Security Exercises

Read on for a crash course on how to plan effective security exercises.

PREVIOUS
Dave Taylor's
Work the Shell

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.

REGULAR SECURITY EXERCISES are, bar none, the most powerful, cost-effective tool for maturing a project's information security operations—when done well. Unfortunately, courses and certifications on InfoSec tend to focus on how to implement specific controls or how to select some baseline best practices when starting from scratch. Little to no attention tends to be paid on how to test what you have and iterate on it. Prepare for a crash course.

Security Exercise? What's That?

A security exercise is a drill designed to propel a team or teams through the steps they would take in the case of a real or suspected information security problem in their organization or project. For example:

- Tell your ops team that the server hosting your internal bug tracker has experienced data loss due to a critical RAID controller failure. Have them rebuild the server from backups on spare hardware to show that the backups are viable, spare hardware available and the process known and workable.

- Start running an otherwise innocuous, but memory-intensive, piece of unauthorized software on a development server. See how long it takes for someone to notice and what he or she does about it.

Isn't This Dangerous?

Security exercises are not the first step in running an InfoSec program for a project of any size. The first step is coming up with a plan or set of policies appropriate to the size and complexity of the project. For a very small, all-volunteer open-source project, this may be as simple as "Our project manager, $name, accepts risk on behalf of the project and our information security officer, $name, is in charge in the case of a suspected security incident; the integrity of our code base will be prioritized first, confidentiality of yet-undisclosed vulnerability information second and availability of services third." For a larger, more complex organization with paid staff, this normally will start with a Master Information Security Policy and Procedures document supported by a number of other policy documents.

In either case, step one is establishing roles and responsibilities; step two is establishing operational expectations, and step three is testing that your policies, procedures and expectations *work*. If you aren't testing, you don't really know that it works.

Scheduling exercises at a predictable time and reminding others when it will happen prevents confusion among staff. It is wise to begin with low-impact exercises (more on this below) that don't leverage production systems, and move on to higher-potential-impact exercises only when the organization's infrastructure and personnel have had most of the bugs shaken out. If something as small as a runaway process on a single server can seriously impact your business, it's better to find out at a planned time with all hands on deck than at 4am on a holiday when no one who knows what to do can be reached. The whole point of security exercises is to increase resilience: raise the threshold of what is normal for your team to deal with, what your systems can shrug off.

Why Are Security Exercises Important?

When I respond to a security incident that's gone disproportionately bad—that is, far worse than the incident should have gone given the resources and security needs of the organization—it tends to be true that more than one thing has gone wrong. However, the root cause of how those things were all allowed to go wrong at once is almost one or both of two things: lack of interest in and support for information security from organization leadership, or the failure mode I call "death by supposition".

"Death by supposition" is when we make decisions based on "facts" that are *supposed* to be true, but have not been *tested* by us. For example, suppose that hardware or software will behave the way the vendor said it would. Suppose that anybody in the company remembers the incident response plan that was written, approved and stuck in a drawer two years ago. Suppose that the "best practices" written for companies in your sector don't overlook some way in which the sector has changed, or your company is unique. Suppose that your backups work the way they were designed to, and nothing has gone awry in the 25 updates since the system was put in place 18 months ago. Suppose that your VPN hardware fails closed the way the vendor's sales staff insisted that it would.

Supposition kills, and it's an insidious killer because, unlike bad leadership, it's easy to miss. We often aren't aware of our assumptions until something goes horribly wrong—better for that something to be a simulated attack than a real one, leading only to simulated consequences.

Security exercises, done right, will do the following:

- Reveal whether systems and technical controls (still) work as expected.
- Ensure that security, ops, leadership and other team members are on the same page.
- Reveal holes in procedures and policies.
- Provide your team with vital practice at operations that may someday need to be done quickly and/or under stress, especially disaster recovery and incident response procedures.

