — will be exposed during the move from production to test environments. It's critical when looking at in-place masking options to ensure the masking provider protects data as it is moved an masked to ensure information is not exposed.

To address these concerns in-place masking solutions have evolve into a form of a hybrid approach, masking by local software agents prior to database insertion, or before the database is 'opened' to ensure that sensitive data is never available on the test system. When done with care, in-place masking mitigates performance concerns of impacting the production server and does not expose information in test environments.

Dynamic Masking: View-based Masks

View-based masking refers to storage of *both* production data and a corresponding masked version in the same database. Which copy of data a user sees is determined by the database at the time of the request as the mask is *triggered* dynamically, by the database, depending upon the requesting user credentials. When a user requests data their credentials and session information are examined by the masking platform. Authorized users receive unmasked production data. Users without authorization for sensitive information, or using unapproved applications, or who trigger some other security filter which marks the connection as suspect, receive masked data. The determination of which data to provide is made in real time. The criteria for presenting masked data must be specified in advance, and alerts can be transmitted when the masked data is invoked.



Dynamic view-based masking is either implemented by a) database code to inspect & possibly alter the inbound query to select either original or masked data, and b) a database 'view', which is nothing more than a virtual table. While the view is structurally similar to a real table, it contains only masked data. View-based masking requires a plug-in, trigger, or code residing within the database to divert suspect queries from the unmasked data.

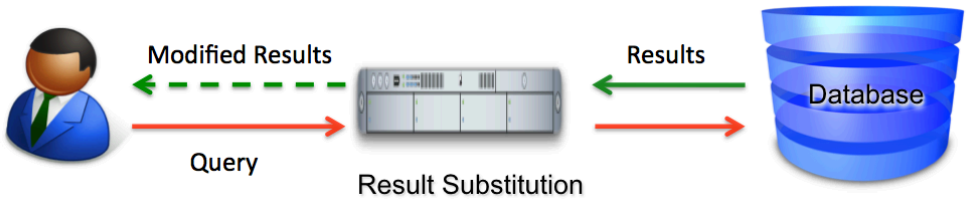
Dynamic masking is an excellent way to protect data in web application environments where users cannot be perfectly authenticated, without costly recoding of applications. Additionally, view-based masking enables testing of production applications in a production environment, by only exposing masked data to less-privileged test users. For firms with very large production databases, which are not worried about extra overhead on production database servers, this model provides an excellent approximation of a production environment for testing.

But of course there are drawbacks. Most customers, especially those running on legacy systems, don't want this burden on production database servers. Additionally, database masking does not prevent data leakage — from logs, data files, or archives. This is no worse than a normal production environment, but it is important to remember that all these other types of secondary data are targets for attackers. Unfortunately, view-based masking is currently only available for relational databases — not file systems or other platforms. Finally, should the view *fail* in some way, unauthorized queries might expose sensitive data or fail completely. Behavior of the masking solutions must be tested under adverse conditions.

When we used to refer to 'dynamic masking', it was the sole domain of view-based masks described above. During our customer surveys, most referred to in-line and proxy-based solutions — which we discuss below — as 'dynamic' as well. As all these variations apply masks *after* examining user requests and session based attributes, they do all act dynamically. To keep things consistent, and avoid confusing both new and existing customers, we'll categorize all of these options as 'dynamic' and distinguish between the different deployment models as necessary.

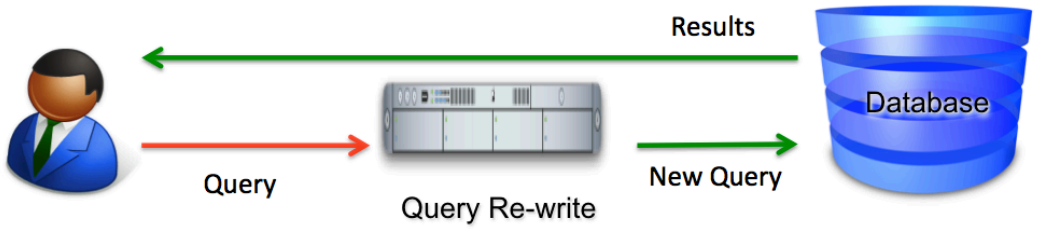
Dynamic Masking: Proxy-based Masks

A more recent dynamic data masking model is where data is masked by an in-line service as it goes into, or comes out from, a database. The results of a user query are intercepted and masked *on the fly*, with masked results *transparently* substituted before being returned to 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. This differs from view-based masking in that it occurs outside the data repository and is available for non-relational systems. The proxy may be an agent on top of the database, an appliance deployed 'inline' between users and data, to force all requests through the proxy. The huge advantage is data protection is enabled without need to alter the database; there are none of the additional programming and quality assurance validation processes.



