10 Rules for an Unhackable Data Vault

by James B. Morris

Editor's Introduction

Most recent publicity on cyber security has focused on preventing attacks by external hackers. While many of these attacks began with an insider, there has been much less discussion about preventing malicious insider exploits. Perhaps that is because untrustworthy insiders are hard to find and block before they strike. The Secure Data Vault (SDV) is an approach to protecting the most sensitive data from malware and insider exploits. Formal verification of the microservices that govern access to the vault will close down almost all malware pathways. The old military N-person rule will close down most insider pathways. This rule allows for a trade-off between security and convenience: the higher the number who have to cooperate to access the vault (N), the greater the security and the less the convenience. When based on this plus nine other construction rules, the SDV will protect sensitive data from malware and malicious insiders.

10 Rules for an Unhackable Data Vault

by James B. Morris

In a presentation entitled “Perspectives on Security” at the 2015 Symposium on Operating Systems Principles (SOSP), Butler Lampson said the following:

In the real world, good security is a bank vault:

- *Hardly any computer systems have anything like this*
- *We only know how to make simple things secure*

When I read the item “Hardly any computer systems have anything like this,” I wondered, Why not? It is unlikely that Lampson was proposing we should build some sort of computerized bank vault. He was probably just making an analogy with the real world, where bank vaults provide security for our most valuable items, something we obviously don't have on the Internet for our most valuable and sensitive data.

Can we even build computerized data vaults that are "unhackable" by both external hackers and malicious insiders? I believe the answer to that question is YES. The Secure Data Vault (SDV) described here will be "unhackable" because (1) it will make the risk (and expense) of an external hacker's theft far greater than the reward and (2) it will reduce the probability of an insider theft to very near zero. The 10 rules outlined here are intended to be a guide for the development of next-generation, trustworthy systems (including SDVs) that will truly be "unhackable."

Attacks by malicious insiders are generally much more difficult to prevent than attacks by external hackers. This article does not make much mention of how to prevent attacks by external hackers, because the success of the [DARPA HACMS project](#) has proven that external hackers can be stopped using technology that is mostly available today. This article mostly discusses how to stop malicious insiders, which has not been studied or discussed much, but needs to be. In particular, this article will suggest that POLA and other access controls have generally been failures, and will further suggest that the N-person rule (enforced by system software) might be a much more effective security mechanism for preventing insider attacks.

The best vehicle to provide a forum for a comprehensive discussion on deterring attacks by malicious insiders is the [Data Security Advisory Group](#).

What is a Secure Data Vault (SDV)?

We will explore the vision of an SDV as a microservices architecture built on a minimal trusted computing base (TCB) that would include the [seL4 microkernel](#) configured as a hypervisor with the formally verifiable properties of being both malware-secure and insider-secure. A system is defined to be malware-secure if and only if malware cannot be externally installed into the system (equivalently, the system cannot be externally infected by malware). A system is defined to be insider-secure if it is very effective at eliminating theft of sensitive data by malicious insiders. Thwarting most insider threats can ultimately be achieved, but providing formal specification and verification that a system is secure from insider threats may not be possible. Instead, insider threats might have to be countered by a cleverly designed set of practical, security-control rules.

Figure 1 is a proposed diagram of an SDV comprised primarily of a ring-structured Database Management System (DBMS). This is what the structure of an SDV would look like if the DBMS used is Apache Cassandra, for example. Cassandra is just one example of a DBMS that might be used internally within an SDV, but there are other candidates. In fact, just about any modern DBMS with a set of features similar to that of Cassandra would suffice for the core component of an SDV.

Figure 1.

An SDV must be a relatively simple, secure solution, which means it can't be built using the overly complex, monolithic monsters that most Windows and *nix systems have become today. Recall that Lampson said, "We only know how to make simple things secure." Each of the blocks in Figure 1 is a [microservice](#), although there will be support microservices that are not shown. Every microservice (that needs one) contains its own database. Each microservice is an independent, simple process virtual

machine (for example, a Java Virtual Machine, or JVM). The seL4 microkernel, configured as a hypervisor, manages the multiple JVMs, including all inter-process communication between the microservices. See Figure 2, in which the client microservice implements a Secure Access Point (see above) and a server microservice implements a DBMS node (a DDN in Figure 1).

