CHAPTER 16
Disaster Planning

By Michael Robinson and Sean Noonan
with Alex Bramley and Kavita Guliani

Because systems will inevitably experience reliability failures or security incidents, you need to be prepared to deal with them. Consequently, we recommend conducting disaster planning activities toward the end of the development cycle, after completing the implementation phase.

This chapter begins with disaster risk analysis—a necessary step for developing a flexible disaster response plan. We then walk through the steps of setting up an incident response team, and provide tips on how to identify prestaging activities that you can perform before a disaster occurs. We conclude with a deep dive about testing your organization before a disaster strikes and a few examples showing how Google prepared for some specific disaster scenarios.

Complex systems can fail in both simple and complex ways, ranging from unexpected service outages to attacks by malicious actors to gain unauthorized access. You can anticipate and prevent some of these failures through reliability engineering and security best practices, but in the long term, failure is almost unavoidable.

Instead of merely hoping that a system will survive a disaster or an attack, or that your staff will be able to mount a reasonable response, disaster planning ensures that you continuously work to improve your ability to recover from a disaster. The good news is, the first steps toward developing a comprehensive strategy are pragmatic and approachable.

Defining "Disaster"

It's rare that you become aware of a disaster only when it's in full swing. Rather than stumbling upon a building fully engulfed in flames, you're more likely to first see or smell smoke—some seemingly small indication that does not necessarily look like a disaster. The fire spreads gradually, and you don't realize the extremity of the situation until you're in the thick of it. Similarly, sometimes a small event—like the accounting error mentioned in Chapter 2—might trigger a full-scale incident response.

Disasters come in various forms:

- *Natural disasters*, including earthquakes, tornadoes, floods, and fires. These tend to be obvious, with varying degrees of impact to a system.
- *Infrastructure disasters*, such as component failures or misconfigurations. These are not always easily diagnosable, and their impact can range from small to large.
- *Service or product outages*, which are observable by customers or other stakeholders.
- *Degradations* of services operating near a threshold. These are sometimes difficult to identify.
- An *external attacker*, who may gain unauthorized access for an extended period of time before being detected.
- *Unauthorized disclosure of sensitive data*.
- *Urgent security vulnerabilities*, requiring you to immediately apply patches to correct new, critical vulnerabilities. These events are treated just like imminent security attacks (see "Compromises Versus Bugs" on page 390).

In this and the following chapter, we use the terms *disaster* and *crisis* interchangeably to mean any situation that may warrant declaring an incident and mounting a response.

Dynamic Disaster Response Strategies

The range of potential disasters is large, but designing flexible disaster response plans will enable you to adapt to rapidly changing situations. By thinking in advance about the possible scenarios you might encounter, you're already taking the first step in preparing for them. As with any skill set, you can hone your disaster response skills by planning, practicing, and iterating on procedures until they become second nature.

Incident response isn't like riding a bicycle. Without routine practice, it is difficult for responders to retain good muscle memory. Lack of practice can lead to fractured

responses and longer recovery times, so it's a good idea to rehearse and refine your disaster response plans often. Well-practiced incident management skills let subject matter experts function naturally during response to incidents—if those skills are second nature, experts won't struggle to follow the process itself.

It's helpful to segment a response plan into phases such as immediate response, short-term recovery, long-term recovery, and resumption of operations. Figure 16-1 shows the general phases associated with disaster recovery.

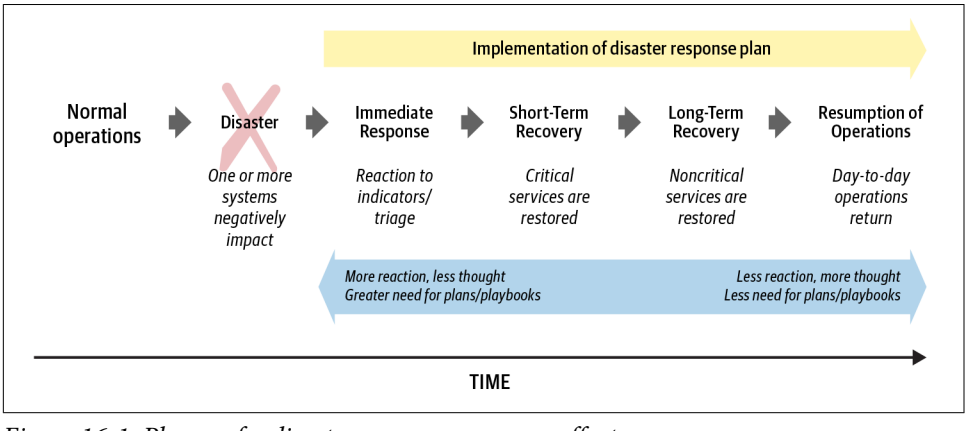

Figure 16-1. Phases of a disaster recovery response effort

The short-term recovery phase should include developing *exit criteria* for the incident—that is, the criteria for declaring that the incident response is complete. A successful recovery might mean restoring a service to a fully operational status, but the underlying solution may have a new design that provides the same level of service. The criteria may also require completely mitigating security threats identified by risk analysis.