Another recent advance in dynamic data masking is query substitution. In this variation the masking platform is smart enough to recognize a request for sensitive data. Such queries are intercepted and re-written to select information from a different (masked) column. For example, a query to select credit card numbers might be modified to request tokenized values of credit card numbers instead. The interception model is very flexible; the `SELECT` statement can be directed to

an alternative 'view', to a file of pre-generated masked values, redact sensitive content or even to a 'join' against another database.



One downside of dynamic masking is the need to create policies which account for *all* the different cases — queries for those of you comfortable with database terminology — which should return masked data. Automated discovery tools and pre-built policies are critical to speeding up deployment and understanding what data is at risk. Regardless of how good the pre-built policy set it, expect to invest time in creating and customizing policies *for your environment*. As with any other in-line service it is important to consider failover, performance, and scalability prior to production deployment.

Data Sources

Technically, the source of data for *extraction* — just like the destination to *load* with masked data — could be just about anything. It can be a single file or multiple files, a quasi-relational database, indexed/ISAM files, a NoSQL database, a document management system, optically transcribed (OCR) imagery, or even an ephemeral data streams such as a digitized telephone conversation. As long as the masking platform is able to identify which data to mask, just about anything is possible.

- **Databases:** The most common data sources are relational database platforms. These databases require strict data definitions, ('metadata') for every column of every table — this rigorous structure is extremely useful for locating and masking sensitive data. Insertion and extraction are commonly handled through SQL queries and database export & load functions. SQL queries offer data in human-readable format, and easily accommodate filtering and formatting of information. On the other hand, queries may place unacceptable load on production databases, or take an unacceptable amount of time to complete. Exports and imports are much less flexible, but masking providers support native export/import formats to save time and reduce server impact. Masking systems connect either though native ODBC/JDBC connections, or by issuing queries directly to the database.

- **Non-relational/Indexed Databases:** Mainframe databases, data warehouses, business intelligence, CRM, and other large analytic systems are increasingly run on non-relational platforms to improve scalability and performance. These platforms behave much like relational databases but lack some of their formal data structures; this makes discovery less accurate and masking more difficult. To address the growing popularity of these platforms, and the difficulty of monitoring and encrypting them, masking can provide a preventative security control — removing sensitive data while preserving analytic and data-mining capabilities. Again, masking systems connect though native ODBC/JDBC connections, or by issuing queries directly to the database.

- **Files:** Files are sources of sensitive information as frequently as databases. Corporate IP, financial spreadsheets, medical documents, and the like are stored on file servers; they can all contain sensitive data which must be protected from hackers and disgruntled employees. Keep in mind that we are using a broad definition of the word 'file', including data in digitized voicemail or structured XML documents as well common Word and text documents. Some organization choose Data Loss Prevention or File Activity Monitoring solutions to monitor and protect files, but masking provides a simple, low-cost mechanism to proactively detect and remove sensitive data from unauthorized locations. Files can be masked in-place as data is discovered in unauthorized locations, or masked files can be generated periodically to provide non-sensitive datasets to multiple audiences. Files are typically accessed through software agents installed at the OS or file system layer, which can intelligently provide either direct or masked data. However, like in-place masking of databases, you need to verify that temporary files don't leak data.

- **NoSQL Databases:** Hadoop, Mongo, Cassandra, and other NoSQL database platforms are known for their scalability and powerful search and analysis capabilities. NoSQL platforms lack the formalized data integrity checking of relational databases, which makes data insertion much faster. Unfortunately they also lack fine-grained security controls. Their lack of formal data type definitions makes discovery of sensitive data much more difficult, and complicates masking of such data once found. It is becoming increasingly common to substitute valuable information for masked data *before it is loaded* into such repositories, so compromise of these platforms is not catastrophic.

