



POINT OF VIEW

More Strategies for Eliminating Cardholder Data

By Branden R. Williams, Senior Consulting Manager, VeriSign® Global Security Consulting

After another year in the field working with customers struggling to comply with the Payment Card Industry (PCI) standards, I've had some time to revisit some of the more popular methods for securing credit card data. I've seen excellent implementations of hashing or encrypting techniques, as well as poorly planned and hastily put together ones. Many believe in employing multiple methods, including this particular author, who still recommends hashing as a solution in certain situations.

In this article, I'll analyze several popular methods and discuss strategies for applying them to your data protection strategy for PCI and other data security concerns.

+ Eliminate, Erase, Eradicate... and other words that start with E.

In reviewing the business requirements related to payment systems in both financial companies and retailers, I've found that many share a common feature: people are getting credit card data that they do not need in order to do their jobs. Some do not know they are getting the data in the first place, and others know they receive it, but avoid taking adequate responsibility for securing the data once they obtain it.

Here's an example to paint the picture. At VeriSign, several of our customers have been caught in this situation, and I'd be willing to bet that a large percentage of the reading audience will relate quite nicely. Let's say you are a retailer and you maintain your own Point Of Sale (POS) system. On any given day, some of these machines—so vital to your business—may break down and need repair. To address this, you commission a POS Field Support team that goes out and physically works on the devices. In the course of their troubleshooting, they might enable some debugging functions. For example, they may temporarily log track data to reproduce a given problem, or they might just take transaction logs (T-Logs) from the POS device and analyze them locally on their laptops to find errors.





A couple of things should jump out at you here. First, you have now violated the PCI Data Security Standard (DSS) by storing track data. I'll go out on a limb here and say that for a temporary debugging purpose, a card association or bank would allow such an action as long as everything was returned to normal and the data was scrubbed before the machine was put back into production. But what happens to the data that was put on the technician's laptop during analysis?

Uh-oh.

After further investigation of several customers, we found that many of their POS support technicians not only had full track data on their laptops from production troubleshooting, but they also had thousands of regular numbers from T-Logs! These laptops did not use full disk encryption (or any encryption for that manner), and as a normal course of business may be backed up.

Double uh-oh.

Now we have multiple violations of the PCI-DSS, and a risk of a breach from a stolen laptop, lost backup tape, or accidental exposure due to the data leaking through the organization. Take a look at the Privacy Rights Clearinghouse Web site at <http://www.privacyrights.org/>, and review the breaches noted there to date. Over 100 breaches are a result of a stolen laptop. Laptops are relatively easy to steal or lose. Without adequate protection, you might be facing an embarrassing press release, or worse, massive fines and a PR nightmare on your hands based on what was lost.

How do we prevent something like this? Easy. By revisiting the data flows and understanding who absolutely needs credit card data, by ensuring that they know they have it, and that they know what they need to do to secure and/or destroy this critical data. Finally, for ongoing maintenance and measurement, you should be constantly evaluating your IT assets to see if they are following these procedures. There is quite a bit of focus on USB drives in the enterprise and the inherent risk they bring, but not enough focus on laptop security and data flows. If the data does not go to that user in the first place, who cares if they have a USB drive or not?

Even today when we are called into a customer, we still do not see a holistic approach to understanding data flows. There are certainly experts who know their part, and 80% of the time they are right on. But they often lack an over-arching perspective of the data flows, and are unaware of data flows that lie outside of their immediate purview. The level of documentation required for overarching visibility is considerable, but it is also extremely valuable. Imagine being able to see the entire picture at once and instantly be able to identify risky areas or understand how a new service or acquisition could compromise security.

Once you have a firm grasp of all applicable data flows, the next step is understanding who consumes the data and for what purpose. As previously written, most consumers of this data do not need a card number, but they need a way to distinguish a card number in their environment in a reliable and repeatable fashion. This method needs to work over the lifetime of the card, not just the lifetime of the transaction. I'm going to discuss several methods for doing this: hashing, masking, and tokenizing.



WHAT'S A RAINBOW TABLE?

Rainbow Tables attempt to subvert a mathematical problem created by hashing. The digest created is one way, and there is no way to mathematically restore the original data from the digest. For example, you can make a bunch of oranges (the message) into a glass of orange juice (the digest), but you can't take a glass of orange juice and make a bunch of oranges. Hashes are designed to be secure, based on the notion that it is computationally infeasible to calculate every possible digest for every possible input. This breaks down slightly when you look at a fixed input (16 digits for most cards) with a limited number of possibilities (up to 10 or 11 digit BINs for some bank's products, and only 1 in every 10 numbers passes a Luhn check).

If you know how someone is calculating a hash, you can easily create a rainbow table that will exhaustively calculate all possible hashed results and store them with their corresponding inputs in less than a week's time. Now in order to "reverse" the hash, you simply match the hash values and look at what the input should be.

Hashing, revisited!

So you have eliminated credit card data in your environment pretty much everywhere that you can, and now you need to solve some business problems presented by those consumers that use the data for non-payment purposes. In a previous paper, I presented a solution for this problem that involved hashing. Hashing is an easy and inexpensive way to solve this problem, but it certainly has its drawbacks.

For example, if you simply run an MD5 hash against a credit card number, and leave all other data associated with it nearby, one can easily compromise your database by creating a rainbow table. The weakness is twofold. First, if you find a matching hash, you know that there is a high probability that the card is valid, and second, if there is other data associated with the record, you can grab additional information about that cardholder. So if you store the name, address, and card hash in the same table, the chances of the card being used fraudulently are much higher.

Where hashing can be helpful is if you are looking at trend data. Fraud detection is a natural fit if you are only looking at buying patterns and only use the card number (hash) to run heuristic analysis. If you are using it for marketing purposes, however, you are more likely to have other sensitive data nearby and these hashes should be secured like any other unencrypted data in your enterprise.

