Au: Please check chapter cross-refs

7

Please change to "Learning From Security Breaches"

Au: Chapter 7 title differs from tracking grid. Please choose correct title for this page and for the verso running heads.

Learning from Security Failures

"In times like these when unemployment rates are up to 13 percent, income has fallen by 5 percent, and suicide rates are climbing, I get so angry that the government is wasting money on things like the collection of statistics!"

delete double quotes x2

Hans Rosling, quoting a caller on a radio talk show, *The Joy of Stats*

Au: Check verso running heads. Global any changes.

When organizations experience a security event, their natural reaction is to focus on getting back to normal as fast as possible. They see the event as a sign of failure or an embarrassment and everything they do centers on minimizing the impact and putting the event behind them. In that environment, they often overlook one important task and miss the silver lining.

During such an event, a rich set of a data is generated and just waiting to be collected and analyzed. Think of it—If you could somehow gather that data, make sense of it, and perhaps even compare and contrast it with other security events, you could learn how to prevent the next attack. Maybe even better, you could identify trends and patterns so that you could prevent multiple common attacks with a single preventative control. Achieving such a benefit is the goal of this chapter. You'll learn how to determine what data to collect and how to manage it. The chapter also discusses how to analyze and share this data.

In order to tackle the challenge of learning from breach data, this chapter leverages the Vocabulary for Event Recording and Incident Sharing (VERIS) framework. One of the authors of this book (Jay) and the RISK team at Verizon have been developing and evolving VERIS in order to produce the Data Breach Investigation Report (DBIR). In an effort to promote adoption and use, Verizon has opened VERIS. Details about its use and implementation are hosted at `http://variscommunity.net`. Because it is open, any organization can adopt the framework and start collecting data from their own internal events. When it comes to information sharing, the data will be ready to pass around and exchange.

> ***Note***
>
> *The Verizon Data Breach Investigations Report (DBIR) leverages the VERIS framework for its data collection and data analysis and may help you get a context for this chapter. The most recent report can be found at* `www.verizonenterprise.com/DBIR/`

monofont not italic / period

Besides being an open framework, VERIS has another benefit. There is a relatively new project called the VERIS Community Database (VCDB) that offers a free and downloadable data set of publicly disclosed security events, which are all recorded using the VERIS format. This means you have thousands of VERIS records you can download and analyze throughout this chapter. At the time of this writing, the VCDB data is being housed at GitHub (`https://github.com/vz-risk/VCDB`).

Setting Up the Research

First and foremost, you should approach this breach analysis as a research project. If you think of this as a "metrics program" or a "security project," you might fool yourself into thinking this is somehow unique to information security, but it isn't. This is all about data collection and analysis, something that has been done countless times before across many disciplines and generations. Approaching this as if it were a unique project and trying to reinvent the (data analysis) wheel is not only wasteful of time and resources, but you'd be laughed at and ridiculed by all the grown-up data scientists. Let's avoid all that and call this what it is—a research project.

Most of the work in this book has been of an exploratory nature. You worked with data to see what it contained and then formed the questions you wanted to answer with the data and went back into the data.

This effort is different because you are starting with no data. If you jumped right in and started to collect the breach data, you'd waste countless resources, capture data that you'd discover later to be meaningless, and end up wishing you had data you didn't collect. Therefore, it's better to set a frame for this effort and think of a handful of questions you'd like to explore. From that, you can determine which data points you want to collect.

VERIS was developed to support the strategic decision process. In other words, where can you focus your limited resources to get the biggest benefit for your security spending? Given a list of audit findings or remediation projects, how can you prioritize those so you fix the most critical first? Perhaps even more importantly, you also want to answer the opposite questions. Can you identify areas and tasks where you do ***not*** want to spend your time and money? Supporting these questions is the goal of this chapter and can be summarized as follows:

line space x2

The goal in collecting and analyzing breach data is to support the decision-making process within security leadership.

Notice the word ***support*** in there. This research will exist to ***support*** a decision process. It is not intended to be or replace the decision process. You need to have the wherewithal to recognize that security prioritization is a complex issue and those working in the industry are just beginning to scrape away at it. At this point in that scraping, where you have very little data, you should not make the assumption that you'll get the research perfect right out of the gate. You want to focus on how much uncertainty you have now and strive to reduce that uncertainty as much as you can through this work. The decisions have been made for far too long without data analysis. You need to support that decision process so it uses every bit of information you can gather.

comma

Breach Data Reduces Uncertainty

Although it would be great to collect breach data to create a perfect and prescriptive list of priorities, it just won't happen. The data will simply help you know more than you currently do, but it can't definitively show you the path forward. This raises the question for some whether it's worth it to collect this data. Is it worth spending the time and resources to create information that doesn't tell you exactly what to do?

The answer is an emphatic yes.

Uncertainty exists in the gap between what you know and what you need to know to make the best decision. Although it's tempting to toss out imperfect information because it contains uncertainty, the value of the information should be assessed by comparison. Not between the perfect information you'd want and the information you'll get, but instead between the information you ***currently have*** and the information ***you will have***. This is where you see the value of this type of data analysis. Data will help you reduce your uncertainty by reducing the gap between what you know and what you need to know. It will help you work from a better place than you were working from before. You will be making progress and setting a foundation for reducing even more uncertainty next time. This is how scientific knowledge has evolved: a series of small steps, each reducing uncertainty a little more. Therefore, the goal is not to aim for perfect information and give up when you miss that. You should aim to simply learn more than you know now. That is where you will find value.

Considerations in a Data Collection Framework

Generating data manually from a process has several pitfalls, and if that process isn't approached carefully, it can produce shaky data and probably several big headaches. Since we have an inside view of how VERIS has developed (and some of those headaches), we have gathered the following set of guidelines for manual data collection. These are not limited to the collection of security event data, though. Any type of manual data collection can benefit from these guidelines.

Aiming for Objective Answers

First and foremost the questions you ask should ***aim for objective answers***. If a question asks for an opinion, the answers will have a whole lot of variety and be influenced by strange things like the weather or what the analyst had for lunch. In some cases this may be okay, because inconsistent answers might be better than no answers at all, and sometimes you do want to solicit the opinion of an expert with some restrictions. Most of the time, however, the questions should be focused on asking about facts that are observable or deducible from observations.

For example, it's far better to ask whether malware was involved in the attack and which functions it performed instead of asking how advanced the malware was. The investigator during the breach can answer yes or no as to whether malware was used. If the investigator has the resources to analyze the malware (or the malware is identifiable), there isn't a lot of guesswork about what it's capable of. These are things you either know or don't know.

Limiting Possible Answers

Next, constrain the possible answers to a short set of options. If the question asks for a sentence or description, it won't be useful directly in the data analysis without a lot more effort. Most of the time, free text fields are helpful to record unique aspects or to set context if you ever want to understand why these data points look like they do. With this in mind, manual data collection should make judicious use of Notes fields and a field for the overall Summary of the event. But remember, all of the data analysis will use the data found in the constrained lists or numbers. Having the data limited to a set of values will make that analysis easier in the long run.

Allowing "Other," and "Unknown" Options

Most every constrained list of answers must allow "Unknown" and "Other" answers. Even though a question may seem so easy that everyone should know the answer, the world will always create a circumstance to prove that assumption wrong. Including an "Unknown" option allows you to differentiate between when you really don't know and when the question isn't applicable. This is a subtle distinction, but one that can really mess up the analysis. There are a few rare questions that don't need an "Unknown" option, but they are rare and you'll know them when you see them.

comma

For example, if you'd like to know if a server is virtualized or not, it may be tempting to create a simple checkbox if it was virtualized. But that doesn't account for the circumstances when the information isn't available (and there will ***always*** be a circumstance when it won't be available). Now you've set up the response as "yes" and "everything else," where "everything else" represents both "no" and "I don't know."

The result is you would not be able to create a percentage of hosts that were virtualized or look at non-virtualized systems because you can't tell a "no" answer from an "I don't know" answer.

The second field you need to add is "Other" or, depending on the question, "Not Applicable." Avoid trying to capture all the options in an exhaustive list. Exhaustive lists become unmanageable (which slows down data entry), and you need to capture only ***most*** of the answers. You'll find that a handful of common answers, especially within security events, will be used most of the time. The common answers create the trends and statistics, whereas the uncommon answers do little more than create interesting stories. Therefore, you want to capture the common things for data analysis and relegate the uncommon to the "Other" category and the Notes field. Keep an eye on anything marked "Other," but if you create a good list of options, they should show up few and far between.

Note

It's okay to be lazy when creating lists of selection options by seeking out standards to reference and leverage. For example, don't create your own list of industries to gather. Leverage the good work of the U.S. Census Bureau and its North American Industry Classification System (NAICS). The Census Bureau has already figured out that industries are nested and created a system to capture both the general industry and an organizations specific function within that industry. They assign a six-digit code where each digit adds a level of detail about the industry of the organization. Details about the NAICS classification can be found at `http://www.census.gov/eos/www/naics/`.

has apostrophe

remove italics

Avoiding Conflation and Merging the Minutiae

The last two points may seem subtle, but you want to avoid conflation and merge the minutiae where possible. These two concepts are opposites, and you have to find the middle ground between them. Conflation occurs when a question (and its answers) combines more than one concept.