- Provide your team with *stress inoculation*. This is something that SWAT teams, martial artists, search-and-rescue teams, firefighters, military and so on already know is an essential part of their live drills: getting used to something so it doesn't register as such a large stressor any more.

- Provide non-security personnel and security personnel alike with valuable hands-on security training.

- Improve the relationships needed to make security improvements and incident response go more smoothly.

Most important, well-executed security exercises take your organization from the land of supposition to actually knowing where your weaknesses are, where your resources should be going, and what you are doing right. Don't guess. Know.

What Makes a Good Security Exercise?

Asking what makes a good security exercise a lot like asking what makes a good martial arts or search-and-rescue exercise. If you exercise only once or do only one kind of exercise, you won't get the results you are after. The right question is "What makes a good security exercise *program*?"

The answer is:

- Regularity.

- Purpose and focus.

- Attention.

- Follow through.

Good Security Exercises Happen Regularly

In a small organization without much in the way of infrastructure, run an exercise once per quarter. It doesn't have to be over-the-top in scope. Just make sure you are doing *something* regularly.

In a medium-sized organization with some dedicated IT resources and some in-house infrastructure to look after, run an exercise once per month. This gives you enough time to design the exercise well, do a thorough postmortem and integrate what you've learned into your security operations.

In a large organization with complex IT infrastructure, security exercises should be a near-constant affair, carried out within various units and across units with support from your security team. Consider building out some infrastructure to make exercises easy to run frequently.

These are rough guidelines only; use your brain and a little trial and error to find the right interval for your organization. Don't be afraid to run exercises when key people are missing. Often, real incidents happen at the least convenient time possible: when the security officer is on a long flight, when a needed systems administrator is out sick and so on. Get used to the unexpected.

Purpose and Focus Matters

If I listed all the security exercises I could think of, and your organization drew and ran a random one each month, you'd probably be better off than if you ran no exercises at all. However, exercises tailored to your organization and infrastructure are far more effective. Much of an exercise's quality comes from its planning.

What Are You Exercising?

Each security exercise may be exercising people, systems, policy and procedures, or some combination of the above. Note that I said "exercising" rather than "testing". Security exercises are most effective when they are used as a diagnostic and training utility rather than as a performance evaluation. Using security exercises as a performance metric for personnel tends to decrease the quality of collaboration and initiative during both real and simulated incidents. In the organizations with the most effective security programs, exercises pit team members against the exercise, rather than against one another or against an evaluation mechanism.

In organizations where security exercises are new, they often are broad—for example, "Can we restore this system from backup?" or "What do we do if our password database is leaked?" In organizations with more practice, exercises often are a mix of the broad, end-to-end kind with more targeted exercises that test specific capabilities, such as detection, ability to bring specific systems up or down smoothly, response to specific attacks and so on.

I keep a list of things I'd like to test via security exercises for each project/organization I'm responsible for. The list contains:

- Any issue for which we've argued whether the control we have is the "right" control.
- Any system or component we haven't tested recently (or at all).
- Any known vulnerability we *think* we've closed.
- Any known vulnerability we haven't effectively closed and to which I'd like to draw leadership or team member attention.
- Anything that looks like a single point of failure—including people.
- Any behavior we assume our team members will do, but haven't tested recently.
- Procedures for "black swan" events—potentially devastating security events that also are rare/unlikely enough that we have practice dealing with them only if we create that practice.
- Procedures that involve roles for which we've had personnel turnover.
- Procedures where I'm not sure it's clear who will be doing what task or job.

Prep

Once you've chosen your exercise focus, prepare for it. Unless your organization is *very* mature from a security standpoint, this will begin with setting a schedule and notifying everyone in the organization, then reminding them again just before the exercise starts.

The simplest security exercises are what we call *tabletop exercises*.

In a tabletop exercise, all the relevant parties sit down together and, in real time, walk through a hypothetical scenario noting everything that would be done in response to a problem. Tabletop exercises are the least informative type of security exercise because they lack realism, but they're also the most lightweight exercise to run.