When developing a strategy to address immediate responses, short- and long-term recovery, and the resumption of operations, your organization can prepare by doing the following:

- Performing an analysis of potential disasters that will either likely affect the organization or have significant impact
- Establishing a response team
- Creating response plans and detailed playbooks
- Configuring systems appropriately
- Testing your procedures and your systems
- Incorporating feedback from tests and evaluations

Disaster Risk Analysis

Conducting a disaster risk analysis is the first step toward determining an organization's most critical operations—operations whose absence would cause total disruption. Key operational functions include not only important core systems, but also their underlying dependencies, such as networking and application-layer components. Disaster risk analysis should identify the following:

- Systems that, if damaged or offline, could incapacitate operations. You can classify systems as mission-essential, mission-important, or nonessential.
- Resources—either technological or human—that you'll need in order to respond to an incident.
- Disaster scenarios that are likely to occur for each of your systems. The scenarios can be grouped by likelihood of occurrence, frequency of occurrence, and impact to operations (low, medium, high, or critical).

Although you may be able to intuitively conduct an assessment of your operations, a more formalized approach to risk assessment can avoid groupthink and highlight risks that aren't necessarily obvious. For a thorough analysis, we recommend ranking the risks facing your organization by using a standardized matrix that accounts for each risk's probability of occurrence and its impact to the organization. Appendix A provides a sample risk assessment matrix that both large and small organizations can tailor to the specifics of their systems.

Risk ratings provide a good rule of thumb about where to focus attention first. You should review your list of risks for potential outliers after ranking them—for example, an improbable event might rate as critical because of its potential impact. You might also want to solicit an expert to review the assessment to identify risks with hidden factors or dependencies.

Your risk assessment may vary depending on where your organization's assets are located. For example, a site in Japan or Taiwan should account for typhoons, while a site in the Southeastern US should account for hurricanes. Risk ratings may also change as an organization matures and incorporates fault-tolerant systems, like redundant internet circuits and backup power supplies, into its systems. Large organizations should perform risk assessments on both global and per-site levels, and review and update these assessments periodically as the operating environment changes. Equipped with a risk assessment that identifies which systems need protection, you're ready to create a response team prepared with tools, procedures, and training.

Setting Up an Incident Response Team

There are various ways to staff an incident response (IR) team. Organizations typically staff these teams in one of several ways:

- By creating a dedicated full-time IR team
- By assigning IR duties to individuals in addition to their existing job functions
- By outsourcing IR activities to third parties.

Because of budget and size constraints, many organizations rely on existing employees to wear two hats, with some employees performing their regular job duties as well as incident response when the need arises. Organizations with more complex needs may find staffing a dedicated IR team with internal employees worthwhile to ensure that responders are always available to respond, are appropriately trained, have the necessary system access, and have sufficient time to respond to a wide variety of incidents.

Regardless of which staffing model you implement, you can use the following techniques to create successful teams.

Identify Team Members and Roles

When preparing a response plan, you need to identify the core team of people who will respond to incidents and clearly establish their roles. While small organizations may have individual contributors from various teams or even a single team that responds to every incident, organizations with more resources might choose to have a dedicated team for each functional area—for example, a security response team, a privacy team, and an operational team focused on the reliability of public-facing sites.

You can also outsource some functions, while keeping other functions internal. For example, you might not have sufficient funding and workload to fully staff an in-house forensics team, so you might outsource that expertise while keeping your incident response team in house. A potential drawback to outsourcing is that external responders may not be immediately available. It's important to consider response time when determining which functions to keep in-house and which resources to outsource and call upon during an emergency.

You may need some or all of the following roles for an incident response:

Incident commander
: An individual who leads the response to an individual incident.

SREs
: People who can reconfigure an impacted system or implement code to fix a bug.

Public relations
: People who can respond to public inquiries or release statements to the media. These individuals frequently work with the communications lead to craft messages.

Customer support
: People who can respond to customer inquiries or proactively reach out to affected customers.

Legal
: An attorney who can provide counsel on legal matters such as applicable laws, statutes, regulations, or contracts.

Privacy engineers
: People who can address impact on technical privacy matters.

Forensic specialists
: People who can perform event reconstruction and attribution to determine what happened and how it occurred.

Security engineers
: People who can review the security impact of the incident and work with SREs or privacy engineers to secure a system.

When determining which roles will be staffed by in-house personnel, you may need to implement a rotational staffing model, where the IR team operates in shifts. It's critical to staff shifts during an incident in order to reduce fatigue and provide ongoing support during the incident. You can also adopt this model to provide flexibility as main job responsibilities evolve and flow over time. Keep in mind that these are roles, not individuals. An individual may hold multiple roles during an incident.

Avoid Single Points of Failure

Incidents do not respect meeting invitations, the standard workday, travel plans, or vacation schedules. While on-call rotations can increase your ability to respond to incidents, pay special attention to avoiding single points of failure. For example, executives should empower delegates who can approve emergency code fixes, configuration changes, and communication messages during incidents when you can't wait for someone's vacation travel to end. If your organization is multinational, appoint delegates across time zones.

After you determine which roles should be staffed internally, you can create an initial list of personnel to serve on the individual IR teams. Identifying these people in advance helps to clarify roles, responsibilities, and ownership during a response effort, and minimizes chaos and setup time. It's also helpful to identify a *champion*

for your IR team—a person with enough seniority to commit resources and remove roadblocks. The champion can help assemble a team and work with senior leaders when there are competing priorities. Check out Chapter 21 for more information.

Establish a Team Charter

An IR team's charter should start with the team's *mission*—a single sentence describing the types of incidents they'll handle. The mission allows the reader to quickly understand what the team does.

The *scope* of the charter should describe the environment you work in, focusing on technologies, end users, products, and stakeholders. This section should clearly define the types of incidents the team will handle, which incidents should be handled by internally staffed resources, and which incidents should be assigned to outsourced teams.

Team Morale

When establishing a team charter—whether for a dedicated response team or a cross-functional virtual response team—be sure to consider if the scope and workload are appropriate. When people are overworked, their productivity may decrease; over time, they may even leave your organization. For a discussion of morale during incident response, see "Morale" on page 405.

To ensure that the IR team focuses on qualifying incidents, it is important that the organization's leadership and the IR champion agree on the scope. For example, while an IR team could certainly respond to individual customer inquiries about system firewall configurations and log enablement/verification, those tasks may be better suited for a customer support team.

Finally, it's important to define what *success* looks like for the team. In other words, how do you know when the IR team's job is done or can be declared done?

Establish Severity and Priority Models

Severity and priority models can assist an IR team in quantifying and understanding the gravity of incidents and the operational tempo needed to respond to them. You should use both models concurrently, as they're related.

A *severity model* allows the team to categorize incidents based on the severity of their impact on the organization. You might rank incidents on a five-point (0–4) scale, with 0 indicating the most severe incidents and 4 indicating the least severe incidents. You should adopt whatever scale best fits your organizational culture (colors, animals, etc.). For example, if you're using the five-point scale, unauthorized individuals

on a network may qualify as a severity 0 incident, while temporary unavailability of security logs may be severity 2. When building the model, review any previously performed risk analyses so you can assign categories of incidents appropriate severity ratings. Doing so will ensure that not all incidents receive a critical or moderate severity rating. Accurate ratings will help incident commanders prioritize when multiple incidents are reported simultaneously.

A *priority model* defines how quickly personnel need to respond to incidents. This model builds upon your understanding of incident severity and can also use a five-point (0–4) scale, with 0 indicating high priority and 4 indicating low priority. Priority drives the tempo of the work required: an incident rated 0 merits immediate incident response, where team members respond to this incident before any other work. You can handle an incident with a rating of 4 with routine operational work. An agreed upon-priority model also helps keep various teams and operational leads in sync. Imagine one team treating an incident as priority 0, while a second team with a limited awareness of the total circumstances considers it a priority 2. The two teams will likely operate at a different tempo, delaying proper incident response.

Typically, once it's fully understood, the severity of an incident will remain fixed throughout the incident's lifecycle. Priority, on the other hand, may change throughout an incident. During the early phases of triaging and implementing a critical fix, the priority may be 0. After a critical fix is in place, you may lower the priority to 1 or 2 as the engineering teams perform cleanup work.

Define Operating Parameters for Engaging the IR Team

After you've established severity and priority models, you can define operating parameters to describe the day-to-day functioning of the incident response team. This becomes increasingly important when teams perform regular operational work in addition to incident response work, or when you need to communicate with virtual teams or outsourced teams. Operating parameters ensure that severity 0 and priority 0 incidents receive a timely response.

Operating parameters might include the following:

- The expected time it takes to initially respond to reported incidents—for example, within 5 minutes, 30 minutes, an hour, or the next business day
- The expected time it takes to perform an initial triage assessment and develop a response plan and operating schedule
- Service level objectives (SLOs), so team members understand when incident response should interrupt day-to-day work.

There are many ways to organize on-call rotations to ensure incident response work is properly load-balanced across a team or balanced according to regularly scheduled

ongoing work. For a detailed discussion, refer to Chapters 11 and 14 of the SRE book, Chapters 8 and 9 of the SRE workbook, and Chapter 14 of Limoncelli, Chalup, and Hogan (2014).[1]

Develop Response Plans

Decision making during severe incidents may be challenging because responders are attempting to work quickly with limited information. Well-crafted response plans can guide responders, reduce wasted steps, and provide an overarching approach for how to respond to different categories of incidents. While an organization may have a company-wide incident response policy, the IR team needs to create a set of response plans covering the following subjects:

Incident reporting
: How an incident is reported to the IR team.

Triage
: A list of the IR team members who will respond to the initial report and start triaging the incident.

Service level objectives
: A reference to the SLOs on how quickly the responders will act.

Roles and responsibilities
: Clear definitions of the roles and responsibilities of the IR team participants.