- **Document Management:** Collaboration and document management applications are to key productivity in just about every enterprise, and they contain large amounts of sensitive data. Masking capabilities that support collaboration software provide similar benefits as in relational databases, for unstructured document and messaging systems. Integrated masking for collaboration systems is still new, and commercial offerings are not full-featured, but they can help discover and catalog sensitive data, and in some case can replace documents with masked variants.

Technical Architecture

We mentioned in the introduction that masking has come a long way since the early days of simple scripts run by a system or database administrator. Back then users connected directly to a database or ran scripts from the console, and manually moved files around. Today's platforms proactively discover sensitive data and manage policies centrally, handling data security and distribution across dozens of different types of information management systems, automatically generating masked data as needed for different audiences. Masking products can stand alone, serving disparate data management systems simultaneously, or be embedded as a core function of a dedicated data management service.

Base Architecture

- **Single Server/Appliance:** A single appliance or software installation that performs static ETL data masking services. The server is wholly self-contained — performing all extraction, masking, and loading from a single location. This model is typically used in small and mid-sized enterprises. It can scale geographically, with independent servers in regional offices to handle masking functions, usually in response to specific regional regulatory requirements.
- **Central Server Architecture:** Multiple masking servers, centrally located and managed by a single management server, are used primarily for production and management of masked data for multiple test and analytics systems.
- **Distributed:** This option consists of a central management server with remote agents/plug-ins/appliances that perform discovery and masking functions. The central server distributes masking rules, directs endpoint functionality, catalogs locations and nature of sensitive data, and tracks masked data sets. Remote agents periodically receive updates with new masking rules from the central server, and report back sensitive data that has been discovered, along with the results of masking jobs. Scaling is achieved by pushing processing load out to the endpoints.
- **Proxy/Bridge Cluster:** One or more appliances or agents that dynamically mask streamed content, typically deployed in front of relational databases, to provide proxy-based data masking. This model is used for real-time masking of non-static data, such as database queries or for loading into NoSQL databases. Multiple appliances provide scalability and failover. This may be deployed in a single-tier or two-tier architecture.

Appliances, software, and virtual appliance options are all available. But unlike most security products, where appliances dominate the market, masking vendors generally deliver their products as software. Windows, Linux, and UNIX support are all common, as is support for many types of files and relational databases. Support for virtual appliance deployment is common among the larger vendors but not universal, so inquire about availability if that is important to your IT service model.

A key masking evolution is the ability to apply masking policies across different data management systems (file management, databases, document management, etc.) regardless of platform type (Windows vs. Linux vs. ...). Modern

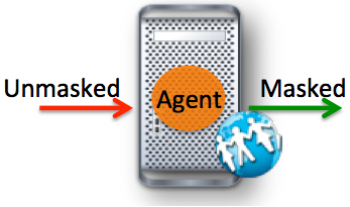
masking platforms are essentially data management systems, with policies set at a central location and applied to multiple systems through direct connections or remote agents. As data is collected and moved from point A to point B, one or more data masks are applied to one or more 'columns' of data.

Deployment and Endpoint Options

Masking architecture is conceptually simple but there are many different deployment options, each particularly suited to protecting one or more data management systems. Masking technologies must work on static data copies, live database repositories, and dynamically generated data (streaming data feeds, application generated content, *ad hoc* data queries, etc.), so a wide variety of deployment options are available to accommodate these different data management environments. Most companies deploy centralized masking servers to produce safe test and analytics data, but vendors offer the flexibility to embed masking directly into other applications and environments where large-footprint masking installations or appliances are unsuitable. The following is a sampling of common deployments used for remote data collection and processing.

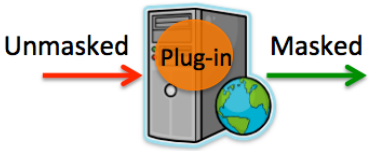
Agents

Agents are software components installed on a server, usually the same server that hosts the data management application. Agents can be as simple or advanced as the masking vendor cares to make them. They can be nothing more than a data collector, sending data back to a remote masking server for processing, or might mask as data is collected. In the latter case the agent masks data as it is received — either completely in memory or from a temporary file. Agents can be managed remotely by a masking server or directly by the data management application, effectively extending data management and collaboration system capabilities (*e.g.,* MS SharePoint, SAP). One advantage of endpoint agents over in-database stored procedures (which we will describe in a moment) is that all traces of unmasked data can be destroyed. Whether by masking in 'ephemeral' memory, or by ensuring temporary files are overwritten, sensitive data can be prevented from leaking through temporary storage. Agents do consume local processor cycles, memory, and storage — a significant issue for legacy platforms — but only a minor consideration for virtual machines and cloud deployments.