A tabletop security exercise requires:

- An exercise scenario, written up in as much detail as possible and well understood by the person running the exercise.
- A way for everyone on the team to meet in real time: conference room, conference call, IRC, video conference—whatever works best for your team.
- All of the principals relevant to the organization's potential response to this scenario.
- Anyone else who would benefit from participation in the exercise.
- Excellent note-taking.

That's it. So, you now have no excuse not to at least run tabletop security exercises within your group.

Live exercises are a bit more involved, but they provide a wealth of information and experience to your team that can't be gotten in any other way apart from having something actually go wrong. There are, of course, degrees of "live-ness". It's acceptable—and often easiest on your team—if you start at the less-ambitious end, where you present a hypothetical then step through the resolution live, then progress to more involved exercises where problems are introduced on actual systems.

A live exercise will involve at least a Blue Team and a Red or White Team (possibly both). The Blue Team is the defenders. That team will be doing its job throughout the exercise much as it would during a real incident. The Red Team is the attacker. That team causes whatever problem sets off the exercise, and in advanced adversarial exercises, may continue to act throughout the duration of the exercise. The White Team is the referee. It may provide clarification throughout the exercise, observe and so on, but should not be actively participating (usually).

The most important part of preparation for a live exercise is setting and communicating the boundaries of the exercise. Consider the following:

- *Who should be informed that the exercise is going on?* Until your organization's information security is fairly mature, this should be everyone within the organization. Once live exercises are part of your regular routine, it can become fun to schedule some exercises in secret, telling only key members of the Red and White teams when it will happen, in order to get more realistic responses. Surprise exercises tend to end badly in organizations that aren't very practiced at running exercises, however, because if not planned carefully, they can backfire. Also, consider whether to inform anyone outside the organization. For example, you might want to warn your data center's staff (if you colocate) before running an exercise to prevent them from initiating a security incident upon observing suspicious traffic to/from your systems.

- *How much degradation of live services is acceptable due to an exercise, and how will you ensure that this limit is not exceeded?* Live exercises can be unpredictable. This is good, because real incidents are unpredictable. However, it is important to scope exercises so that they don't exceed the organization's tolerance for interruptions to normal service. Once you have a great deal of practice running exercises, you'll be able to play it closer to the wire, but while your organization is new at this, consider limiting security exercises to operating on non-production systems, and/or systems that are easily re-imaged after the exercise.

- *How will you clean up the mess after the exercise and ensure that it was cleaned up thoroughly?*

- *How will you handle conflicts between the security exercise and other duties?* Ideally, the answer here is "the same way we would handle conflicts between a real incident of this magnitude and other duties". However, institutionalizing that will take work in most organizations. Beginning with lower-grade simulated incidents (for which diverting effort from other projects might not be acceptable in real life) and working your way up may help. After a few successful responses, plan to simulate a more critical incident (preferably while nobody is in the middle of putting out any real-life fires) and see how your teams do. If adequate effort isn't shifted to the exercise, it's important to point this out as a metric of how a real exercise will be treated. It's been my observation that managers who refuse to reallocate effort during an exercise almost always refuse to reallocate effort (or reallocate more than inadequate, token effort) during a real incident.

Red Team

When forming the Red Team, do your best to pull members of your staff who have not been on the Red Team in the past, or at least not recently. Using non-security-team personnel on the Red Team and rotating those personnel regularly can provide an incredible morale boost to your organization because:

- It's fun and different from being the defender.
- It's a good learning experience.
- It keeps people from feeling like being on the Blue Team is a test.
- It builds investment in the success of security exercises.

Not everyone on the Red Team needs to be technical. Plenty of exercises can have a social-engineering aspect to them, and those are carried out just as well by non-technical staff from time to time.