You might be inclined to argue that an SDV cannot be a *simple* solution by its very nature. However, the idea of microservices can help to make the overall complexity of an SDV manageable. If the overall system architecture consists largely of independent microservices, then the isolation guarantees provided by the underlying seL4-based operating system allows one to deal with the microservices mostly in isolation, dramatically reducing the difficulty of getting each microservice correct. If each microservice is itself well architected, with a minimal TCB (as discussed above), then formal verification of the critical parts becomes tractable. It's basically the time-honored divide-and-conquer approach.

An SDV doesn't need a platform VM (virtual machine) or an OS-level VM. A process VM (e.g. a JVM) has almost everything an SDV microservice needs, and the system services implemented on top of the seL4 microkernel will supply what the JVM needs but doesn't have. Each microservice is a virtual machine because the microservices need to be encapsulated and isolated from each other for better security. The seL4 microkernel provides the necessary isolation. A microservice needs to be simple, because we will need to formally verify the virtual machine, and we can't formally verify a "full" virtual machine because it is far too complex. Simplicity is critical for good security.

Figure 2.

The following four technical requirements will need to be considered in a specification for an SDV.

1. The executable resident code of a microservice cannot be changed, except by a secure update of the executable resident code in its entirety
2. The executable resident code of each microservice is constantly size-monitored and checksum-monitored to detect any changes in the executable resident code
3. Stored data can change, but stored data will never be executed
4. Monitoring features, logging features, and the N-person rule (see below) will eliminate theft of data by malicious insiders

An SDV must be adequately protected from both external hackers and malicious insiders.

The following 10 rules are guidelines for building, deploying, and using an SDV.

Rule Number 1: An SDV must be structured as a microservices architecture that can be decomposed into microservices that are simple enough that they can be individually verified using formal methods.

Rule Number 2: Each microservice must have only one purpose (or goal or function). This aids in the development of relatively simple microservices.

Rule Number 3: An SDV must guarantee (by formal verification, if possible) that it is immune to external infection by malware.

Rule Number 4: User authentication (e.g., user “log-ins”) must rely on multi-factor biometrics.

Rule Number 5: The difficulty and inconvenience of retrieving sensitive data from an SDV should be directly proportional to the amount of sensitive data to potentially be retrieved.

Rule Number 6: The number of humans (at least two in the case of least-sensitive data) required to simultaneously submit a request for retrieval for any and all sensitive data from an SDV should be directly proportional to the sensitivity level of the data. This is similar to the [Two-Man Rule](#), which requires two humans to take simultaneous action in order to launch a nuclear weapon. Inconvenient, but necessary.

Rule Number 7: Security (access controls) must be automated as much as possible, and it should be extremely difficult for humans to disable or relax security protections.

Rule Number 8: Manual security (access-control) configuration required by humans should be kept to a minimum, or, if possible, should even be non-existent.

Rule Number 9: Software updates should update a microservice in its entirety, the update must be secure, and the update should be accomplished remotely “over the air,” as is done with all smartphones today.

Rule Number 10: Network communication (over a path that contains insecure computers) between microservices that exist on two physically different computers in the network must be encrypted.

Rule Numbers 1, 2, and 3 and the four technical requirements given above will be necessary to ensure that an SDV is protected from data loss or data damage by external hackers.

But what about malicious insiders? This is arguably more difficult to prevent than external hacking, and from this point forward this article will discuss how to stop malicious insiders.

Let's take a closer look at Rule 5 and 6, because these two rules are critical in preventing successful attacks by malicious insiders.

Rule Number 5: The difficulty and inconvenience of retrieving sensitive data from a secure data repository should be directly proportional to the amount of sensitive data to potentially be retrieved.