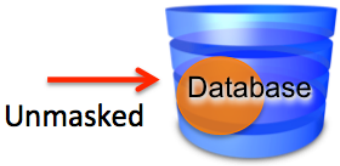
Web Server Plug-ins

These plug-ins are a type of agent installed as web application services as part of a web application stack to support the local application which manages data. Plug-ins are an efficient way to transparently implement masking within existing application environments, acting on the data stream before it reaches the application or extending the application's functionality through Application Programming Interface (API) calls. This enables plug-ins to be managed by a remote masking server through web API calls, or directly by a larger data management system. Deployment in the web server environment gives plug-ins the flexibility to perform ETL, in-place, and proxy masking. Proxy agents are most commonly used to decipher XML data streams, and increasingly common for discovering and masking data streams prior to loading into "Big Data" analytic databases.



Stored Procedures and Triggers

A stored procedure is basically a small script that resides and executes inside a database. A trigger is a small function that manipulates data as records are inserted and updated in a database. Both these database features have been used to implement data masking. Stored procedures and triggers take advantage of the database's native capabilities to seamlessly mask data inside the database. They can be used to mask 'in-place', replacing sensitive data, or periodically to create a masked 'view' of original data. Triggers are not optimal because they are relatively expensive in terms of computational overhead, they don't work with certain types of applications, and both triggers and stored procedures can leak sensitive information through change logs if care is not taken. Still, with some databases types, these native capabilities remain viable.



ODBC/JDBC Connectors

ODBC stands for Open Database Connector, and JDBC for Java Database Connectivity; each offers a generic programmatic interface to any type of relational database. These interfaces are used by applications that need to work directly with the database to query and update information. You can think of it something like a telephone call between the database and the masking platform: The masking server establishes its identity to the database, asks for information, and then hangs up when the data has been transferred. The masking platform can issue *any* database query, so it's ideal for pulling incremental changes from a database and filtering out unneeded data. This is critical for reducing the size of the data set, reducing the total amount of processing power required to mask, and making extraction and loading operations as efficient as possible. These interfaces are most often used to retrieve information for ETL-style deployments, but are also leveraged in conjunction with stored procedures to implement in-place masking. Standard interfaces like ODBC/JDBC are of critical importance when managing hundreds of databases from a single location, and reduce management costs of maintaining agents and plug-ins across all of your systems.

Platform Management

Beyond the basics of data collection and transformation, the management interface for a masking product is critical to customer satisfaction. Central management is one of the core additions that transforms masking from a simple tool into an enterprise data security platform, and what you use to orchestrate masking operations. Central management is not new, but capabilities and maturity, of these interfaces are evolving rapidly to meet new customer requirements. As the management interface is what you use day in and day out, it's critical that it be intuitive and easy to use. A bad management interface makes simple tasks difficult, increases the likelihood of mistakes, and dissuades administrators from keeping the system up to date.

Central Management

This is the proverbial "single pane of glass" for management of data, policies, data repositories, and task automation. The user interface is how you interact with data systems and control the flow of information. A good UI can simplify your job, but a bad one will make you want to avoid the product! Management interfaces have evolved to accommodate both IT management and non-technical stakeholders alike — allowing them to set policy, define workflows, understand risk, and manage where data goes. Some products even provide the ability to manage endpoint agents. Keep in mind that each masking platform has its own internal database to store policies, masks, reports, user credentials, and other pertinent information; some also offer visualization technologies and dashboards to help see what exactly is going on with your data. We offer a list of management features to consider when evaluating the suitability of a masking platform:

- **Policy Management:** A policy is nothing more than a rule on how sensitive data is to be treated. Policies usually consist of a data mask and a data source to apply it. Every masking platform comes with several predefined masks, as well as an interface to customize them to your needs. But policy interfaces go one step further, associating a mask with a data source. Some platforms even allow a policy to be automatically applied to specific data types, such as credit card numbers, regardless of source or destination. Policy management is typically simplified with predefined policy sets, as we will discuss below.