Give the red team plenty of preparation time, but urge them to keep the nature of the planned exercise a secret. Most white-hats who don't do security full time (and many who do!) don't have much experience carrying out mischief, so they'll need time to familiarize themselves with tools and test techniques. This becomes faster and more lightweight the longer your organization does security exercises, because once you are practiced, there likely will be one experienced Red Team member participating in each exercise to help the less-experienced ones.

White Team

Not all exercises need a White Team. However, if any part of the exercise is hypothetical rather than happening live, a White Team is necessary to answer any questions not explained in the hypothetical. The White Team may have to improvise, if it's asked something the exercise designer did not expect!

White Team members often play bit parts in the exercise as well, representing entities outside the project, such as frustrated users or curious reporters.

Blue Team

The Blue Team is everyone not explicitly placed on the White or Red Team and not explicitly excluded from the exercise. The Blue Team is generally responsible for reacting to the simulated security incident as it would to a real one. The main differences will be that, unless you have a partner organization that participates in your security exercises, any outside communication that would happen in a real incident is directed instead at the White Team during exercises.

Follow Through

It is of paramount importance that members of every team record their actions and ideas throughout the exercise. The most important part of any exercise is what is learned from it, and if the knowledge isn't captured, the team as a whole won't learn.

Debriefing

Debriefing an exercise ideally is done within a few hours of the exercise's conclusion. However, with longer, more complex exercises, this may not be possible. I cannot stress enough the importance of good record-keeping to ensure that nothing significant is forgotten before the debrief.

Typically, the incident response leader (Blue Team lead) is responsible for writing a report on the exercise. However, it's been my preference to ask that person to withhold the report until *after* everyone involved in the exercise has had a meeting to debrief the exercise so as not to taint anyone else's recollections.

The debriefing meeting should walk through the exercise from start to finish, giving everyone who participated the chance to voice thoughts, opinions and observations. Anyone interested should have the opportunity to ask questions, and good notes should be taken so that the Blue Team incident response team lead can integrate anything new and interesting into the final report.

The Report

Report-writing may sound boring, but it's an essential part of the process. You've just invested quite a bit of time and effort in a learning exercise. Losing what you've learned would negate that investment. It is important to get the details down so you can refer back to them later when you want to compare a similar incident (real or simulated) or remember how some piece of software or equipment performed. It's also important to have enough information to back up the conclusions and recommendations at the end of your report.

Reports don't have to be fancy or formal if that's not your organization's usual mode of communication. What they should have is a narrative describing the exercise—who was there, what happened, what the timeline was—a summary of what was learned and any suggestions as to how security could be improved through technical controls, policy, training, resource allocation or other methods.

Don't Put It in a Drawer

Finishing the report is not the end of the exercise: your organization either needs to implement the recommendations made in the exercise report, or the person who accepts risk on behalf of your organization needs to document which recommendations will not be implemented and why.

Lather, Rinse, Repeat

These exercises are not an effort to train until you succeed but to train until you can't fail. Although no security program is perfect, if you've trained to the point of near perfection against advanced persistent threat drills, runaway script kiddies become child's play.

In the event of a true failure, the exercise should be rerun with a slight variation within six months. This verifies that new training and controls have remedied the problem, provides needed practice and gives the team an opportunity to overcome the loss, increasing morale.

Tips and Tricks

Here are some pseudo-random thoughts about planning, running and using the information from security exercises:

- Keep track of what is learned in exercises, and keep copies of exercise reports. Ideally, these are great fodder for demonstrating the success of your efforts in improving information security for your project. In the worst case, when recommendations go unheeded, referring decision-makers back to this after a real incident often can bring them around to taking security issues more seriously in the future.

- Have fun! Be willing to see exercises as a game. Encourage creativity and limits-testing. Drop funny Easter eggs into the exercise. This is how you'll get the best bang for your buck in terms of learning and morale.

- Be willing to adapt. The planned exercise doesn't have to be the exercise if something goes wrong. Pivot, and keep everyone on their toes.

- Consider how you'd like your team to respond during real incidents, and be sure that this is the behavior you encourage during exercises.

- Treat every exercise like a success, even when the results are embarrassing. If your incident response usually goes perfectly smoothly, your exercises aren't hard enough. Expect some things to need tweaking after most exercises. It is very important that your team members not see security exercises as an opportunity for them to be graded. If someone performs badly, your response should be that more training is required.

- Start small, and build the difficulty and complexity of exercises over time. Just like weightlifters can't lift 400 pounds on the first day or progress if they don't add weight over time, a team won't get better if it's not challenged. If you are in fact learning, what was challenging last month won't be challenging next year.

- Notes and debrief discussion from Red, White and Blue Team members will identify additional scenario opportunities. Keep track of these ideas as they come up so you have them at the ready when you need to come up with a scenario.

- If you experience resistance to security exercises from The Powers That Be, figure out what influential people you can invite to the Red Team for an exercise. Don't make them token members; make sure they are active and having fun. This tends to turn people around on the practice.

- Don't try too hard for absolute realism in all exercises. Realism is where you begin, but if you are willing to venture into the unreal occasionally, you will learn more. The best Red/Blue exercise I ever participated in was part of an ICS-CERT training out at their facility in Idaho. They built out a surprisingly realistic playground for us to attack and defend, then set us loose with a ridiculous constraint: under no circumstances can you take this infrastructure down to fix its obviously life-threatening problems. No sane person issues that edict in real life. However, not being able to take down the network that the White Team so helpfully built with security akin to Swiss cheese after a mouse convention and shore it up before attackers struck made the Blue Team—of which I was a part—try things we'd never do in real life. I found myself breaking into my own systems to reclaim them from the Red Team, using ARP-spoofing tricks I'd thought died out in the 1990s to reclaim IP addresses on my internal network, and all sorts of other shenanigans. It made me think fast about how the Red Team was operating, and it led me to teach the other Blue Team members details of OSI layer 2 manipulation that many had not been exposed to.

- If the White Team is experienced in exercise design and experienced in running live exercises, do not be afraid to break my non-interference rule. In the aforementioned ICS-CERT training, the White Team kept us on our toes in part by messing with whomever was in the lead and helping whichever team was struggling, in subtle ways. If done badly, White Team interference can ruin an exercise. If done well, it can ensure that everyone is pushed to the limits, even when the Red and Blue Teams have a significant disparity in skill, resources or team cohesion. ■

Example Security Exercises

1) It's Gone

Pick a system, any system. Think of a reason why it's completely hosed—failure of the entire RAID array, fire in the data center, evil script kiddies, sysadmin mistake—and see how your team copes. Some questions to ask when all is done:

- If you don't have another of these systems to fail over to, where are your users going while the system is down? What stopped working and for how long?
- If you have a failover system, how long did it take to fail over? What did your users experience in the meantime?
- How hard was it to replace the system? Were backups adequate? Did the available personnel know what to do and have the authority to do it?
- What data was lost? Are backups being made often enough?
- Were any other systems impacted by this system's death? For example, if your LDAP server died suddenly, did administrators still have access to other systems? Did anything fail open?

2) Naughty Ned

Choose a team member with elevated privileges (any member of your security or systems administration/ops team is usually a good choice, so might be a leadership team member or a developer). Pretend he or she has been fired, and revoke all of his or her privileges. Now he or she gets to cause whatever chaos he or she can with any privileges that remain. This is a great way to test your off-boarding checklist.

3) Wolf in Sheep's Clothing

Most of the Red Team plays the part of ordinary users here. One plays a malicious user. Can the Blue Team terminate the malicious user's activity without negatively impacting any of the nice users?

4) Committer Should Be Committed

This is a great one for software development teams. A developer, working while sleep-deprived (thank you Red Team), has committed something to the master branch of the repo that he or she shouldn't have. It might have been login credentials for an internal system or naked pictures of the boss' dog—the content doesn't matter. The important thing is that it has to go.