Outreach
: How to reach out to engineering teams and participants who may need to assist in incident response.

Communications
: Effective communication during an incident does not happen without advance planning. You need to establish how to do each of the following:

  - Inform leadership of an incident (for example, via email, text message, or phone call), and the information to include in these communications.
  - Conduct intraorganization communication during an incident, including within and between response teams (standing up chat rooms, videoconferencing, email, secure IRC channels, bug tracking tools, and so on).

1 Limoncelli, Thomas A., Strata R. Chalup, and Christina J. Hogan. 2014. *The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems*. Boston, MA: Addison-Wesley.

- Communicate with external stakeholders, such as regulators or law enforcement, when the situation requires it. You need to partner with the legal function and other departments of your organization to plan and support this communication. Consider keeping an index of contact details and communication methods for each external stakeholder. If your IR team grows large enough, you might automate those notification mechanisms as appropriate.
- Communicate incidents to support teams who interact with customers.
- Communicate with responders and leadership without tipping off an adversary if a communication system is unavailable, or if you suspect that a communication system is compromised.

Security and Reliability Risk: Communicating When Your Email or Instant Messaging System Is Compromised

Many incident response teams rely heavily on email or an instant messaging system when coordinating a response to an incident. Team members who are distributed across remote sites can look at communication threads to determine the current status of a response effort. Unfortunately, reliance on a single communication system can work against an IR team's efforts. For example:

- An adversary who has compromised an email or instant messaging server or a domain controller may choose to join the distribution group used to coordinate the response. The attacker can then follow the response effort and sidestep future detection efforts. Additionally, they may be able to learn the status of mitigation efforts and then pivot to systems that the IR team has declared clean.
- If the communication system is offline, the IR team may not be able to contact stakeholders or teams in other locations. This can add an appreciable delay to response effort coordination and the time it takes to bring your organization back online.

To avoid such an event from disrupting your ability to stand up an IR team, ensure that the communications section of your IR plan covers backup communication methods. We discuss the topic of operational security (OpSec) in more detail in "Operational Security" on page 394.

Every response plan should outline high-level procedures so a trained responder can act. The plan should contain references to sufficiently detailed playbooks that trained responders can use to carry out specific actions, and it might also be a good idea to outline the overarching approach for responding to particular classes of incidents. For example, when dealing with network connectivity issues, a response plan should contain a broadly defined summary of the areas to analyze and the troubleshooting

steps to perform, and it should reference a playbook that contains specific instructions on how to access the appropriate network devices—e.g., logging into a router or firewall. Response plans might also outline the criteria for an incident responder to determine when to notify senior leadership of incidents and when to work with localized engineering teams.

Create Detailed Playbooks

Playbooks complement response plans and list specific instructions for how a responder should perform particular tasks from beginning to end. For example, playbooks might describe how to grant responders emergency temporary administrative access to certain systems, how to output and parse particular logs for analysis, or how to fail over a system and when to implement graceful degradation.[2] Playbooks are procedural in nature and should be frequently revised and updated. They are typically team-specific, which implies that the response to any disaster may involve multiple teams working through their own specific procedural playbooks.

Ensure Access and Update Mechanisms Are in Place

Your team should define a place to store documentation so that materials are available during a disaster. A disaster may impact access to documentation—for example, if company servers go offline—so make sure you have copies in a location that is accessible during an emergency, and that these copies stay up to date.

Systems get patched, updated, and reconfigured, and threat postures can change. You may discover new vulnerabilities, and new exploits will appear in the wild. Periodically review and update your response plans to make sure they're accurate and reflect any recent configuration or operational changes.

Good incident management requires frequent and robust information management. A team should identify a suitable system for tracking information about an incident and retaining incident data. Teams that handle security and privacy incidents may want a system that's tightly controlled by access on a need-to-know basis, while response teams that deal with service or product outages may want to create a system that is broadly accessible throughout the company.

Prestaging Systems and People Before an Incident

After you've performed risk analysis, created an IR team, and documented the appropriate procedures, you need to identify prestaging activities that you can perform before a disaster occurs. Make sure you consider prestaging that's relevant to every

2 These topics are described in Chapter 22 of the SRE book.

phase of the incident response lifecycle. Typically, prestaging involves configuring systems with defined logging retention, automatic responses, and clearly defined human procedures. By understanding each of these elements, a response team can eliminate gaps in coverage between sources of data, automated responses, and human responses. Response plans and playbooks (discussed in the previous section) describe much of what is required for human interactions, but responders also need access to appropriate tools and infrastructure.

To facilitate rapid response to incidents, an IR team should predetermine appropriate levels of access for incident response and establish escalation procedures ahead of time so the process to obtain emergency access isn't slow and convoluted. The IR team should have read access to logs for analysis and event reconstruction, as well as access to tools for analyzing data, sending reports, and conducting forensic examinations.

Configuring Systems

You can make a number of adjustments to systems before a disaster or incident to reduce an IR team's initial response time. For example:

- Build fault tolerance into local systems and create failovers. For more information on this topic, see Chapters 8 and 9.
- Deploy forensic agents, such as GRR agents or EnCase Remote Agents, across the network with logs enabled. This will aid both your response and later forensic analysis. Be aware that security logs may require a lengthy retention period, as discussed in Chapter 15 (the industry average for detecting intrusions is approximately 200 days, and logs deleted before an incident is detected cannot be used to investigate it). However, some countries, such as those in the European Union, have particular requirements about how long you can retain logs. When setting up a retention plan, consult your organization's attorney.
- If your organization commits backups to tape or other media, retain an identical set of the hardware and software used to create them so you can restore the backups in a timely fashion if the primary backup system is unavailable. You should also perform periodic restore drills to make sure your equipment, software, and procedures work correctly. IR teams should identify the procedures they'll use to work with individual domain teams (e.g., email teams or network backup teams) to test, verify, and perform data restoration during incidents.
- Have multiple fallback paths for access and recovery in emergencies. An outage that impacts your production network may be hard to recover from unless you have secure alternative pathways to access the network control plane. Similarly, if you discover a breach and aren't sure how widespread the compromise of your

corporate workstations is, recovery will be much easier if you have a known safe set of air-gapped systems that you can still trust.

Training

IR team members should be trained on severity/priority models, the IR team's operating model, response times, and the locations of response plans and playbooks. You can read more about Google's approach to incident response in Chapter 9 of the SRE workbook.

Training requirements for incident response extend beyond the IR team, however. During emergencies, some engineers may respond without thinking about or realizing the consequences of their actions. To mitigate this risk, we recommend training the engineers who will assist the IR team on the various IR roles and their responsibilities. We use a system called Incident Management at Google (IMAG), which is based on the Incident Command System. The IMAG framework assigns critical roles like incident commander, operational leads, and communications lead.

Train your employees to recognize, report, and escalate an incident. Incidents may be detected by engineers, customers/users, automated alerts, or an administrator. Separate, clear channels should exist for each party to report an incident, and company employees should be trained on how and when to escalate an incident to the IR team.[3] This training should support the organization's IR policy.

There should be a finite limit on the amount of time an engineer can grapple with an incident before escalation. The amount of time available for a first responder depends upon the level of risk the organization is prepared to accept. You might start with a 15-minute window, and adjust that window as necessary.

You should establish the criteria for decision making before the heat of the moment to ensure that responders choose the most logical course of action, rather than making a gut decision on the fly. First responders are frequently faced with the need to make immediate decisions about whether to take a compromised system offline or what containment method to use. For more discussion on this topic, see Chapter 17.

You should train engineers to understand that incident response can require addressing competing priorities that may appear to be mutually exclusive—for example, the need to maintain maximum uptime and availability while also preserving artifacts for forensic investigation. You should also train engineers to create notes about their response activities, so they can later differentiate these activities from artifacts left by an attacker.

3 See Chapter 14 of the SRE book and the production-specific worked examples in Chapter 9 of the SRE workbook.

Processes and Procedures

By establishing a set of processes and procedures to follow before an incident occurs, you drastically reduce the response time and cognitive load for responders. For example, we recommend the following:

- Defining rapid procurement methods for hardware and software. You may need additional equipment or resources, such as servers, software, or fuel for generators, during emergencies.
- Establishing contract approval processes for outsourcing services. For smaller organizations, this may mean identifying outsourced capabilities such as forensic investigation services.
- Creating policies and procedures to preserve evidence and logs during security incidents and prevent log overwrites. For more details, see Chapter 15.

Testing Systems and Response Plans

Once you've created all the materials your organization needs to be prepared for an incident, as described in the preceding sections, it's essential to evaluate the effectiveness of those materials and improve upon any deficiencies you identify. We recommend testing from several perspectives:

- Evaluate automated systems to make sure they're operating correctly.
- Test processes to eliminate any gaps in the procedures and tools used by first responders and engineering teams.
- Train personnel who will respond during incidents to make sure they have the necessary skills to respond to a crisis.

You should run these tests on a periodic basis—annually, at a minimum—to ensure that your systems, procedures, and responses are dependable and applicable in case of an actual emergency.

Each component plays a vital role in returning a disaster-stricken system to an operational state. Even if your IR team is highly skilled, without procedures or automated systems, its ability to respond to a disaster will be inconsistent. If your technical procedures are documented but aren't accessible or usable, they'll likely never be implemented. Testing the resilience of each layer of the disaster response plan decreases those risks.

For many systems, you'll need to document the technical procedures for mitigating threats, audit the controls regularly (for example, quarterly or annually) to ensure they're still being implemented, and provide a list of fixes to engineers to correct any

weaknesses you identify. Organizations just beginning their IR planning may want to investigate certifications around disaster recovery and business continuity planning for inspiration.

Auditing Automated Systems

You should audit all critical systems *and* dependent systems—including backup systems, logging systems, software updaters, alert generators, and communication systems—to make sure they're operating correctly. A full audit should ensure the following:

The backup system is operating correctly.
: Backups should be created correctly, stored in a safe location, stored for the appropriate amount of time, and stored with the correct permissions. Conduct data recovery and validation exercises periodically to ensure that you can retrieve and use the data from backups. For more information about Google's data integrity approach, see Chapter 26 of the SRE book.

Event logs (discussed in the previous chapter) are stored correctly.
: These logs allow responders to construct an accurate timeline when reconstructing events during forensic investigations. You should store event logs for a time period appropriate to the organization's level of risk and other applicable considerations.

Critical vulnerabilities are patched in a timely fashion.
: Audit both automatic and manual patch processes to reduce the need for human intervention and the likelihood of human errors.

Alerts are generated correctly.
: Systems generate alerts—email alerts, dashboard updates, text messages, etc.—when particular criteria are met. Validate each alert rule to make sure it fires correctly. Also, make sure to account for dependencies. For example, how are your alerts impacted if an SMTP server goes offline during a network outage?

Communication tools, such as chat clients, email, conference-call bridging services, and secure IRC, work as intended.
: Functioning communication channels are essential for response teams. You should also audit the failover capability of these tools and ensure that they retain the messages you'll need to write the postmortem.

Conducting Nonintrusive Tabletops

Tabletop exercises are incredibly valuable tools for testing documented procedures and evaluating the performance of response teams. These exercises can be starting points for evaluating end-to-end incident responses, and can also be useful when practical testing—for example, causing an actual earthquake—isn't feasible. The simulations can range from small to large in scope, and are typically nonintrusive: because they don't take a system offline, they don't disrupt production environments.

Similar to the Wheel of Misfortune exercises described in Chapter 15 of the SRE book, you can run a tabletop exercise by presenting participants with an incident scenario with various follow-up storyline variations. Ask participants to describe how they would respond to the scenario, and what procedures and protocols they would follow. This approach lets participants flex their decision-making skills and receive constructive feedback. The open structure of these exercises means that they can incorporate a wide range of participants, including these:

- Frontline engineers, following detailed playbooks to restore a crippled system to service
- Senior leadership, making business-level decisions with respect to operations
- Public relations professionals, coordinating external communications
- Lawyers, providing contextual legal guidance and helping craft public communications

The most important aspect of these tabletops is to challenge responders and provide an opportunity for everyone involved to practice the relevant procedures and decision-making processes before a real incident occurs.

Here are some of the key features to consider when implementing tabletop exercises:

Believability
: Tabletop scenarios should be believable—an engaging scenario motivates participants to follow along without suspending disbelief. For example, an exercise might posit that a user falls for a phishing attack, allowing an adversary to exploit a vulnerability on the user's workstation. You can base pivot points—the steps by which an attacker moves throughout a network—on realistic attacks and known vulnerabilities and weaknesses.

Details
: The person who crafts the tabletop scenario should research that scenario in advance, and the facilitator should be well versed in the details of the event and typical responses to the scenario. To aid believability, the creator of the tabletop can create artifacts that participants would encounter during a real incident, such as log files, reports from customers or users, and alerts.

Decision points
: Much like a "choose your own adventure" story, a tabletop exercise should have decision points that help the plot unfold. A typical 60-minute tabletop exercise contains approximately 10–20 storyline decision points to engage the participants in decision making that affects the outcome of the exercise. For example, if tabletop participants decide to take a compromised email server offline, then the participants can't send email notifications for the rest of the scenario.

Participants and facilitators
: Make tabletop exercises as interactive as possible. As the exercise unfolds, the facilitator may need to respond to the actions and commands executed by the responders. Rather than simply discussing how they would respond to an incident, participants should demonstrate how they would respond. For example, if an IR playbook calls for an incident responder to escalate a ransomware attack to a member of the forensics team for investigation and also to a member of the network security team to block traffic to a hostile website, the responder should carry out these procedures during the tabletop exercise. "Performing the response" helps the incident responder build muscle memory. The facilitator should familiarize themselves with the scenario in advance so they can improvise and nudge the responders in the right direction when needed. Again, the goal here is to enable participants to actively engage in the scenario.

Outcomes
: Rather than leaving participants feeling defeated, a successful tabletop exercise should conclude with actionable feedback on what worked well and what didn't work so well. The participants and facilitator should be able to make concrete recommendations for areas of improvement for the incident response team. Where appropriate, participants should recommend changes to systems and policies to fix inherent weaknesses they find. To make sure that participants address these recommendations, create action items with specific owners.

Testing Response in Production Environments

While tabletop exercises are useful for simulating a range of incident scenarios, you need to test some incident scenarios, attack vectors, and vulnerabilities in a real-world production environment. These tests operate at the intersection of security and reliability by allowing IR teams to understand operational constraints, practice with real-world parameters, and observe how their responses affect production environments and uptime.

Single system testing/fault injection

Rather than testing an entire system end to end, you can break large systems into individual software and/or hardware components for testing. Tests can come in a

variety of forms and can involve a single local component or a single component with organization-wide reach. For example, what happens when a malicious insider connects a USB storage device to a workstation and attempts to download sensitive content? Do the local logs track the local USB port activity? Are the logs sufficiently aggregated and escalated in a timely fashion, enabling a security team to respond quickly?

We particularly recommend that you conduct single-system testing by using fault injection. Building fault injection into your systems allows you to run targeted tests without disrupting the entire system. More importantly, fault injection frameworks allow individual teams to test their systems without involving their dependencies. As an example, consider the open source Envoy HTTP proxy, which is often used for load balancing. In addition to its many load-balancing features, the proxy supports a fault injection HTTP filter, which you can use to return arbitrary errors for a percentage of traffic or to delay requests for a specific amount of time. Using this type of fault injection, you can test that your system handles time-outs correctly, and that time-outs don't lead to unpredictable behavior in production.

When you do find unusual behavior in production, a well-exercised fault injection framework can enable more structured investigation, where you can reproduce production issues in a controlled way. For example, imagine the following scenario: when a company's users attempt to access specific resources, the infrastructure checks all authentication requests using a single source. The company then migrates to a service that requires multiple authentication checks against various sources to obtain similar information. As a result, clients began to exceed the configured time-out for these function calls. The error handling within the caching component of the authentication library erroneously treats these time-outs as permanent (rather than temporary) failures, triggering many other small failures throughout the infrastructure. By using an established incident response framework of fault injection to inject latency in some of the calls, the response team can easily reproduce the behavior, confirm their suspicions, and develop a fix.

Human resource testing

While many tests address technical aspects of a system, tests should also consider personnel failures. What happens when specific personnel are unavailable or fail to act? Often, IR teams rely on individuals with strong institutional knowledge of the organization rather than following set processes. If key decision makers or managers are unavailable during a response, how will the rest of the IR team proceed?

Multicomponent testing

Working with distributed systems means that any number of dependent systems or system components might fail. You need to plan for multicomponent failures and create relevant incident response procedures. Consider a system that depends on

multiple components, each of which you tested individually. If two or more components fail simultaneously, what parts of incident response must you handle differently?

A thought exercise might not be sufficient to test every dependency. When considering a service disruption in a security context, you need to consider security concerns in addition to failure scenarios. For example, when failovers occur, does the system respect existing ACLs? What safeguards ensure this behavior? If you're testing an authorization service, does the dependent service fail closed? For a deeper dive into this topic, see Chapter 5.

System-wide failures/failovers

Beyond testing single components and dependencies, consider what happens when your entire system fails. For example, many organizations run primary and secondary (or disaster recovery) datacenters. Until you fail over to operate from your secondary location, you can't be confident that your failover strategy will protect your business and security posture. Google regularly cycles power to entire datacenter buildings to test if the failure causes a user-visible impact. This exercise ensures both that services retain the ability to operate without a specific datacenter location, and that the technicians performing this work are well practiced in managing power-off/power-on procedures.

For services running on another provider's cloud infrastructure, consider what happens to your service if an entire availability zone or region fails.

Red Team Testing

In addition to announced testing, Google practices disaster preparedness exercises known as *Red Team* exercises: offensive testing performed by its Information Security Assurance organization. Similar to DiRT exercises (see "DiRT Exercise Testing Emergency Access" on page 384), these exercises simulate real attacks in order to test and improve detection and response capabilities, and to demonstrate the business impact of security issues.

A Red Team typically provides no advance notice to incident responders, with the exception of senior leadership. Because Red Teams are familiar with Google's infrastructure, their tests are much more productive than standard network penetration tests. Since these exercises occur internally, they provide an opportunity to balance between fully external attacks (where the attacker is outside of Google) and internal attacks (insider risk). Additionally, Red Team exercises supplement security reviews by testing security end to end, and by testing human behavior through attacks like phishing and social engineering. For a deeper exploration of Red Teams, see "Special Teams: Blue and Red Teams" on page 465.

Evaluating Responses

When responding to both live incidents and test scenarios, it's important to create an effective feedback loop so you don't fall victim to the same situations repeatedly. Live incidents should require postmortems with specific action items; you can similarly create postmortems and corresponding action items for testing. While testing can be both a fun exercise and an excellent learning experience, the practice does require some rigor—it's important to track a test's execution, and to critically evaluate its impact and how people throughout your organization respond to the test. Performing an exercise without implementing the lessons you learn from it is merely entertainment.

When evaluating your organization's responses to incidents and tests, consider the following best practices:

- Measure the responses. Evaluators should be able to identify what worked well and what did not. Measure the amount of time it took to implement each stage of the response so you can identify corrective measures.
- Write blameless postmortems and focus on how you can improve the systems, procedures, and processes.[4]
- Create feedback loops for improving existing plans or developing new plans as needed.
- Collect artifacts and feed them back into signal detection. Make sure you address any gaps you identify.
- So that you can perform forensic analysis and address gaps, make sure you save the appropriate logs and other relevant material—especially when conducting security exercises.
- Evaluate even "failed" tests. What worked, and what do you need to improve?[5]
- As discussed in "Special Teams: Blue and Red Teams" on page 465, implement color teams to ensure that your organization acts on the lessons you learn. You may need a hybrid Purple Team to make sure that the Blue Team addresses the vulnerabilities exploited by a Red Team in a timely fashion, thereby preventing attackers from leveraging the same vulnerabilities repeatedly. You can think of Purple Teams like regression testing for vulnerabilities.

