Postmortem

What to do after a security incident.

PREVIOUS
Shawn Powers'
The Open-Source
Classroom

NEXT
New Products

SUSAN SONS

Susan Sons serves as a Senior Systems Analyst at Indiana University's Center for Applied Cybersecurity Research (http://cacr.iu.edu), where she divides her time between helping NSF-funded science and infrastructure projects improve their security, helping secure a DHS-funded static analysis project, and various attempts to save the world from poor information security practices in general. Susan also volunteers as Director of the Internet Civil Engineering Institute (http://icei.org), a nonprofit dedicated to supporting and securing the common software infrastructure on which we all depend. In her free time, she raises an amazing mini-hacker, writes, codes, researches, practices martial arts, lifts heavy things and volunteers as a search-and-rescue and disaster relief worker.

INCIDENTS HAPPEN. Vulnerabilities happen. The quality of your response can make the difference between a bad day and a disaster. What happens after the response can make the difference between endless firefighting and becoming stronger with every battle. A quality postmortem analysis is free ammunition. Every incident is someone or some event showing where a system's weaknesses are, if only one is willing to listen.

This is how a good information security officer, or an engineer who's a true information security evangelist, can make a difference:

1. Something happens. It may be an exercise, or a real incident.

2. You now have real information to go on. You are in a very different position from when you were working from the theoretical.

3. *If* you know how to understand that information, and what information you need, you may have a new

Postmortem mistakes can have long-term implications, but they also can take a long time to identify. A bad postmortem feels just as satisfying as a good postmortem to someone who doesn't know the difference.

understanding of the project or organization's security needs. Even if this is only confirmation of what you knew before, it is important because...

4. This information and analysis, if communicated effectively, especially in the aftermath of an incident, can be a powerful tool for fixing problems.

5. Next time around, the organization will be a little more on its game, and another set of weaknesses can be shored up. Every iteration makes the organization stronger.

How to Sabotage Your Postmortem

Postmortem mistakes can have long-term implications, but they also can take a long time to identify. A bad postmortem feels just as satisfying as a good postmortem to someone who doesn't know the difference. Unfortunately, it fills a team—or a whole organization—with false beliefs, missed opportunities and bad data, eroding its ability to mature its security. These erosions are small individually, but like water lapping up against a beach, they eventually aggregate. Learn these anti-patterns and be certain to recognize them.

Play the blame game. Yes, some incidents are clearly one person's fault. Most of the time though, there's plenty of blame to go around. Blame is an out that makes it too easy to ignore systemic problems, and looking for someone to punish makes valuable sources of information go silent. Deal with personnel issues separately from incident postmortem, except in cases of actual malicious insider attacks.

Stop at the vulnerability. Calling it quits once you've found something to patch or a configuration to change is perhaps the most common mistake. In the best of cases, looking deeper can confirm what's working and what isn't. In the majority of cases, there is more than one cause to be found. Don't stop looking once you've found something; poke in all the corners first.

Stop at the forensics. Another common mistake is to look for signs of technological vulnerability or compromise, such as incorrect firewall configurations, software bugs, rootkits and so on without looking at the bigger picture of what may be causing those things to happen. Poor software engineering practice or inadequate tools for engineers will raise the incidence of bugs. So will overwork and poor morale. Similarly, a lack of configuration management for systems can cause mistakes due to forcing administrators to repeat processes by rote many times.

What Actually Works

See failures as information. Every failure, including not having enough information to do a proper postmortem, is itself information. Do not lose sight of this. If you find yourself at a loss in a postmortem, start looking at what you would have needed to do a postmortem that you don't have. That is your first lesson learned.

Treat "root cause" as an adjective. There's never only one root cause, because if there is only one root cause, the other root cause is "we failed to practice fault tolerance by implementing defense in depth". Root cause analysis is the act of finding root causes, plural, not the search for a single root cause.

Go back to first principles. In my day job at Indiana University's Center for Applied Cybersecurity Research (http://cacr.iu.edu), we've been working on a set of seven principles from which cybersecurity in general can be derived (http://cacr.iu.edu/principles). First principles work in reverse as well: they are not only a tool for performing information security, but also for figuring out how information security failed.

- **Comprehensivity.** Was there a system no one knew about? Was a risk being ignored? Comprehensivity failures tend to be failures of scope.

- **Opportunity.** Did something go unmaintained because the burden was placed on under-resourced in-house staff instead of using well maintained common tools? Were staff under-trained so that they didn't recognize something they should have? Was no one staying abreast of current threats?

- **Rigor.** Was the organization caught out by assumptions that weren't being verified? Did monitoring fail? Was something not specified clearly enough to ensure that everyone was on the same page? Was automation not put in place to ensure that repetitive tasks were not done precisely and consistently across time and space?

- **Minimization.** Was something a bigger target than it needed to be? Were there more ways in, or more moving parts, than there needed to be? Could something become easier to protect by eliminating or shrinking some part of it?