For example, the breach types used by DataLossDB (`http://datalossdb.org/analysis`) conflate the actor, actions, and assets into the type. Their framework lists a type of "Hack" for a "computer based intrusion" (no asset or actor defined), or "snooping," which is an "employee . . . accessing confidential records" (conflating the actor and action). You can specify "stolen media" or "stolen drive" or "stolen tape," which are all unique options that conflate and repeat the action (physical loss) with the asset.

The assignment of a single conflating "breach type" should not be thought of as wrong or bad, it just represents a different goal within the research. Just be aware that conflation of terms like this will create a challenge during data analysis. You have the most flexibility and see the most benefit when you can compare and contrast across specific categories, but with conflated terms you'll find it challenging to clearly separate the categories. The result is that the data analysis may not be able to do anything more than simply count the frequency of the conflated breach types.

Where conflation combines more than one concept into a single variable, you have to be careful of the opposite, whereby you split a single concept into too many details. You want to get just enough detail and separation in the list to support your goal.

An example of collecting too much detail is when you try to collect how incidents are discovered and want to classify the discovery method. Although it may be interesting to know if it was an external security researcher, and perhaps amusing to know what color hat they wore (white, black, or even grey), creating several options based on details wouldn't help you achieve your goals. You have to split one concept (an external security researcher) across multiple selections (such as Researcher-white hat, Researcher-gray hat). In this case, maybe you'll just want to drop the distinction of an external security researcher altogether and merge everything into one broad field of "an external unrelated party."

Having said this, don't be afraid to go into detail when necessary. As an example, the list of possible assets within VERIS is split into several categories and dozens of detailed assets under each category. There are times you'll want to split details and times you can combine them—the trick is getting that balance right.

Luckily, all these issues have been in consideration as VERIS has been evolving. One of the biggest challenges is saying no to new questions. We've found there is always more we'd like to know, but we know that each data point we try to collect has a cost. And it's sometimes proven to be a higher cost than imagined.

Consider the Cost per Datum

During a manual data-collection effort, it is very tempting to dream up all sorts of questions you'd like answered. Creating such a list isn't bad, and it may even be helpful to lay out all the questions to answer. But choose the questions you are going to ask very carefully because every question adds exponential cost across the lifetime of the data. Even before the question is answered, you have to build a method to collect it, so every question must be built into the data-collection methodology. When the analyst is entering an incident, each question will require some thought and perhaps even some research before it can be answered, again adding time and effort. That data point may require processing and clean up, and will need to be stored and managed. Anytime you want to parse the data (and you'll want to parse it in many different ways), you might have to consider this field, or worse, consider all the interactions of all the fields. Beyond that, there are dozens of other subtle interactions with the data that will increase the cost of each data point beyond what you can imagine as the questions and data points are selected.

cleanup

comma

It's helpful to pretend you are about to take a long journey to a wise sage who lives on top of a mountain. You will have a limited amount of time to ask questions before the sage says something mysterious and vanishes. What questions will have the greatest impact? You'll want to identify a handful of questions you really need answered—maybe a handful of questions you'd like to have answered and then you'll have a mountain of questions you wish you had time to ask, but you'll just have to make do without them. The same is true with manual data collection. If the post-incident questionnaire asks too many questions or is too painful, people will lose interest quickly and the answers will end up being of poor quality—including the handful of questions you need answered. You must choose your questions wisely.

comma

An Introduction to VERIS

When a security event is investigated, a narrative naturally emerges from the process. The investigator will typically try to answer the question, "Who did what to what (or whom) with what result?" That question presents a good core set of data points to collect. Therefore, as a starting point, you'll want to focus

on those four points—"Who (threat actor) did what (action) to what or whom (asset) with what result (attribute)?"

But that's not all you may be interested in; you may also want to know how you discovered and responded to the incident and if possible the impact you experienced as a result. Finally, you'll have some housekeeping items (an identifier, summary, workflow status, and so on) and if you aggregate breaches or share the information, you'll want to record some victim demographics. Overall, you can break down the sections of data you want to gather like those in the VERIS framework, as shown in Table 7-1.

TABLE 7-1 *Sections Within VERIS*

VERIS Section	Purpose
Incident Tracking	Metadata about the incident for management and tracking purposes.
Threat Actor	One or more people who cause or contribute to an incident.
Threat Actions	What the threat actor(s) did or used to cause or contribute to the incident.
Information Assets	Information assets that were compromised or affected during the incident.
Attributes	What happened to the asset during the incident
Discovery/Response	Timeline, discovery method, and lessons learned.
Impact	What was the overall effect of the incident to the organization.
Victim	Demographic information like industry and organizational size.
Indicators	Optional indicators of compromise (IP addresses, malware hashes, domains, and so on).
Plus	Optional section for extending VERIS

lc

delete period

delete period

delete period

delete period

question mark

delete period

question mark

delete period

delete period

delete period

Although it's tempting to dig into the data (and you will), it's important to understand the significance of these fields so you don't misapply them. Therefore, the following sections go through each part of VERIS in more detail and discuss these fields. Keep in mind that these sections are separated so analysts can think about the structure. There is nothing in the actual data denoting the `incident_id` field as helping with incident tracking, for example.

Warning

Although we are covering VERIS with some depth, we will not go into every field, and we don't cover every detail about the framework. For example, we won't call out all the places the framework specifies a "Notes" field (which is almost every section), and we don't cover the "Indicators" section in detail. Just keep in mind that the framework is actively maintained and evolving. This chapter discusses the 1.2.1 release, so be sure to refer to `http://veriscommunity.net/` for all the details and current specification of the VERIS framework.

Incident Tracking

Some of the fields within VERIS exist to simply describe or track the incident. These fields help you keep records straight by identifying each with a unique identifier and tracking the source of the incident and any related incidents. You use the `source_id` field to compare your unique "source" of incidents to something like the VCDB (which has `vcdb` in that field). If something has a value of `factor`, that means it is a restrictive list (we discussed creating those lists previously), and only those values are expected. See Table 7-2 to see the incident tracking fields.

comma

TABLE 7-2 *Incident Tracking Fields*

Field	Value	Description
schema_version	string	VERIS version (currently 1.2.1)
incident_id	string	Unique identifier (VCDB uses GUID)
source_id	string	Origin of the data (VCDB data has vcdb)
reference	string	URL or internal ticketing system ID
security_incident	factor	Confirmed, Suspected, False Positive, Near Miss
confidence	factor	High, Medium, Low, None
summary	string	Free text summary of incident
related_incidents	string	Free text, other incident_ids
notes	string	Free text

There are only one or two fields you'll use during analysis and those are the two `factor` variables (again, this means they are restricted to a list of expected answers). The security incident is required and will help you split the analysis on whether the event was a confirmed security incident (an asset has a security attribute affected). The confidence rating is a rare subjective field. It enables the analysts to record their subjective assessments as to how confident they are in the accuracy of the data they entered. This optional field is not heavily used and won't appear much in the VCDB incidents that you'll look at.

Threat Actor

Earlier in this chapter you read about the challenge of conflation. This is something you want to be aware of, especially in these next three sections (actor, actions, and assets). You read about the framework DataLossDB used with a single conflated breach type, and you'll see the same thing with the framework used by Privacy Rights Clearinghouse (`http://www.privacyrights.org/data-breach`). Their framework also uses a singular "breach type" to define each event and again it simplifies the actors and actions into the one label.

comma

For example, they have an "insider (INSD)" type, which is defined as "someone with legitimate access intentionally breaches information—such as an employee or contractor." There is also the "physical loss (PHYS)" type, which is defined as "lost, discarded, or stolen non-electronic records, such as paper documents."

These simplified labels can quickly become confusing during data entry if, for example, an insider steels paper documents.

You may see insiders breach systems, drop malware, and social engineer, just as an external actor would and you want to separate the insiders from external actors clearly in the data. VERIS tackles that conflation by separating the who from what they did and what was affected. Note that the method that the Privacy Rights Clearinghouse uses isn't wrong. It just has a different focus and represents different priorities and goals. VERIS has the goal to inform and support security decisions, which benefits from more detail than a single "breach type" label. Table 7-3 shows the threat actor fields.

steals

comma

TABLE 7-3 *Threat Actor Fields*

Actor	Field	Value	Description
external	motive	factor	Helps understand intentions; same enumeration for the three actor types
	variety	factor	Shapes resources/capability of external actor
	country	factor	ISO-3166-1 two-digit country field
internal	motive	factor	Helps understand intentions
	variety	factor	Shapes resources; capability of internal actor
partner	motive	factor	Helps understand intentions
	industry	string	U.S. Census NAICS code
	country	factor	ISO-3166-1 two-digit country field

These two things should be "and". Make them "Defines resources and capabilities of…"

("Shapes" was a bad choice of words)

semicolon

No, see note on other side

The threat actor section also introduces the nesting feature of VERIS. At the top level is the actor, so there is a section in the data for "actor," and then there are three classes of actors—external, internal, and partner—all of which are optional. Within each of those classes you'll want to add details about that type of actor. Looking down the values in this section, you can see all factors. That means you should be able to include any of these or use them as pivot points. In other words, if you want to support a threat-modeling exercise that compares different threat communities, you could extract the actions for financially motivated actors and compare them to disgruntled employees. See Figure 7-1.

Threat Actions

This section collects variables to describe what the threat actor(s) did or used during the event. Again, there are nest variables under the top-level categories:

- **Malware:** Malicious software, script, or code run on an asset that alters its state or function.
- **Hacking:** Person (at a keyboard) attempting to access or harm an asset without authorization.
- **Social:** Exploiting the human element (phishing, pretexting, and so on).
- **Misuse:** Abusing resources or privileges contrary to intended use.

em dash / period x4

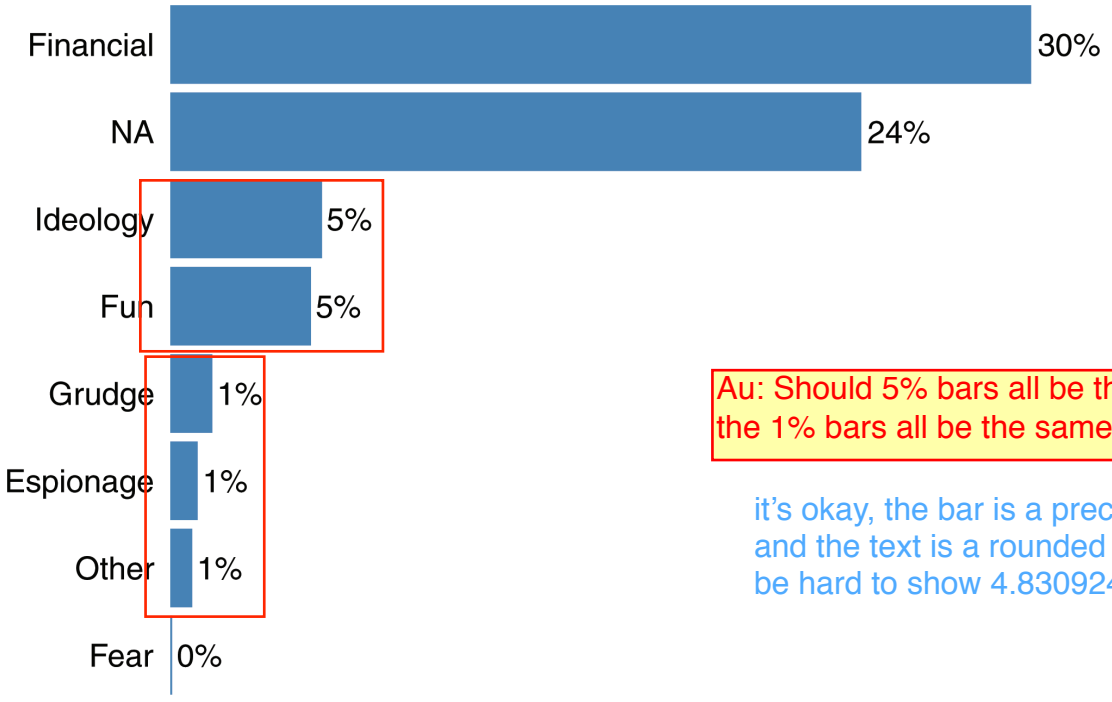
em dash /period

em dash /period

em dash

period

- **Physical:** Personal actions involving proximity, possession, or force.
- **Error:** Anything done (or left undone) incorrectly or inadvertently.
- **Environmental:** Natural events and hazards within the immediate environment or infrastructure of assets.



Au: Should 5% bars all be the same size & the 1% bars all be the same size?

it's okay, the bar is a precise value and the text is a rounded value. It'd be hard to show 4.8309245% and

FIGURE 7-1 *Known motives across all actors (Percentage of Events)*

You have to be careful as you work with these categories. There are many opportunities for misinterpretation and misclassification across categories. These categories and the factors in each category are explained in detail along with use-case examples at the VERIS website (`http://veriscommunity.net/`). Once you spend some time and look at a few examples, these categories become more clear and eventually will become intuitive. See Table 7-4 to see the action fields.

rebreak

TABLE 7-4 *Action Fields*

Action	Field	Value	Description
malware	variety	factor	Functionality of malware
	vector	factor	How the malware was installed/infected

~~(continues)~~

remove extra space

TABLE 7-4 *Action Fields (Continued)*

Action	Field	Value	Description
hacking	variety	factor	Type(s) of hacking action
	vector	factor	Path of attack
social	variety	factor	Type(s) of social action
	vector	factor	Path or method of communication
	target	factor	Role of targeted person
misuse	variety	factor	Type(s) of misuse action
	vector	factor	Path or access method for misuse
physical	variety	factor	Type(s) of physical actions
	vector	factor	Method of physical access
	location	factor	Physical location of action
error	variety	factor	Type(s) of error actions
	vector	factor	Cause of error
environmental	variety	factor	Type(s) of environmental actions

Notice how `variety` and `vector` are repeated over and over? Every action category has a `variety` field with unique enumerations for each category. All but the environmental actions have a `vector` field, again with unique enumerations for each category. Finally, social actions also ask for the target of the social action, and physical actions ask for a location of the action. That explains the whole section! Notice how every field here is a factor, meaning you can split, pivot, and/or filter based on these fields. See Figure 7-2.

comma

comma

Multiple Events in the Attack Chain

Anyone who has been around information security knows that breaches tend not to be simple and single events. Oftentimes, the attacker will perform multiple actions, which complicates the recording process. Most of the factors in the VERIS framework support multiple answers. On one hand, this is very nice because you don't have to pick "the one best answer" for a complex security event, but on the flip side, this adds complexity for data management and analysis. As you'll see later in this chapter, this isn't as hard as it first seems.

continues

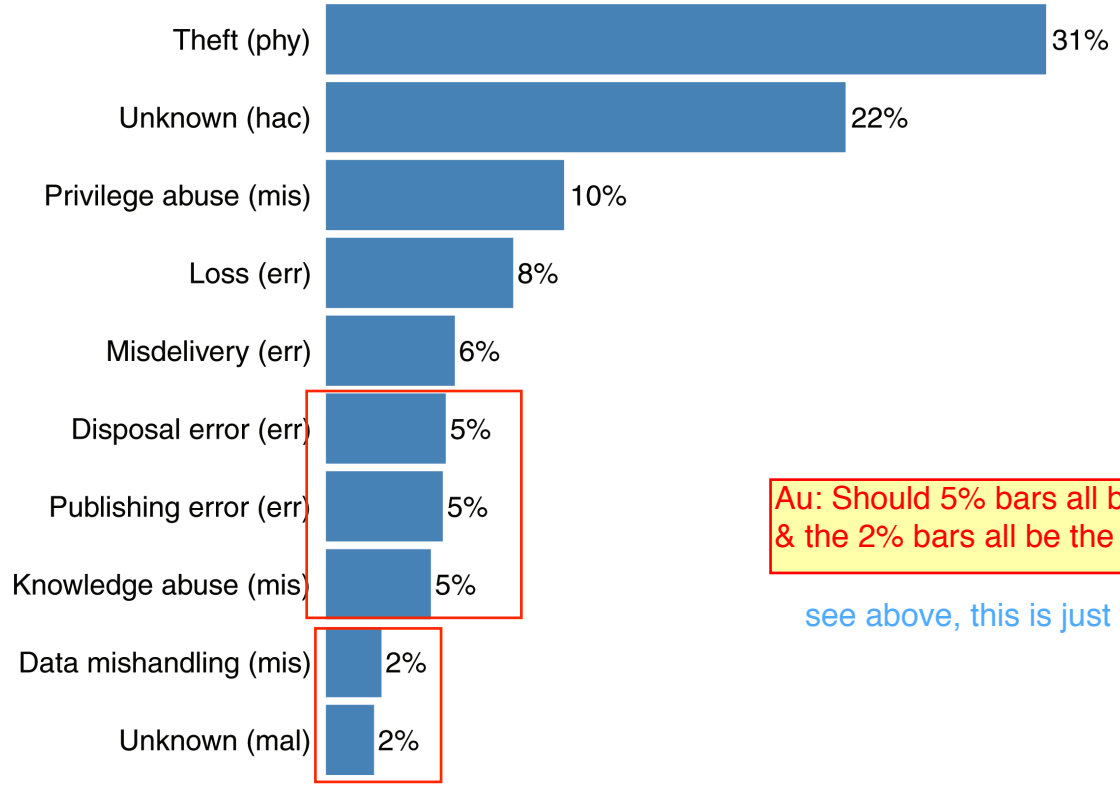
lc

(Continued)

As an example, suppose an attacker sends a phishing email to an executive's assistant and quickly follows that up with a phone call pretending to be a business partner who sent the email. These are two actions, and you should see both "pretexting" and "phishing" selected in the `social` `variety` field. If the phishing email contained malware that is installed, you'd also see the malware action along the variety and vector of "email," since it was installed via the phishing attack. When you represent this data and count the actions, you'll have more individual actions than total events. This naturally precludes the use of pie charts, which ends up being beneficial for all parties involved.

rebreak

There is also a common notion within information security of the "attack chain" or "kill chain." The concept is to establish the actions of the attacker in the order they happened. Although VERIS allows multiple actions, it does not record the order in which they occurred. This was a conscious trade-off of cost versus benefit. Attempting to put the events in order created substantial overhead for the analysts and was taking too long to enter. Most of the time the order of events in reports and tickets (definitely in media articles) are either vague or missing. As a result, VERIS simply records the presence of events within an attack to reduce the time and effort spent in data collection.