Rule Number 6: The number of humans (at least two in the case of least-sensitive data) required to simultaneously submit a request for retrieval for any and all sensitive data from a secure data repository should be directly proportional to the sensitivity level of the data. This is similar to the Two-Man Rule (more on this below), which requires two humans to take simultaneous action in order to launch a nuclear weapon. Inconvenient, but necessary.

In his talk at SOSP 2015, Butler Lampson also said, “Access control doesn’t work—40 years of experience says so. The basic problem: its job is to say ‘No’. This stops people from doing their work, and then they relax the access control—usually too much, but no one notices until there’s a disaster.” It is not quite correct to say that access control doesn't work. It is the relaxation of access controls that results in access controls not working.

One method of access control that has been popular for many years is the [Principle of Least Authority \(POLA\)](#), which requires that "every module (such as a process, a user, or a program, depending on the subject) must be able to access only the information and resources that are necessary for its legitimate purpose."

I have several objections with the POLA and similar types of access controls, as follows:

1. POLA controls must be manually configured for each user
2. It is almost universally true that POLA access controls can be easily relaxed, and it is done all the time by inconvenienced users
3. It is usually the case that a *single* system administrator can change any part of a POLA configuration

Objection number 1 arises because manual configuration of access controls is a tedious, error-prone configuration process that must be handled by highly skilled system administrators. Errors in setting POLA configurations can have disastrous consequences. Refer to Rule Numbers 7 and 8 above: it would be best if manual configuration of access controls by humans (especially a human acting alone) can be kept to a minimum in an SDV.

Objection number 2 is the problem that Lampson warns us about. As General B.W. Chidlaw said in 1954, “If you want security, you must be prepared for inconvenience.” The best way to prevent insider theft of sensitive data is to make it very inconvenient to extract the sensitive data. That's going to mean a big change for most users of an SDV in the future, because computers have brought a lot of convenience into our lives and most people don't like to be inconvenienced. General Chidlaw was likely right, however, and, like it or not, we are going to have to embrace his point of view if we want systems that are secure, especially from malicious insiders. Expect to be inconvenienced if you potentially have access to sensitive data, and the more sensitive the data, the more likely you are going to be inconvenienced. If Rules 7 and 8 are followed, then access controls like POLA would never be relaxed. It's an issue of convenience, of course. Humans relax access controls like POLA because they want convenience.

Objection number 3 exposes the real threat to good security. Humans cannot be trusted to be honest at all times. Just because someone has worked at an organization for twenty years or holds a top-secret clearance, you can never know what is happening in that person's life at any point in time. Any of a number of different scenarios can turn an honest person into a thief, at least for a while. In an

organization of a thousand people, there are probably at least 2–3 people at any one time who are willing to be dishonest. But which 2–3 people? Answer? Suspect everyone at all times.

Rules 5 and 6 are just another form of access control, but they are a relatively new and relatively untried type of access control. They are access control based on the amount of data to be extracted and on the Two-Man Rule. First of all, we need to modernize the name of this rule to the “Two-Person Rule.” We also should generalize it to be called the “N-person Rule” ($N > 1$).

The Two-Person Rule is used in several different kinds of situations where security must be very strict, so this rule is not totally new to situations where high data security is required. Below are some links to articles you might want to read about the Two-Person Rule applied to data security:

- [“Bring a Friend: The 'Two-Man Rule' & What It Means for Data Center Security”](#)
- [“Why the ‘two-man rule’ is only the beginning”](#)

The NSA apparently got the message loud and clear too, but not until after the Snowden Leak. (See the article entitled [“NSA Implements 'Two-Man Rule' to Prevent Future Leaks.”](#))

Rule 6 is an attempt to apply the N-Person Rule to the case of data security. The N-Person rule will be enforced automatically by the operating system and other software in an SDV, never manually by humans merely as a result of organizational policy. In the article above about the NSA implementing the Two-person rule, it appears that the implementation of the rule is left up to humans to enforce, instead of being enforced by a software system. Depending on humans to follow orders or be honest 100 percent of the time is not the way to implement good data security.