As expected, hashing is probably not your silver bullet, but it can play a very important role when looking to reduce risk related to the storage of credit card numbers.

+ Masking: Hide your face so the world will never find you!

Broadway plays aside, it may be time to throw a Masquerade! When working with credit card data, many users will require a view of part of the credit card number and some other transactional items in order to identify transactions. In my experience, you can positively assure you have found the right transaction with the following information:

- First 6 and/or Last 4 numbers of the card number
- Amount of purchase
- Date/Time (down to the second if necessary) of purchase
- Authorization Number (6 digit response)

There may be a very good case for masking data when it is presented to anyone based on this information. After looking at your card processing flows, you may realize that those "middle" 6 digits¹ are not needed for any part of your business and you can truncate the cardholder data. Interesting thought, and potentially a great way to eliminate the risk that comes with storing credit card data! After all, you can't lose what you don't have!

+ How many tokens for a dollar?

By the time we reach tokens, you've identified a few users that still need the cardholder data, and will not be able to minimize the stored data enough to implement hashing, and truncation and/or masking. Let's discuss the benefits and drawbacks to tokens.

¹ Middle six meaning the six between the first 6 digits of the BIN and the last 4 digits of the card number (123456XXXXXX1234).



POINT OF VIEW

Tokens are unique representations of data that are tracked using a database table such that you can retrieve the original data if you know the token. Sometimes these are called reference tables, look-aside tables, or look-up tables. The nice thing about this setup is that you can centralize all of your actual cardholder data into one giant lookup table and apply appropriate access controls to protect it. The hard part about it is that you need to adjust your applications to deal with this reference number like it would deal with a credit card number.

Ideally you would incorporate the token into the authorization process. Consider a model in which the POS terminal sends the card number out for authorization and receives a token in return. Doing this ensures you are protecting the card number at any aggregation point post-authorization. An alternative to doing this at the POS would be to convert the data during the daily collection of transactions.

At the very minimum it could be done post-settlement. This method results in higher risk as the credit card number will aggregate in multiple locations across the way. Consider the following best case scenario.



- 1 Customer swipes card, POS terminal builds authorization package.
- 2 Authorization package goes to POS controller, and on to corporate for processing.
- 3 Corporate sends to bank for approval.
- 4 Authorization is received from the bank, and corporate systems then check to see if they have seen this card before.
 - a. If the card has been used before, retrieve token.
 - b. If this is a new card, generate a new token.
- 5 Return the token to the POS controller and complete the transaction.

When it comes time for settlement, you would need a process to convert tokens to account numbers (unless of course you introduce tokens post-settlement). This process would be done just prior to sending your settlement file to your acquirer. At the time of this writing, there are products on the market that can go one step further and manage the tokens for you. You would send tokens along to the bank for settlement and all conversions are handled on the back end.



WHAT'S A PROPER SEGMENTATION?

If you are looking for a highly debated topic, you don't have to look farther than segmentation.

Prior to performing their first gap analysis against industry or government regulations, most merchants run a "flat" network. A flat network is defined as a network with no access control points. Keeping in mind that most merchants do have a Demilitarized Zone (DMZ) of some sort, what they didn't have was internal firewalls or other controlled segmentation to protect certain areas from others.

For example, your HR systems contain highly confidential information subject to HIPAA governance. To show due care in protecting those assets, you might create a secure enclave where that data is stored and where the users interact with it. Since your corporation probably only has a very small subset of users that need access to that data, due care says prevent the rest from being able to access it by segmenting it.

Financial systems (including payment systems) should be treated in the same regard. As an employee, you should not be able to access these systems without specific authorization. Meaning that, by merely being on the campus in a common area or even a shared conference room, you should not be able to watch transactions going to your bank for authorization. Reduce the risk and scope of your PCI assessment simply by architecting proper segmentation.

Now comes the fun part; let's revisit how this affects the scope of your PCI assessment assuming the best case scenario painted above. Your POS terminals, controllers, and path used for authorization (including supporting infrastructure such as routers and switches) is in scope. Your settlement system may be in scope (depending on if your bank has the ability to turn your tokens into card numbers. With proper segmentation, that may be the extent of your scope! What? No workstations? No other servers? Depending on how you segment, such compliance bliss is possible, saving you time and money on every assessment and letting you sleep better at night!

+ Back in the real world.

Alas, ever a realist, I do realize that such compliance bliss may only work in a utopian payment society. You can get there, or close, but it requires asking the "hard questions". Can we change the company culture to do this? Will we have the management support required to implement sweeping changes? Can I build it to scale to my entire enterprise? Is keeping the card number in more than one place worth the additional risk?

I have personally seen a system close to what is described above implemented in many small retailers and even a large one. Not only have they greatly reduced the risk associated with storing a card number through masking, hashing, or tokenizing, but they have greatly improved their ability to protect data on a large scale and keep their customers' data safe.

+ About the Author

Branden R. Williams has thirteen years of IT experience, eleven of those concentrating in information security. Early in his career, he co-founded an IT consulting business that was later sold to a venture capitalist, and worked with several of their portfolio companies to enable security-conscious growth of their business.

At VeriSign, the leading provider of digital infrastructure for the networked world, Branden runs the PCI Consulting Practice and regularly consults with top global retailers and financial institutions. Branden is also an Adjunct Professor at the University of Dallas's Graduate School of Management where he teaches in their NSA Certified Information Assurance program.

Branden can be reached via email at bwilliams@verisign.com. For general inquiries about VeriSign's PCI Related services, please email pci@verisign.com, or call (650) 426-5310.

Visit us at www.Verisign.com for more information.