Au: Should 5% bars all be the same size & the 2% bars all be the same size?

see above, this is just fine.

FIGURE 7-2 *Top 10 varieties of threat actions*

Information Assets

Assets are the information containers (servers or other devices) that you are trying to protect. Like the other sections we've covered so far, there are top-level categories, which are as follows:

em dash x6

- **Server (S)** System providing service(s)
- **Network (N)** Infrastructure device or appliance
- **User Device (U)** End-user equipment (laptop or desktop)
- **Media (M)** Data storage devices or physical documents
- **People (P)** Since people can be affected
- **Kiosk/Public Terminal (K)** Public-use devices

Within each category there are several varieties of assets, but the category and variety are stored in the same field. For example a mail server is stored as "S - Mail" and a desktop computer is a "U - Desktop." Associated with each asset is an optional `amount` field, which allows you to record multiple assets with the same variety when they are involved in one event. Table 7-5 shows the asset fields.

as / ?

TABLE 7-5 *Asset Fields*

Asset	Field	Value	Description
assets	variety	factor	Specific type of asset; prepended with letter for category
	amount	integer	Count of the above asset
asset	accessibility	factor	How accessible the assets are
	ownership	factor	Who owns the assets.
	management	factor	Who manages the assets .
	hosting	factor	Where (physically) is it hosted,
	country	factor	Location of assets (if different from victim)
	cloud	factor	Type of cloud service, if cloud

assets / ?

This is correct as is, it's a weird section in the framework.

question mark x3

There is quite a bit packed into the assets, and these are relatively recent additions to the VERIS framework (version 1.2). There is a lot of focus on mobile devices and employees bringing their own devices into the corporate environment. Also, there may be unique exposures from cloud hosted applications and assets, so that is captured here as well. Note also that these are all factors, so there are only a handful of possible answers. You cannot write in "very" for accessibility of the asset as an example. See Figure 7-3.

Attributes

The attributes of the preceding assets are what you work hard in information security to not have affected. Attributes are based on the C.I.A. triad, which stands for confidentiality, integrity, and availability.

For a while VERIS extended these three with three more attributes to record the Parkerian Hexad (named after their originator, security pioneer, and long-time security researcher, Donn Parker). The extra three attributes included possession/control, authenticity, and utility. But the added fields just did not yield enough benefit for the added cost of separate categories so they were combined with the three top categories. For simplicity, when a VERIS record is stored, the sections are labeled with the three primary categories (confidentiality, integrity, and availability). The three main sections of attributes are as follows:

comma

em dash x3

- **Confidentiality, possession, and control:** Data was observed or disclosed to an unauthorized actor; owner may no longer have exclusive custody.
- **Integrity and authenticity:** Asset is incomplete or changed from authorized content and function; conforms to expected state.
- **Availability and utility:** Asset is not accessible, useful, or fit for use.

These categories can be quite helpful to the security team in determining the areas to focus on. The Verizon Data Breach Investigation report has exclusively focused only on breaches where the confidentiality attribute was affected and there was a confirmed data disclosure. See Table 7-6 for a look at the attribute fields.

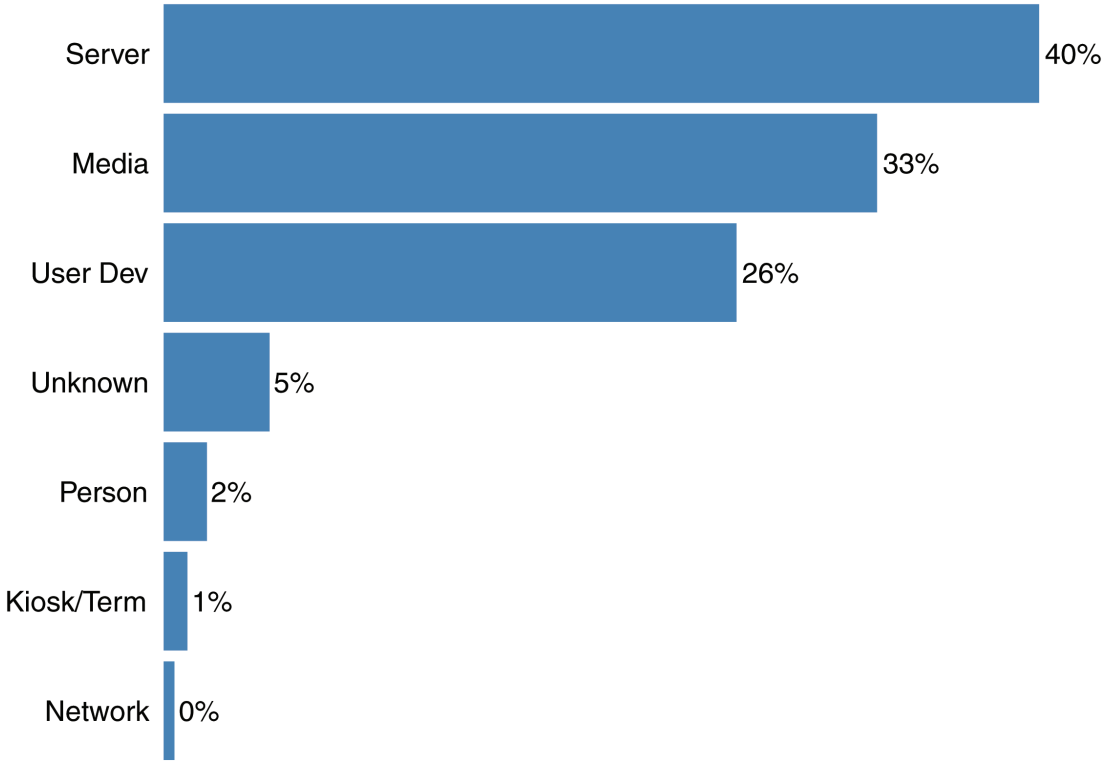


FIGURE 7-3 *Asset categories*

TABLE 7-6 *Attribute Fields*

Attribute	Field	Value	Description
confidentiality	data_disclosure	factor	Status of confidentiality breach
	data_total	integer	Number of records (see below)
	data.variety	factor	Type of data disclosed
	data.amount	integer	Number of records
	state	factor	State of data when disclosed
integrity	variety	factor	Nature of effect
availability	variety	factor	Nature of effect
	duration	time range	Duration of availability/utility loss

There is a new field type called `time range`, which is actually two fields, a "unit" of time and a "value" for that unit of time. The unit has basic measurements of time: seconds, minutes, hours, days, weeks, months, and years. The value represents how many of those, such as 3 weeks or 6 months. VERIS tried to support specific date/time fields instead of time ranges but found specific date/time was either time consuming or just impossible. However, VERIS analysts found knowing whether a range was days versus weeks was obvious even if a specific range was not known. For example, analysts might know that the server went offline during the DDoS attack, and that it was down for more than 60 minutes but not a full day. Tracking down the specific times may be difficult, but in that case, you would see "hours" in the unit, and if the specific number of hours is known, you'd see a value. Otherwise, it's blank if the precision is unknown.

three / six

Counting Records

One of the more common pieces of information disclosed in publicly disclosed breaches is the number of records affected. Perhaps reporters and the general public demand this, and the victims are forced to provide a number, even if it's all the records in a database. Records can be relatively easy to count when the data is obviously separated. Payment (credit) cards, identities, and medical records are all quite clear in their separation and lend themselves to being counted. But when you get into more complex types like classified information or trade secrets, the capability to count records becomes more difficult. Perhaps the number of physical documents could be used, or the number of files disclosed, but oftentimes it's difficult to count them. Overall, analysts struggle to record a precise number for the data varieties of classified or internal information and trade secrets. You will have to account for that in the analyses and visualizations.

Discovery/Response

You just saw your first time range in the availability attribute and you'll see that a lot more in this section for the timeline data. Some of these fields are not in a section as you saw in the previous four VERIS sections, but the timeline does have its own section. See Table 7-7.

TABLE 7-7 *Discovery and Response Fields*

Section	Field	Value	Description
	discovery_method	factor	How the event was discovered
	control_failure	string	Free text field to describe what, under the victim's control, failed
	corrective_action	string	Free text field describing what the victim should do
	targeted	factor	Targeted or opportunistic attack
timeline	incident_date	date	Date of incident
	compromise	time range	Time to initial compromise
	exfiltration	time range	Time from initial compromise to data exfiltration
	discovery	time range	Time from initial compromise to discovery
	containment	time range	Time from discovery to containment

Notice the new type of value called `date` here, which is not a standard date field. Because VERIS has to account for unknown values, the date fields are in separate variables. Too often than you'll like for your analyses, the precise date of the incident isn't known or isn't reported precisely. The framework assumes at least the year is known, but the month, day, and time fields are all optional in that date field. The other fields in the timeline are the same time range values you saw in the availability attribute.