4 See Chapter 15 of the SRE book.

5 See Chapter 13 of the SRE book.

Google Examples

To make the concepts and best practices described in this chapter more concrete, here are a few real-world examples.

Test with Global Impact

In 2019, Google conducted a test of the response to a major earthquake in the San Francisco Bay Area. The scenario included components to simulate the impact on the physical plant and its facilities, transportation infrastructure, networking components, utilities and power, communication, business operations, and executive decision making. Our goal was to test Google's response to a large disruption and the impact on global operations. Specifically, we tested the following:

- How would Google provide immediate first aid to individuals injured during the earthquake and multiple aftershocks?
- How would Google provide help to the public?
- How would employees escalate information to Google's leadership? In the event of communication disruption—for example, a downed cellular network or disrupted LAN/MAN/WAN—how would Google disseminate information to employees?
- Who would be available for on-site response if employees had conflicts of interest—for example, if employees needed to take care of their family members and homes?
- How would unpassable secondary roadways impact the region? How would the overflow impact primary roadways?
- How would Google provide assistance to employees, contractors, and visitors who were stranded on Google's campuses?
- How could Google evaluate the damage to its buildings, which might include broken pipes, sewage issues, broken glass, loss of power, and broken network connections?
- If locally affected teams could not initiate a transfer of authority/responsibility, how would SREs and various engineering teams outside the geographic area take control over systems?
- How could we enable leadership outside the affected geographic area to continue with business operations and making business-related decisions?

DiRT Exercise Testing Emergency Access

Sometimes we can test the robustness of both reliability and security operations simultaneously. During one of our annual Disaster Recovery Training (DiRT) exercises,[6] SREs tested the procedure and functionality of breakglass credentials:[7] could they gain emergency access to the corporate and production networks when standard ACL services were down? To add a security testing layer, the DiRT team also looped in the signals detection team. When SREs engaged the breakglass procedure, the detection team was able to confirm that the correct alert fired and that the access request was legitimate.

Industry-Wide Vulnerabilities

In 2018, Google received early notice of two vulnerabilities in the Linux kernel, which underpins much of our production infrastructure. By sending specially crafted IP fragments and TCP segments, either SegmentSmack (CVE-2018-5390) or FragmentSmack (CVE-2018-5391) could cause a server to perform expensive operations. By using large amounts of both CPU and wall-clock time, this vulnerability could allow an attacker to obtain a significant scale boost beyond a normal denial-of-service attack—a service that could normally cope with a 1 Mpps attack would fall over at approximately 50 Kpps, a 20× reduction in resilience.

Disaster planning and preparation enabled us to mitigate this risk in two dimensions: technical aspects and incident management aspects. On the incident management front, the looming disaster was so significant that Google assigned a team of incident managers to work on the problem full time. The team needed to identify the affected systems, including vendor firmware images, and enact a comprehensive plan to mitigate the risk.

On the technical front, SREs had already implemented defense-in-depth measures for the Linux kernel. A runtime patch, or *ksplice*, that uses function redirection tables to make rebooting a new kernel unnecessary can address many security issues. Google also maintains kernel rollout discipline: we regularly push new kernels to the entire fleet of machines with a target of less than 30 days, and we have well-defined mechanisms to increase the rollout speed of this standard operating procedure if necessary.[8]

6 See Kripa Krishnan's article "Weathering the Unexpected" and her USENIX LISA15 presentation "10 Years of Crashing Google".

7 A breakglass mechanism can bypass policies to allow engineers to quickly resolve outages. See "Breakglass" on page 67.

8 Chapter 9 discusses additional design approaches that prepare your organization to respond to incidents quickly.

If we'd been unable to fix the vulnerability using a ksplice, we could have performed an emergency rollout at speed. However, in this case, it was possible to address the two affected functions in the kernel—`tcp_collapse_ofo_queue` and `tcp_prune_ofo_queue`—with a kernel splice. SREs were able to apply the ksplice to production systems without adversely affecting the production environment. Because the rollout procedure was already tested and approved, SREs quickly obtained VP approval to apply the patch during a code freeze.

Conclusion

When contemplating how to spin up disaster recovery tests and plans from scratch, the sheer volume of possible approaches can seem overwhelming. However, you can apply the concepts and best practices in this chapter even at a small scale.

As a starting point, identify your most important system or a piece of critical data, and then identify how you would respond to various disasters that affect it. You need to determine how long you can operate without a service and the number of people or other systems that it impacts.

From this first important step, you can expand your coverage piece by piece into a robust disaster preparedness strategy. From an initial strategy of identifying and preventing the sparks that start a fire, you can work your way up to responding to that inevitable eventual blaze.