- **Discovery:** For most customers discovery has become a must-have feature — not least because it is essential for regulatory compliance. Data discovery is an active scan to first find data repositories, and then scan them for sensitive data. The discovery process works by scanning files and databases, matching content to known patterns (such as 9-digit Social Security numbers) or metadata (data that describes data structure) definitions. As sensitive data is discovered, the discovery tool reports on the locations and types of sensitive data found. Once data is discovered there are many options for what to do next. The report can be sent to interested parties, archived for compliance, or even fed back into the masking product for automatic policy enforcement. Discovery results can be used to build a catalog of metadata, physically map locations within a data center, and even present a risk score based on location and data type. Discovery can be tuned to look in specific locations, refined to look for as few or as many data types as the user is interested in, and automated to find preselected patterns on a regular schedule.

- **Credential Management:** Selection, extraction, and discovery of information from different data sources all require *credentialed access* (typically a user name and password) to the file or database in question. As it's infeasible to expect users to provide a user name and password before we begin the masking process, but we need to supply credentials, as a masking task is performed. The masking platform needs to either securely store credentials, or use credentials from an access management system like LDAP or Active Directory and automatically supply them as needed.

- **Data Set Management:** For managing test data sets, as well as for compliance, you need to track which data you mask and where you send it. This information is used to orchestrate moving data around the organization — managing which systems get which masked data, tracking when the last update was performed, and so on. As an example, think about the propagation of medical records: an insurance company, a doctor's office, a clinical trial organization, and the federal government, all receive *different* subsets of the data with different masks applied according to which information each needs. This is the core function of data management tools, many of which have recently added masking capabilities. Similarly, masking vendors have added data management capabilities in response to customer demands for complex data orchestration. The formalization of how data sets are managed is also key for both automation and visualization, as we will discuss below.

- **Data Sub-setting:** Large enterprises often apply masking across hundreds or thousands of databases. In these cases it's essential to be as efficient as possible to avoid overtaxing databases or saturating networks with traffic. People who manage data define the smallest data subset possible that still satisfies application testers' needs for production quality masked data. This involves cutting down the number of rows exported/viewed, and possibly reducing the number of columns. Defining a common set of columns also helps use a single masked data set for multiple environments, reducing the computational burden of creating multiple masked clones.

- **Automation:** Automation of masking, data collection, and distribution tasks are core functions of every masking platform. Automated application of masking policies and integration with third party systems that rely on masked data drastically reduces workload. Some systems offer very rudimentary automation, such as UNIX `cron` jobs, while others include complex features for managing remote jobs and coordinating agents to perform remote tasks. A thorough evaluation of vendor automation is very important because high-quality automation is key to reducing management time; secondary benefits of automation include segregation of duties and distributed management.

- **Reporting:** As with most security platforms, especially those which support compliance regulations, you need to demonstrate that security controls are in place. Reporting is a key feature for audit and compliance, providing auditors both with an explanation of masking controls, and evidence that shows those masks are being applied to key data sets. Some platforms offer basic reporting with the ability to integrate with third-party tools, while others build more elaborate capabilities into the product. Most supply pre-built compliance reports out of the box, used for demonstrating that controls are in place and performing as specified. These can be as simple as reporting when masking tasks are successful or fail, but often include specific information on the location and type of data, where test data sets were sent, and when masks were applied.

Advanced Features

We offer a list of advanced masking functions to give you an idea of how different vendors are trying to differentiate their products, and to provide an idea of where these products are heading. All masking products provide basic functionality, but the basics are insufficient for today's principal use cases. Advanced features and management interfaces differentiate these products and are likely to drive your choice.

- **Advanced Masking:** As masking is applied to large data sets, especially those with complex relationships between different data elements, it is essential that these relationships remain after the data has been masked. For example with patient health data — where location, treatment, and condition have a critical multi-column relationship with each other — the masking process must preserve this *n*-way relationship. Randomizing dates within a timeline is another example, where each date must be randomized while preserving the order of events relative to each other. The ability to customize data masks to maintain complex relationships is becoming more common, and you can tell which verticals a masking vendor serves by which specific industries they include complex masks for.

- **Pre-built Masks:** Prepackaged masks, built specifically to satisfy regulatory requirements such as PCI-DSS, FINRA, PII, and HIPAA are now common. Prepackaged catalogs of meta-data, data locations, and masks for commercial off-the-shelf applications are included in some of the masking platform bundles as well. These help users perform common tasks to make it easier to deploy products and help ensure the appropriate type of protection is applied.

- **Visualization:** Dashboards, network diagrams that illustrate the presence of sensitive data on servers, and flow maps of how data is being moved, are all excellent tools for understanding risks to information. These types of visualization tools are just making their way into masking products. Understanding where data is stored, what applications access it, and the risks it's exposed to along the way, are all very helpful for setting up your data security program. Visualization maps are typically populated from discovery scans, and may be cross-referenced with risk scores to highlight critical systems to help IT management and security practitioners make better decisions on how to mask or apply other security controls to protect data.

- **Diff & Deduplication:** For efficiency, it makes sense to only mask the portion of a data set that has not already been masked. For example, when creating test data, you would mask and export only *new* rows of a database. Partial data set masking is becoming a key feature of some very large systems, such as Teradata and NoSQL databases, where efficiency and speed are critical to keeping pace with the flow of data. This can be accomplished in a variety of ways, from database queries that select only recently modified rows — for both ETL and in-place masking — to more complex features that store previous data sets and mask new additions.

- **Workflow Integration:** Some masking platforms integrate with workflow systems, reporting both new discoveries of sensitive data and creation of masked data sets. This helps compliance groups know their reports have been generated and application testers know their data has been updated, and alerts security teams to sensitive data in new (and possibly unapproved) locations.

- **Encryption & Key Management:** Some systems provide the capability to encrypt data within masked data sets, providing the same format preservation capabilities as masking while allowing recovery of the original content when needed. This is provided through Format Preserving Encryption, usually integrated from a third-party service provider. Encryption does not provide full flexibility in application of masks, and great care must be taken in management of encryption keys, but the abilities of FPE can be extremely useful.

- **Validation:** Validation is verification that data has been effectively masked. Some data sets contain corrupt data which cannot be successfully masked, and other issues can interfere with normal operation, so some masking providers offer validation capabilities to confirm success and detect failure. Validation options verify that the dataset is actually masked, and can differentiate real from masked data. This feature is valuable for ensuring critically sensitive data is not exported without appropriate masking, and helps to satisfy compliance mandates such as the 'distinguishability' requirements of PCI-DSS.

- **Cloud Compatibility:** Cloud computing poses several challenges to masking platforms — including network addressing, identity management, and discovery. But cloud environments present a large and new opportunity for masking vendors, as customers push increasing quantities of data into the cloud. The cloud is especially attractive for Big Data analytics, fast provisioning of test systems, and hybrid data center deployments. And masking is a natural fit for firms that need to protect sensitive data before moving data into multi-tenant environments. Currently compatibility with cloud service models (including SaaS, PaaS, and IaaS) is rare, but we see the first features for NoSQL data management and integration with SaaS.

Buyer's Guide

The following is a simple guide to help customers define what they need a masking platform to do and evaluate how well various products meet those requirements. In our experience understanding your requirements and setting expectations amongst internal stakeholders have just as much influence on product and project satisfaction as product performance. We start with a look at requirements gathering and a discussion of the difficulty (and importance) of getting all the stakeholders to agree on a set of use cases and priorities. We also offer guidance on avoiding pitfalls and vendor BS. Of course you still need to ensure that your requirements are identified and prioritized *before* you start testing, but the process with masking technologies is a bit less complicated than with other technologies.

Once you have clearly documented your requirements the field of potential vendors dwindles rapidly. Customer requirements are narrowly defined along a few principal use cases (test data management, compliance, and database security), so most masking platforms focus their solutions along these lines. This means customers that need ETL are usually focused on ETL, and are not looking for proxy-based solutions. Only a couple full-featured platforms provide both the necessary deployment models along with sufficient database coverage to compete in all cases. We often see a full-featured platform pitted against others that focus on a single use case, because not every customer needs or wants every possible capability. So don't focus solely on 'leaders' in whatever published analyst reports you read, but cast your net across a wider group of vendors to start your 'paper' evaluations. That should give you a better idea of what's available before before you conduct a proof of concept deployment.

Define Requirements

Over and over again,we see dissatisfaction with security products stemming from a failure to fully understand internal requirements *before* product selection. We understand that it is impossible to fully evaluate questions such as ease-of-use across an entire organization before a product is in full deployment. But unfortunately, more often the real issue is lack of understanding of both the *internal* set of expectations for the product and where the organization is headed. So defining needs and getting input from all stakeholders are necessary for a successful product evaluation and selection.