See how your team removes the offending data both from the working tree *and* the repository history, without breaking everyone's workflow beyond recognition.

5) Operation!

If you run a DevOps environment, this one's for you. It's far too easy for deployment workflows to end up with very low bus factors (the number of people who must be hit by a bus before the project is doomed or at least in serious trouble). Watch a deployment or two and figure out who the 1–3 most critical people are in that path, then declare them unreachable for the purpose of the exercise.

Now, suppose that a critical security vulnerability has been found in your deployed product. Challenge your team to make a trivial code change (for example, add a comment saying "We did it!" to the code at a specified point), then run your entire test suite and deploy the code with those critical people gone.

6) Finger in the Dam

Find a (hopefully fairly harmless) proof-of-concept for the most recent security vulnerability for which you applied patches. Run it against everything and find out whether the hole really was plugged.

7) Negative Nancy

Have a Red Team member contact your primary customer support avenue, playing the part of a user who is absolutely certain that his or her private information entrusted to your service has been compromised. Bonus points if the character is a "difficult" personality. See how the team handles it.

8) Fell Off a Truck

Your primary authentication database has fallen off a truck (your choice whether this is your database of external user accounts or something for internal personnel only). Demonstrate how you would notify those affected and force password resets. Bonus points if you can detect and flag attempts to use compromised credentials.

9) Ewe Did It

Start an (otherwise innocuous) process on one of your systems that occupies as much RAM as it can get its hands on. See how long it takes for anyone to notice, and how they respond.

10) Stowaway

Connect an unauthorized network device into your network and let it talk to something. See how your team tracks it down and removes it.

11) Exfiltration

One of your employees has decided he or she would like your big, valuable, internal database. The Red Team tries to exfiltrate the target (any way it likes) without being detected.

12) Nosy Nelly

One of your systems starts nmapping the network. Does anyone notice?

13) Blame the Mailman

A system that never should send mail starts sending (or trying to send) spam. What happens next?

14) Delivery

In a disguise, try to make your way into some limited-access area of the building, such as your data center. It helps to appear pregnant, talk on the phone, tailgate someone, carry something heavy or insist you are making a delivery or have an appointment. See if anyone stops you.

15) Pick-Up Stix

Drop some USB sticks around the building—in the parking lot, the restroom, a conference room, a lobby. Place an autorun executable on the sticks that notifies you when they are inserted in a machine that autoruns USB devices, and place an interesting-looking file on there that also tries to call home when opened.

16) Phishing Expedition

Send a convincing phishing e-mail (with at least one flaw that a reasonable person would pick up on) to your staff, directing them to a fake login page and see who gives up their credentials. Note: this one is likely to rankle some people who feel duped when you come out and tell them what happened, but it's really good at driving home the importance of phishing awareness if you can afford the political fallout.

17) Compromising Positions

Suppose that a rootkit has been discovered on a critical piece of infrastructure on your internal network (for example, your satellite server or your LDAP server). Challenge your team to prove that none of your *other* systems have been compromised (not *assume*, *prove*).

18) Failure Is Always an Option

Single points of failure frequently are discussed in meetings. Pay attention and document these, then break one. Does the scope of the outage match expectation? Does the recovery time/process match expectations?

19) Free for All

This is a big, high-investment exercise to run, but it's also the best. Set up a dedicated environment for your exercise to run in that is not connected to your other internal networks or to the public internet. Provide a set of services that needs to be kept running and consider adding some data meant to be kept confidential. Don't set up that environment in the most secure way possible.

Set targets for the Red and Blue Teams with various point values—for example, 10 points to each team for each system it controls at the end of the exercise, 20 points for the Blue Team for every half hour that a particular service continues without interruption, 50 points for the Red Team if it finds and decrypts such-and-such a file. Then set both teams loose with nothing but a time limit and see what happens.

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

RETURN TO CONTENTS