Notice also that the control failure and corrective action suggestions are free text. This makes them difficult to include in the data analysis without more effort. Finally, the discovery method is one of the rare enumerations that cannot have multiple answers. The framework assumes that once the incident was discovered it could not be discovered again, so only one method of discovery is allowed.

Impact

The impact section (see Table 7-8) is perhaps one of the most, if not the most, sparsely populated section in the incidents. This has nothing to do with the framework, and everything to do with the lack of accurate data to collect and record about the impact. The result is that this section has some subjective measurements and estimates.

Notice the repeating rating and monetary estimations. There is a dedicated `overall` field here for those fields, but the loss section is defined in the data as an array. This means that the analyst can add multiple loss sections in the data for each variety of loss being recorded. The loss varieties are specific types of loss, for example "response and recovery" costs or "legal and regulatory" costs.

TABLE 7-8 *Impact Fields*

Section	Field	Value	Description
	currency	factor	ISO 4217 currencies for monetary estimations
overall	rating	factor	Qualitative rating of overall impact
	min_amount	number	Minimum estimated monetary amount
	amount	number	Most likely estimated monetary amount
	max_amount	number	Maximum estimated monetary amount
loss	variety	factor	Specific category of loss
	rating	factor	Qualitative rating of overall impact
	min_amount	number	Minimum estimated monetary amount
	amount	number	Most likely estimated monetary amount
	max_amount	number	Maximum estimated monetary amount

Victim

The last section I want to cover is the victim entry. If VERIS is being implemented inside a single organization and the victim is always the same, you can skip or hard-code the fields in this section. (This assumes that you aren't tracking incidents at partner organizations, suppliers, or customers.) For cases like the VCDB, which is aggregating across many victims, this section is vital. You want to capture data about the victim with the intention of contrasting and comparing breach data when you split these fields.

Chapter 5 discussed regression analysis, and you attempted to find independent variables that could help describe the outcomes you observed. The data you are collecting about the victim can go a long way to describe the types of threat actors and their actions. For example, in the 2013 DBIR, Verizon saw state-affiliated espionage in at least three out of every four cases within the manufacturing industry, yet none in the retail industry. Although industry alone is not a perfect predicting variable, it does help reduce the uncertainty, which is what you are after here. Table 7-9 shows the victim fields.

Au: I or we?

I don't know what this comment is referring to

OK

TABLE 7-9 *Victim Fields*

Victim	Field	Value	Description
victim	victim_id	string	Identifier or name of victim
	industry	string	U.S. Census NAICS code
	employee_count	factor	Label for number of employees

(continues)

continues

lc

remove extra space

TABLE 7-9 *Victim Fields (Continued)*

Victim	Field	Value	Description
	country	factor	ISO 3166-1 two-digit country code
	state	string	State, province, or region in country
	locations_affected	integer	Number of locations affected
	revenue	integer	Annual revenue of the victim
secondary	victim_id	string	List of secondary victim_id or name(s)
	amount	integer	And/or count of secondary victims

The most recent change to the VERIS framework (version 1.2.1) changed how this section is stored. In version 1.2 and before, the entire victim section could be repeated for each victim involved in the incident. For example, if an organization is breached and was processing data on behalf of another organization, they would become a victim of the same breach. This was found to be confusing though, and the victim was reduced to just supporting one single victim. The fields in the "secondary" section where added in 1.2.1 to capture what was treated as a multiple victim breach. See Figure 7-4 to see the top five victimized industries according to the VCDB data set.

were

Healthcare

Let me know if this image must be recreated, but you have the EPS

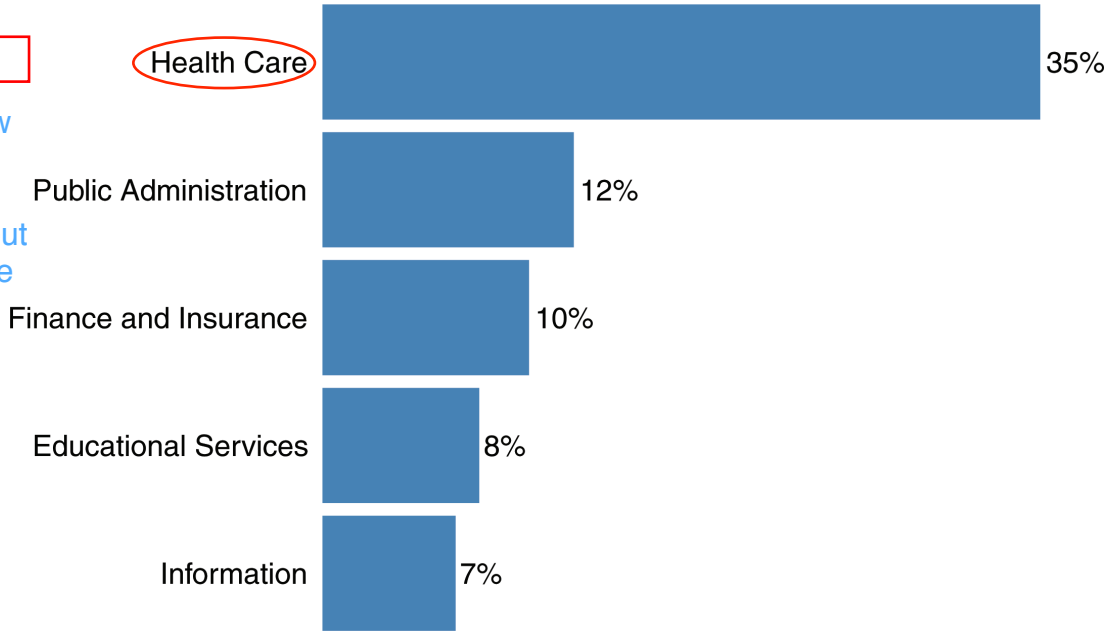


FIGURE 7-4 *Top five industries in VCDB data set*

Anywhere there is an industry (which is in this victim section and in the threat actor partner section), they are listed as a "string" but they should not be free text. Following one of the guidelines to leverage other resources wherever possible, VERIS leverages the U.S. Census Bureau's North American Industry Classification System (NAICS) mentioned earlier in this chapter. Doing so adds flexibility and a level of detail not possible with other industry classification systems. If analysts tried to create a list of industries, they'd probably come up with a dozen or so high-level categories and call it a day. NAICS started there (20 top-level categories actually), but then made it extendible and includes more detail for the industry specification. Industries within NAICS are represented by two- to six-digit integers, which is why VERIS stores them as a string and not a factor. The list is enormous (you can find it at `http://www.census.gov/eos/www/naics/`).

rebreak

As an example, consider the pizza shop down the street. The NAICS code for such a place is 722511, which represents "pizza parlors, full service." Maybe in this case, however, the analyst just knows it's a restaurant, so she records 7225, or maybe she knows the place offers some type of food or beverage service, so she enters 722. If she is unsure as to what type of service establishment it is, she might just enter 72 for "accommodation and food services." When you analyze this field, you can drill down or up depending on the level of detail you want.

Indicators

We didn't want to go into detail about the indicators section, and that's because VERIS is set up to capture indicators from a security incident, which is often just a handful of IP address and malware hashes. If you're looking to capture indicators from multiple sources, we suggest you look at the Structured Threat Information Expression (STIX). It is a "collaborative community-driven effort to define and develop a standardized language to represent structured cyber threat information" and detail about it can be found at `http://stix.mitre.org/`. If you'd like to see the details of the indicators section, they are covered on the VERIS community website at `http://veriscommunity.net/`.

comma / s

Extending VERIS with Plus

Finally, VERIS has the catch-all section labeled "plus." Within the VERIS specifications there technically is nothing specified in this field and the data schema simply allows anything to exist in this section. It exists to allow individual implementations to record additional fields not in the base VERIS schema. If you look at the VCDB repository, for example, each incident has a plus section with the analyst who recorded the incident and the time it was created along with a few other fields. Any implementation can apply the guidelines we presented earlier in this chapter (or not, at your own peril) and add their own fields here. If you have fields that are particularly useful, feel free to suggest the change to the core framework!

comma

Seeing VERIS in Action

It's always helpful to take some time before jumping into the analysis to look directly at the data. It helps set the context and may help shape your approach to the analysis. Since the average incident is about 100 lines of JSON, we don't include the whole incident. Please take some time to surf around the VCDB repository and look at the data there for full records. As a good example, Listing 7-1 shows the actor and action sections from an incident from VCDB.

Listing *-0 was used as a chapter setup. This didn't work out that way

Au: There is no Listing 7-0. Please re-number listings sequentially.

7-0

Remove Double Listing (only have green Listing)

Au: This code line is italic in other chapters. Check that all are correct.

I did not intend the label of the listing to be included as a comment.

LISTING 7-1

```
#Listing 7-1
"actor": {
    "external": {
        "country": [ "SY" ],
        "motive": [ "Ideology" ],
        "variety": [ "State-affiliated" ]
    }
},
"action": {
    "hacking": {
        "variety": [ "Use of stolen creds" ],
        "vector": [ "Web application" ]
    },
    "social": {
        "target": [ "End-user" ],
        "variety": [ "Phishing" ],
        "vector": [ "Email" ]
    }
}
```

If you have never seen JSON before, this is what it looks like. Rarely if ever would you want to edit the JSON by hand. It's not that JSON is terribly difficult, but it is terribly easy to mistype something. You could forget a comma or quote or something would prevent the data from loading properly. If you do attempt to create or modify a JSON file by hand, be sure you have a way to check your work, validate the JSON, and if possible, validate the values and factors within the data.

The best part about working with JSON is that it typically imports right into native objects in the languages you use. Within Python, an incident in JSON is imported directly into a Python dictionary. The Python code to load a JSON object and view the hacking variety is relatively simple (Listing 7-2).

remove

LISTING 7-2

```
#Listing 7-2
# python to load JSON and read hacking variety:
import os, json
# set working directory to chapter location
# (change for where you set up files in ch 2)
os.chdir("~/book/ch07")
# Open the JSON file and read the raw contents into jsondata
jsondata = open("data/vcdb/F58E9169-AC07-400E-AB0E-DB784C6CAE59.json")
# convert the contents into a python dictionary
incident = json.load(jsondata)
# now access the hacking variety (assuming it exists)
print(incident['action']['hacking']['variety'])
```

This code would print the Python list object for hacking variety and display `[u'SQLi']` (an array of one Unicode string showing SQL injection was the only hacking variety used). In production code, you should wrap the `json.load()` command with `try-except`. If the file has any errors in the JSON syntax, they

will be caught that way. Plus the hacking action is optional, and you'd want to test if the `hacking` key existed before you attempt to read it. But this example shows how easy JSON can be to load and work with.

Within R, JSON files are converted to a native list object. Before you run the R code in this chapter, you'll need to set your working directory and load the libraries used in this chapter (see Listing 7-3).

LISTING 7-3

remove

```
#Listing 7-3
# set working directory to chapter location
# (change for where you set up files in ch 2)
setwd("~/book/ch07")
# make sure the packages for this chapter
# are installed, install if necessary
pkg <- c("devtools", "ggplot2", "scales", "rjson")
new.pkg <- pkg[!(pkg %in% installed.packages())]
if (length(new.pkg)) {
  install.packages(new.pkg)
}
```

Now within R, in order to performing the same function we did in the python example in Listing 7-2, you need to load the `rjson` library in order to read in JSON data (Listing 7-4).

LISTING 7-4

remove

```
#Listing 7-4
library(rjson)
# fromJSON accepts a filename to read from
jsonfile <- "data/vcdb/F58E9169-AC07-400E-AB0E-DB784C6CAE59.json"
incident <- fromJSON(file=jsonfile)
# print the hacking variety
print(incident$action$hacking$variety)

## [1] "SQLi"
```

The R code returns a one-element vector with the value in the hacking variety. Again, in full-featured code, you'd want better error checking than this, but it does show how easy JSON data is to load into native objects and work with.

While the data from the book website (at…) has the VCDB data we use in this chapter, you may want to grab the current version of the VCDB data (there will be more incidents when you are reading this).

Working with VCDB Data

See other side

Au; Check sentence for sense.

the

This section walks through some data from VCDB using R. While the data available from the book website (at `www.wiley.com/go/datadrivensecurity`) has the VCDB data we created the figures and output from here, you may want to grab a newer version of the VCDB data. Head on over to the VCDB GitHub repository at `https://github.com/vz-risk/VCDB` and either fork, copy, or download the `VCSB-master.zip` file of the repository (use the Download ZIP button on right side of the window). The incidents themselves are quite small, but you can still learn quite a bit in spite of the data not being "big." Feel free to explore the incidents in the repository and get a feel for the files and the data. Keep in mind that all of these incidents are collected from publicly disclosed events, which makes many of the incidents rather light on the details.

For this analysis you'll leverage the `verisr` package, which was developed by our own Jay Jacobs and is in his GitHub repository (found at `https://github.com/jayjacobs/verisr/tree/master/R`). Note that the `verisr` package is actively in development, so be sure to refer to the latest documentation of the package for the most current description of its functions. By the time you are reading this, there may be all sorts of wonderful features in the package that aren't there at the time of this writing. Also, that means that some of the figures and output may be slightly different for you.

In order to install the `verisr` package from GitHub, you have to load the `devtools` package first (see Listing 7-5). This is one of many great packages from Hadley Wickham, and it allows you to install R packages directly from their GitHub repositories, which is what you'll do with `verisr`.

LISTING 7-5

remove

```
#Listing 7-5
# load up devtools
library(devtools)
# install the verisr package
install_github("verisr", "jayjacobs")
# load the verisr package
library(verisr)
```

You can now load up the VCDB data with the `verisr` package, if you downloaded current VCDB data, you must first modify the directory shown to be the location where you stored the VCDB JSON files (see Listing 7-6).

LISTING 7-6

remove

```
#Listing 7-6
# requires package : verisr
# set this to where VCDB incidents are stored
jsondir <- 'data/vcdb/'
# create a veris instance with the vcdb data
vcdb <- json2veris(jsondir)
```

This should load fairly quickly, but on slower machines it may take a moment or two. If you're on a computer with just a few gig of RAM or if VCDB grows exponentially, you might not be able to load all of them into memory. (We've loaded over 100,000 incidents into 8G of RAM with `verisr`, so you shouldn't hit that limit anytime soon.) After you load this data, it's time to get to know the data a little bit. Let's begin with the `summary()` command (see Listing 7-7). The `verisr` package implemented its own `summary()`, so the output is very specific to VERIS data.

LISTING 7-7

remove

```
#Listing 7-7
# requires package : verisr
# requires object: vcdb (7-6)
summary(vcdb)
## 1643 incidents in this object.
##
```

```
## Actor:
## external internal  partner  unknown
##      955      535      100       85
##
## Action:
##    error  hacking  malware   misuse physical   social
##      398      416       42      216      508       31
##
## Asset:
## Kiosk/Term  Media  Network   Person   Server  Unknown   User Dev
##         17    534        8       33      656       80        429
##
## Attribute:
##  confidentiality     availability  confidentiality     integrity
##                2              614             1604           165
```

If you've grabbed the latest data from VCDB, you'll undoubtedly see different numbers than these.

JSON Notation

It might take a while to get used to the naming structure in JSON and understand how the variables are accessed in different settings. If you load VERIS JSON data into a mongo database, you'd use JavaScript to query the data and leverage a dot-notation approach to the variables. That dot-notation is used in the `verisr` package since the fields are referenced and retrieved by passing in character strings. This means you can access the top-level action data by referencing `action`. If you want to access the social section within the action, you reference `action.social` and the variety data under that is `action.social.variety`. Take some time and look at the JSON, and then try writing some code in R using `verisr`. With this hands-on experience, the dot-notation method will become second nature.

comma

There are two high-level functions from the `verisr` package that you'll use to dig into the data. The first is a function to create a filter so you focus on certain aspects of the data. The second is a flexible function called `getenum()`, which will get the enumerated data from the data set with a variety of options and extensions. Let's start by looking at the actors. You can replicate the information in the previous summary with the following bit of code (Listing 7-8).

LISTING 7-8

remove

```
#Listing 7-8
# requires package : verisr
# requires object: vcdb (7-6)
actors <- getenum(vcdb, "actor")
# actors is a data frame
print(actors)
##       enum   x
```

continues

Listing 7-8 *(continued)*

```
## 1 external 955
## 2 internal 535
## 3  partner 100
## 4  unknown  85
```

Within this data frame, you can see the raw numbers, but that isn't very helpful. Some incidents will contain multiple actors, so you can't simply add them and get a total number of incidents. Luckily, the `getenum` function can also return the total number of incidents where the field is defined. If you add the `add.n=TRUE`, you get an additional column of the full sample. If you add `add.freq=TRUE`, you can the percentage associated with each entry. Let's look at both of those options in one example (see Listing 7-9).

delete: the

get

LISTING 7-9

remove

```
#Listing 7-9
# requires package : verisr
# requires object: vcdb (7-6)
actors <- getenum(vcdb, "actor", add.n=TRUE, add.freq=TRUE)
print(actors)
##       enum   x    n  freq
## 1 external 955 1643 0.581
## 2 internal 535 1643 0.326
## 3  partner 100 1643 0.061
## 4  unknown  85 1643 0.052
```

From this you can see that there were 1,643 incidents with something defined in the actor section, and external actors were present in 58 percent of them. Since this function returns a data frame, it's relatively straightforward to feed into the `ggplot2` library and produce any number of visuals (see the book's website, `www.wiley.com/go/datadrivensecurity`, for this chapter's R code to see how to create the figures in this chapter).

The `getenum()` function is quite versatile. You can pass in any of the variable names within the VERIS framework and get an object you can visualize right away. As an example, create a function that accepts a VERIS variable name, such as `action.hacking.vector`, and returns an image object that you can print or save or whatever (see Listing 7-10). This could be extendable to include in a report or dashboard.

LISTING 7-10

remove

```
#Listing 7-10
# requires package : verisr, ggplot2
# requires object: vcdb (7-6)
library(ggplot2)
# take in the vcdb object and the field to plot
verisplot <- function(vcdb, field) {
  # get the data.frame for the field with frequency
  localdf <- getenum(vcdb, field, add.freq=T)
  # now let's take first 5 fields in the data frame.
  localdf <- localdf[c(1:5), ]
  # add a label to the data.frame
  localdf$lab <- paste(round(localdf$freq*100, 0), "%", sep="")
  # now we can create a ggplot2 instance
```

```
  gg <- ggplot(localdf, aes(x=enum, y=freq, label=lab))
  gg <- gg + geom_bar(stat="identity", fill="steelblue")
  # add in text, adjusted to the end of the bar
  gg <- gg + geom_text(hjust=-0.1, size=3)
  # flip the axes and add in a title
  gg <- gg + coord_flip() + ggtitle(field)
  # remove axes labels and add bw theme
  gg <- gg + xlab("") + ylab("") + theme_bw()
  # fix the y scale to remove padding and fit our label (add 7%)
  gg <- gg + scale_y_continuous(expand=c(0,0),
                                limits=c(0, max(localdf$freq)*1.1))
  # make it slightly prettier than the default
  gg <- gg + theme(panel.grid.major = element_blank(),
                   panel.border = element_blank(),
                   axis.text.x = element_blank(),
                   axis.ticks = element_blank())
}
```

What's a little funny about that function is that it will get all of the data ready in the first line of the function, trim to the top five entries in the second line, and spend the rest of the function making a pretty picture. But once this function is written and loaded, you can create any number of pictures from the data with a single line of code.

```
print(verisplot(vcdb, "action"))
```

Figure 7-5 shows a few of the possible values passed and printed.

Getting the Most Out of VERIS Data

One of our favorite images from the 2013 Verizon Data Breach Investigation Report was a heat map that compared the assets and actions overall and then separated the individual comparisons by the type of threat actors. You can create a similar image with the `verisr` package without too much effort (Listing 7-11).

heatmap (style sheet)

Listing 7-11

remove

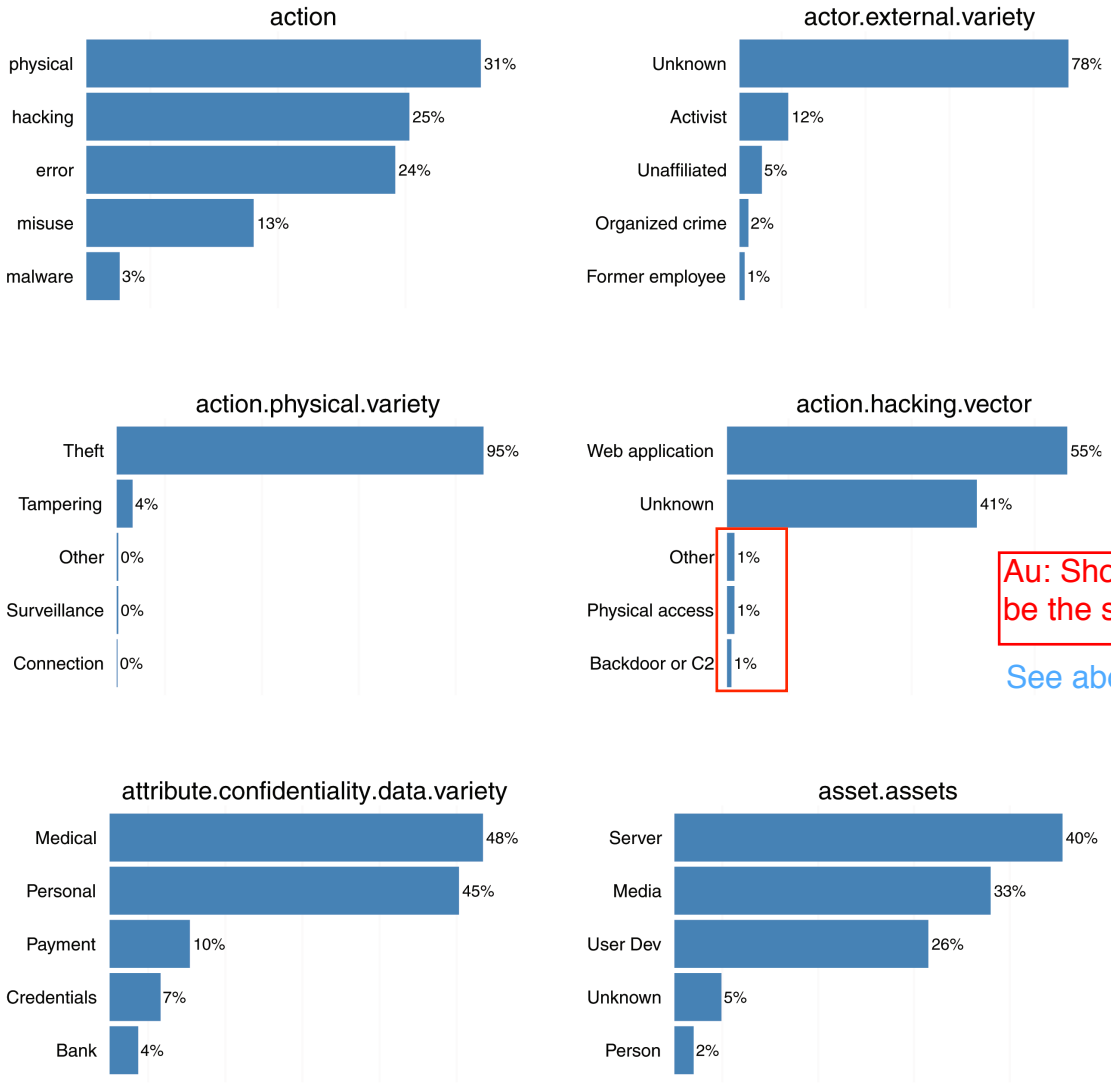
```
#Listing 7-11
# requires package : verisr, ggplot2
# requires object: vcdb (7-6)
# get a data.frame comparing the actions to the assets
# this will add zero's in missing squares and include a frequency
a2 <- getenum(vcdb, enum="action", primary="asset.assets", add.freq=T)
# trim unknown asset and environment action for space
a2 <- a2[which(a2$enum!="environmental" & a2$primary!="Unknown"), ]
# so we should create a "slim" version without zeros to color it
slim.a2 <- a2[which(a2$x!=0), ]
# could sort these by converting to factors (we did in Fig 7-6)

# now make a nice plot
gg <- ggplot(a2, aes(x=enum, y=primary, fill=freq))
gg <- gg + geom_tile(fill="white", color="gray80")
```

continues

Listing 7-11 *(continued)*

```
gg <- gg + geom_tile(data=slim.a2, color="gray80")
gg <- gg + scale_fill_gradient(low = "#F0F6FF",
                               high = "#4682B4", guide=F)
gg <- gg + xlab("") + ylab("") + theme_bw()
gg <- gg + scale_x_discrete(expand=c(0,0))
gg <- gg + scale_y_discrete(expand=c(0,0))
gg <- gg + theme(axis.ticks = element_blank())
# and view it
print(gg)
```



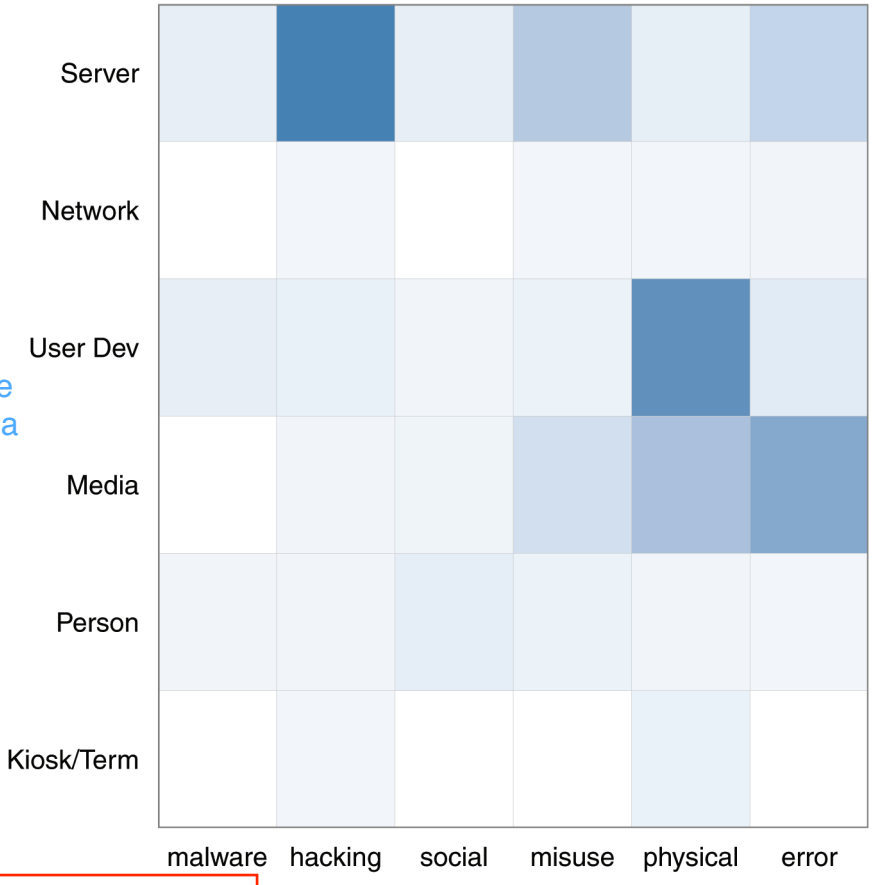
Au: Should 1% bars all be the same size?