The N-person rule introduces a probabilistic notion of security. The rule assumes a human will betray the security of an SDV with a probability of no more than p (which is presumably a function of what is happening in the person's life at any point in time, and is not likely to be a constant). If an access to sensitive data requires two people to access the data, with betrayal probabilities p and q , then the overall risk of betrayal is reduced to $p*q$ (although in reality it's probably higher, as no two people in the same organization are truly independent). If a three-person rule is used, the risk of betrayal is reduced even further to $p*q*r$. As N increases in the N-person rule, the risk of betrayal becomes less than the probability of a giant meteor striking the earth, which will make computer security irrelevant to humans anyway.

Let's look at an example of accessing an SDV using the N-person Rule. Suppose that every document (or atomic database entry) in the database has a sensitivity classification of level 1–5, with level 5 being the most sensitive. As a side note, the N-person rule should always be used whenever document classifications levels are assigned or modified; one person acting alone should never be able to assign or modify classification levels of sensitive data.

The value of N in the N-person rule is a function of the sensitivity level of the data and the amount of data to be extracted from an SDV. Specifying the algorithm for calculating N is a matter of policy that

must be set by management, and it should be obvious that the N-person rule should be strictly followed when setting or amending the N-person-rule calculation policy. In addition, the N-person rule should be strictly followed among system administrators whenever the POLA configuration is changed for any user of an SDV. One system administrator should never be able act alone.

Here are some examples of accessing an SDV using the N-person rule.

1. Request for level 1 document with a total of 5K bytes of data: Requires two people to submit the request ($N = 2$).
2. Request for level 5 document with a total of 100K bytes of data. Requires three people to submit the request ($N = 3$).
3. Request for several level 3–5 documents with a total of one gigabyte of data. Requires five people to submit the request ($N = 5$).

Perhaps there are cases in which the N-person rule can be relaxed to allow one person to access sensitive data at certain times of the day for a certain period of time. For example, a doctor with appointments every half-hour can access sensitive data (only) for the patient he or she is seeing for that 30 minutes during the time the patient has a scheduled appointment. Or, maybe a better way to handle the doctor situation is to use a two-person rule in which a doctor and a nurse (or security monitoring person) must both request access to sensitive data at all times.

Conclusions

Providing bullet-proof security against malicious insiders will undoubtedly require both the POLA and the N-person rule in order to access sensitive data in an SDV. If one person is allowed to access an SDV for any reason, you are asking for trouble. Depending on the perceived value of the sensitive data stored in an SDV, the cost of the inconvenience and extra manpower to provide bulletproof security in an SDV will be far less than the cost of the data in the SDV being compromised. If this is not the case, one should probably use currently available, insecure data storage instead of an SDV.

There have been at least four insider thefts of data within the intelligence community in the last few years: Edward Snowden, Chelsea Manning, Harold Martin, and a person or persons unknown who perpetrated the CIA theft and subsequent release to Wikileaks. Those are just the ones we know about for sure. I have heard rumors that the FBI arrested a fifth person a few months ago, but I don't know whether this is true or not. The U.S. government's reaction to these insider thefts has been sort of like a deer standing on the road and staring into the headlights of an oncoming truck, i.e. paralysis. The only reaction we get from the government after a major sensitive-data theft is, "Let's catch the bad guys and throw 'em in jail." That's no better than closing the barn door after the horses have escaped. It's time for the government to fund research that will fix the problem. The SDV described here is a reasonable approach that should be investigated, developed, improved until it works well, and then commercialized so that organizations can put a stop to the external hackers and malicious insiders for good.

About the Author

Jim Morris is a software developer, computer systems architect, businessman, and serial entrepreneur with more than 40 years of experience, most recently at FullSecurity Corporation researching solutions for microservices architectures and secure data-storage systems that are immune to external infection by malware and resistant to theft by malicious insiders. He has started and sold a successful technology company, did very early research (1970s) in OO programming languages at the Los Alamos National Laboratory, and was an associate professor of computer science at Purdue. His areas of interest are cryptography, software development, operating systems, and hacking methodology. He has a Ph.D. in computer science and a B.S. in electrical engineering, both from the University of Texas at Austin.

DOI: 10.1145/3081882