- **Create selection team:** Even small firms have at least the technically-focused security and IT operations groups cooperate during the selection process; but typically different business units, along with risk, audit, and compliance, have input as well. Identify the major stakeholders and designate a spokesperson for each group.
- **Determine what needs protecting:** You need to identify the systems (file servers, databases, etc.), and data types to be protected. Summarize what the data is and how the systems are used, and map desired data flow if possible.
- **Define how data will be protected:** Map your protection and compliance needs to the systems, processes, and data from the previous step. Accept input from each stakeholder on the security and compliance requirements for each data type, and the risk or criticality of that data.

- **Design your ideal deployment:** Now that you have an understanding of what needs to be protected and how, document the specifics of integration and deployment. Determine what masks are appropriate for each data type, how data flows through your systems, and where your integration points should be.

- **Define tests:** Determine how you will verify that vendors meet your requirements. Decide what samples, data sources, and data types need to be tested. Confirm that adequate resources are available to thoroughly test the system. Pulling an old laptop from a drawer or an older server from a closet to run tests on is one way to ensure failure. Determine and assign responsibilities for who will test and who will evaluate the results. Finally, figure how you will validate the efficacy of the masks, and whether they are genuinely producing suitable results.

- **Formalize requirements:** At this point you should have a very clear picture of what you need, so it's time to document some of your requirements for a formal Request For Information (RFI) and Request For Proposals (RFP) to identify which vendors offer appropriate solutions, and then select the ones that best match your requirements for further evaluation. You should also have a good idea of your budget by this point — it will help guide your selection, and may force a phased deployment.

Key Evaluation Criteria

1. **Deployment Architecture:** Architecture is key because it determines compatibility with your environment. It also directly correlates with performance, scalability, management and ease of deployment. Centralized masking servers, distributed deployments, on-database masking, and agents are all options — but which is best depends entirely on your environment and how you want to deploy. So testing your deployment model across sufficient systems is essential for developing a good idea of how well the masking solution fits your environment.

2. **Platform Coverage:** Verify that the vendors support the relational and quasi-relational databases you need, as well as their ability to work with the applications and file servers you wish to integrate with. This is typically the first area where some vendors "wash out" of the evaluation, when they don't *fully* support one of your critical platforms. You should review vendors' published support matrices, but we find that the list of supported platforms are idealistic, don't fully capture reality and need to be vetted. We suggest you test your critical platforms to make sure they work to your satisfaction. How data is collected and managed varies from vendor to vendor, and how well each solution works with different database types can be an eye-opening comparison.

3. **Use, Customization, and Management:** Test the day-to-day tasks of adding data sources, performing discovery, adding masks and customizing reports. You will be living with this UI and workflow on a daily basis, so ease of use is a major consideration. If the product is annoying during the evaluation process, it is unlikely to become more pleasant with familiarity. Poor user interfaces make administrators less likely to tune the system, and poor workflows are more likely to cause mistakes. Ease of use is rarely listed as an evaluation criterion, but it should weigh heavily in your choice of platform.

4. **Pre-built rules and templates:** Small scale masking deployments don't take too long, with some products offering exceptionally easy to use, self-guided interfaces. But for most, your going to deploy masking across many databases, and employ masks that are specific to your security and compliance requirements. Pre-built meta-data catalogs for critical applications like SAP, PeopleSoft, and Oracle Applications save a lot of time in discovery and mask selection. Similarly pre-built masks and templates reduce deployment time and take some of the guess work

out of choosing the right mask to meet specific regulatory mandates. Look both for the type of pre-built features, and dive into the *quality* of these features as it will give yo a good idea as to the maturity of the product, as well as the vendors level of understanding regarding the problems you face.

5. **Scale and Performance:** Vendor reported performance and real world performance are usually quite distinct, so you need to evaluate all phases of the masking process. Agents may not meet expectations, in-place masking may have unacceptable impact on target systems, the masks you select may require extensive computational overhead, or centralized masking solutions may clog your network. Performance of these facets of a solution — especially scaled across your organization — provides a good idea of both performance and scalability, and forewarning of possible issues to anticipate. It is essential to test and analyze scaling with very large databases such as Teradata and Hadoop, as well as latency of in-line masking products — there is simply no alternative to testing a production-sized installation.

6. **Integration:** Most customers only evaluate integration with their platforms and applications that house and serve data, but it's important to determine how well the masking platform integrates with other supporting systems as well. Encryption, key management, reporting, workflow, and identity and access management all often integrate with masking, so investigate and test.

7. **Check References:** What do other customers say about the product, the company, and the support they provide? What did other customers expect from the product and where do they think it excels? These questions can tell you a lot about both the company and the product, so ask your perspective vendor for three references.

Negotiate

So you found an ideal product — can you afford it? When you do the math on what a masking product will cost, you need to account for not only what you need today, but at least your mid-term requirements will be as well. We have heard some vendors are inflexible on pricing, so you can always look for additional ways to reduce overall cost of ownership.

- **Pricing & Pricing Models:** There are a couple different ways vendors price their products: by number of databases under management, by volume of data masked, and by number and size of masking servers. Each of these models can have hidden costs — customer requirements grow over time as companies discover more and more data that must be protected. If your managed data doubles every 3 years, this can get quite expensive, so we recommend negotiation of a pricing plan that allows growth without linear price increases. We have seen several cases where the 'best' product did not win due to pricing.

- **Services:** We recommend that you negotiate, as part of the purchase you negotiate assistance with deployment. Most customers ask for assistance setting up compliance reports or tuning servers to scale across the organization. Most problems missed during the proof of concept pop up early in full deployment, so ask for deployment assistance. Among customers we surveyed ongoing platform management was fairly stable, but there was considerable 'front-loaded' effort for deployment and setup, while the amount of time spent on day-to-day management was relatively low. To rein in initial deployment costs and get up and running quickly, enlist the vendor's team. This also provides

mentors who know the product for your team. Push the vendor to provide some assistance with deployment as part of your contract.

- **Non-cash Add-ons:** As we mentioned, many customers struggle with costs of implementation across all of IT. Keep in mind that the incremental cost of software is zero, zilch, nada – so vendors can often bundle in a bit more to get the deal when pushed to the wall. It never hurts to ask for additional software licenses, support, or whatever.

- **Service Levels:** If your team lacks deep knowledge of how best to handle regulatory challenges, another non-cash sweetener could be an enhanced service level. Perhaps it's a dedicated project manager to get your masks, policies and reporting done. Maybe it's the Platinum level of support, even though you paid for Bronze. For customers who lack in-house security and compliance expertise, a deeper service relationship can come in handy as requirements change.

Conclusion

Masking offers unique value — beyond other data security tools — both in its ability to preserve complex data relationships while protecting data, and its data management capabilities. Masking's combination of discovery, data set management, protection, and control over data migration is unique. No other data security product provides all these benefits simultaneously. Masking reduces risk with minimal disruption to business systems. These characteristics suit masking to meeting compliance requirements. The rapid growth we have seen in the data masking segment — spurred by compliance, risk, and security demands — has driven impressive innovate to capture increased customer demand.

The value proposition is clear, but most vendors still have a way to go in terms of product maturity and providing full features across all types of data management systems. You need to do your homework, to compare your current masking system (if any) against current requirements, and check product roadmaps against your anticipated needs. Several vendors are still focused on specific types of databases, and need to broaden their offerings to remain competitive. We expect iterative improvements in platform support, better integration with supporting systems, and more out-of-the-box support for common compliance initiatives. But we also expect significantly better support for disruptive technologies such as cloud computing and NoSQL databases, as these low-cost high-performance darlings rapidly gain acceptance. But there are already several platforms that meet and exceed most customers' needs in the principle use cases. They provide quality masking capabilities, scalability, and effective data management.

We hope that you find the information in this paper useful. If you have any questions or want to discuss your situation specifically, feel free to send us a note at info@securosis.com.

About the Author

Adrian Lane, Analyst and CTO

Adrian Lane is a Senior Security Strategist with 24 years of industry experience. He brings over a decade of C-level executive expertise to the Securosis team. Mr. Lane specializes in database architecture and data security. 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, L.L.C. 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 Nexus is an online environment to help you get your job done better and faster. It provides pragmatic research on security topics, telling 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 with 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. We maintain our strict objectivity and confidentiality. 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/>.