See above, this is fine.

FIGURE 7-5 *Various top 5 views of VCDB data*

heatmap

This code looks through all of the incidents and produces the simple colored heat map shown in Figure 7-6. Keep in mind that the specifics vary depending on incidents in the VCDB.



This image seems quite large, can it be resized down a bit?

FIGURE 7-6 *A2 grid comparing assets and actions*

Figure caption aligned OK?

Or is the image misaligned?

The real benefit of working with VERIS data is the ***ability to compare across disparate data sets***. If you were to collect your own internal incidents in the VERIS format, it would be a relatively trivial task to run comparisons on very specific slices of data across multiple data sets. Since one of us works for Verizon and has access to the DBIR data set, we decided to show this point by example. You should be able to quickly see differences across the two data sets. Remember, VCDB is collected from news articles and various public sources. Generally speaking, the details are far less prevalent than what you might want.

The Verizon data set is gathered from a variety of primary sources, but primarily from first-hand accounts of the forensic investigators that were brought in after the security event. This means this data has bias—it is generally limited to breaches that were complex or big enough for a victim to seek external help, either from law enforcement (many of the contributing partners are law enforcement) or from an incident response consulting company.

In this example, you could use the same code that generated Figure 7-6 and compare four different fields from the VCDB data and the Verizon DBIR data over the last 3 years. Start with all of the incidents in both data sets in the first row. Then filter out confirmed data loss events (where `attributes.confidentiality.data_disclosure = "Yes"`) in the second row. Then focus on financially motivated attackers with confirmed data loss events in the third row. Finally, look only at attackers motivated by ideology, curiosity, fun, or pride (which covers attackers labeled as activists), again with confirmed data loss. See Figure 7-7.

three

rebreak & delete hyphen

You can see a rather significant difference that's worth talking through. Because the VCDB data set includes only publicly disclosed events, there are a lot of daily "low hanging fruit" type things that would never make it into the DBIR data. Events like lost or stolen laptops, documents tossed in a dumpster without being shredded, or envelopes with personal information mailed to the wrong person appear quite often in the VCDB data set. That's why you see physical (theft/loss) and error (disposal error and bad delivery) in the row labeled "All Events" for VCDB, and those all but disappear when you filter for confirmed data loss in the second row for "Confirmed Data Loss." Keep in mind a lost or stolen laptop just has the potential for data loss.

tr

Another interesting comparison is the malware category. The public disclosures rarely mention whether malware was used, but we know from the DBIR research that malware is often used, either to escalate privilege or to capture and exfiltrate data, yet the malware column is almost completely empty in the VCDB data. You will probably see the same type of effect for user devices. Even though user devices are often leveraged during a breach, a company that is publicly disclosing information will often neglect to mention which assets were involved and just say something vague like "the database was compromised." As a result, you'll see very little recorded events involving user devices.

We could go on and on about the subtle differences across these columns, but it's pretty clear that there is a lot to be learned by recording and comparing breach data.

Summary

You may never be able to shake the "blame the victim" mentality when it comes to data breaches. This means the victims will always try to be discrete and focus only on getting back to normal. You may always be fighting for more disclosures and more data when it comes to security breaches. That data is exactly what you need though, because these events produce a very rich set of data that has yet to be fully explored.

When you break the event down into its atomic components—"Who did what to what (or whom) with what result?"—you can do more research, develop better comparisons, and learn much more than if you apply a label or two on the whole chain of events. Identifying and recording the actors, their actions, the assets involved, and the attributes affected are a very good start. But remember—every data point comes with a cost and you will have to make some tough trade-offs between the time investment and the veracity of the results.

Comp: don't split para

Using JSON has some direct advantages. You can quickly load it into a variety of languages and it feeds right into databases that can take JSON (like MongoDB). Within R you can use the `verisr` package to read in VERIS data and rapidly analyze fields and create visualizations. But the real strength of leveraging

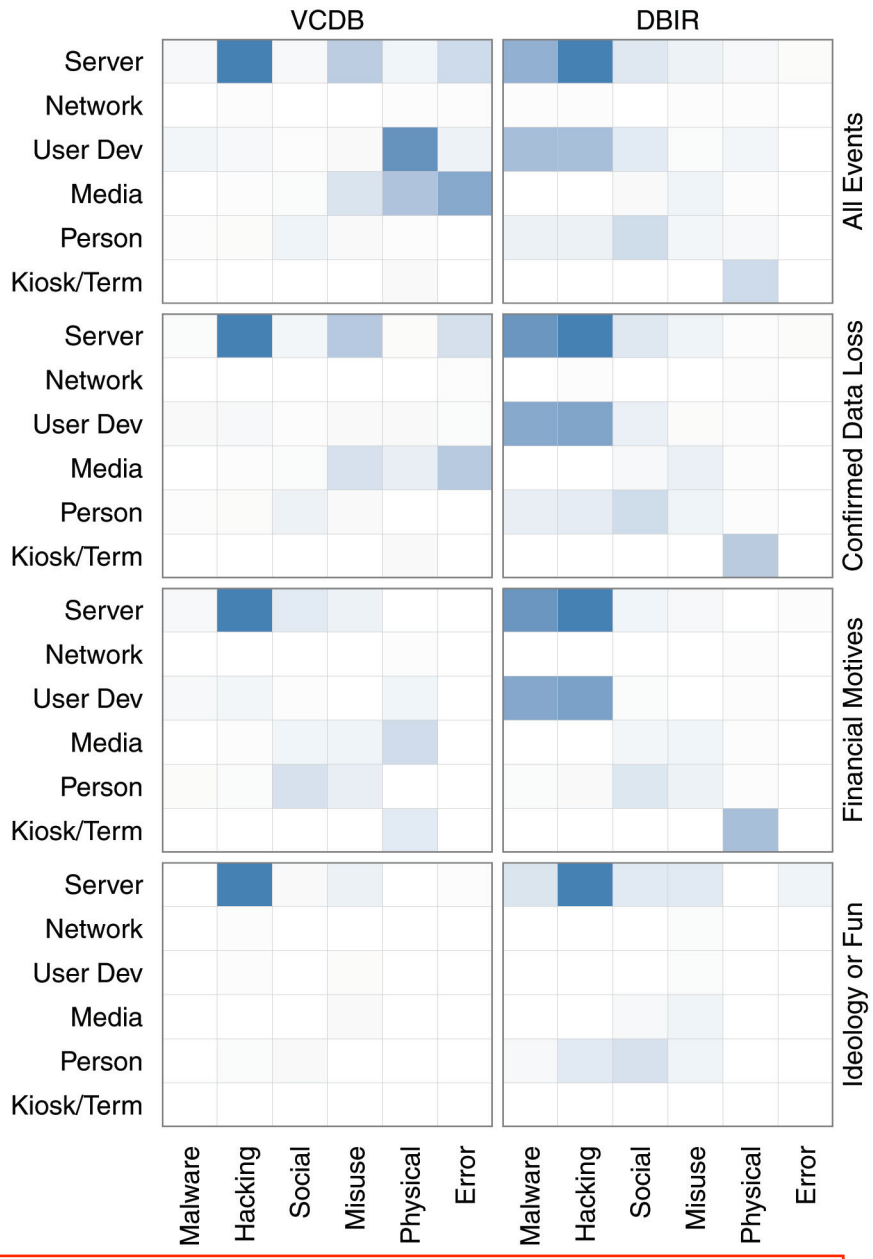


FIGURE 7-7 *The strength of VERIS: Comparing the same views from two very different sources of data*

Set fig.7-7 before Summary?

yes, move up one page into previous section

Figure caption aligned OK?

Comp: don't split para

a framework like VERIS is when making comparisons. Are your problems unique? Or are others in your industry or across all industries seeing the same trends and attacks? Until recently, analysts struggled to answer those questions, but as more organizations take a data-driven approach to security, they'll be asking and answering those questions soon.

Recommended Readings

The following are some recommended readings that can further your understanding on some of the topics we touch on in this chapter.

rebreak

comma x2 / ?

Verizon RISK Team. "2013 data breach investigations report." Available at `http://www.verizonenterprise.com/DBIR`—This report is based on data collected using the VERIS framework. You might find it handy to look at some of the graphics in the report and then attempt to repeat them using the `verisr` package discussed in this chapter and the VCDB data.

`http://veriscommunity.net`—All of the documentation about VERIS and a mailing list is hosted at that website. If you are uncertain what a field is, or what one of the options represents, this is the place to check first.

`https://github.com/vz-risk/VCDB`—This is the location for the VCDB data (at the time of this writing). Be sure to check back often as new events are added regularly.