- **Compartmentation.** Did someone or something have access that it didn't absolutely need? Did isolation fail? Was cryptography not implemented appropriately? Were monolithic systems and processes used when things could have been segmented from one another? Were interfaces between systems or components of systems unclear or overly complex?

- **Fault tolerance.** Was there a single point of failure? Was there a credential that wasn't cheap and easy enough to revoke, so it wasn't replaced when it should have been? Was something built or configured with the assumption that bad things wouldn't happen to it?

- **Proportionality.** Was security, or any systems or software decision, made in isolation, without considering the environment as a whole? This one can be a killer—when security interferes with getting the job done, people will circumvent it. When security is too expensive, no one will implement it. When a business case hasn't been made relative to other risks, the organization won't know what security to invest in and may invest in none at all because doing *all* information security controls is untenable.

It is extremely hard to advocate for resources to be put into security in any organization, because resources are always limited and prevention is impossible to quantify.

It takes time to work with and learn to use the principles for analysis, but it's worth doing so. They are invaluable in flexing one's brain around whatever problem comes along, instead of learning types of problems one at a time. Each principle has much more to it than these brief examples, but the examples here should provide a starting point for how they may crop up in an incident postmortem.

Lessons Learned

Here are a few things I've learned through years of doing postmortem analyses.

There will be more bugs. There always will be more bugs. Sometimes the right answer really is "patch it and move on". However, one should not move on without asking whether one can become more robust. Could patching happen faster in order to prevent future compromises? Is there a secondary control that could be put in place so that a vulnerability in one component doesn't equate to a vulnerability in the system as a whole? How can fault tolerance be increased? Is there adequate monitoring for appropriate response? If the bug is in software you maintain, is the bug just a bug, or is it the result of engineering practices that are holding back your engineers or deeper architectural problems that should be cleaned up in a refactor?

Getting a patch out is nice, but eliminating classes of problems, rather than the one problem that came to your attention, is how a system or organization becomes more mature and more secure in a meaningful way over time.

An incident is proof, and proof is leverage. It is extremely hard to advocate for resources to be put into security in any organization, because resources are always limited and prevention is impossible to quantify. When there is an incident, you have something concrete in your hands, *if* you know how to use it effectively.

Do not fall prey to the temptation to make every incident into a moral panic. Overblown scare tactics just serve to deafen others to the security team's constant cries of disaster. Save that for when the sky really is falling. Instead, look at what the incident cost to mitigate or what specifically was headed off. Look at what underlying problems were revealed, and what the aggregate cost would be of more incidents like this if the underlying problems are not fixed. Speak in risk vs. reward, and have dollar figures and time estimates at hand, even if they are a bit rough. Think about other organizational costs and benefits too, including changes in time to market, personnel turnover, reputation, liability and so on.

Do not provide a laundry list. If you ask for more than the decision-makers can take in, you have lost them. Do not drown them in minutia. They want to hear about the details of the build system's server components about as much as you want to hear about the components of the next shareholder meeting report. Perhaps less. There are entire books on clear communication and working with management, so I won't try to reproduce them here. Please go read one or two of them.

Keep track of change over time. Security professionals, and technologists in general, tend to be problem-solvers by nature. We focus on things that are broken. Not only can this make us seem negative to others, but it can cause us to appear like we're treading water rather than truly making progress, sometimes even to ourselves. Each postmortem—whether of an exercise or actual vulnerability/incident—can be leveraged to spur incremental improvement. Getting the most out of this aggregate requires knowing what the aggregate is.

Concretely demonstrating how security has improved year over year will improve team morale, protect security resourcing and autonomy, help leadership see security as a concrete, tractable problem rather than an infinite and nebulous one, and help the person pushing the security envelope stay sane. Also, if that year-over-year picture isn't a solid improvement in maturity, it's better to know that so you can triage the problem. Whether it's a lack of support, lack of training or something else, find a way to do better. No security is perfect. Strive to do better than your past self.

A Final Note on Forensics

Readers may note that, apart from assuming that some data about the nature of an incident is available, this article didn't talk about doing forensics. The truth is that digital forensics is expensive and requires specialized skills. It's also useless in an organization that doesn't know how to use the resulting information. Master postmortem analysis based on whatever information you have available, and you will soon know when you need more information, and when digital forensics techniques make sense for your budget and the incident at hand.

Don't get blindsided by the shiny sound of forensic techniques before you know whether more rudimentary analysis will get you where you need to go. My life-critical technology projects will engage in digital forensics when it's called for; a $30k/year nonprofit project never will. I have yet other projects that may not have forensic resources themselves, but may cooperate with CERTs or ISACs as appropriate to help understand threats that are relevant to more than one organization.

Remember, the goal of a postmortem is to improve your defenses, not to answer every question. Real life is not a Sherlock Holmes novel. You don't always get a neat resolution with all loose ends neatly tied up. In fact, that almost never happens.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS