Recovering from a disaster begins way before the event occurs. It starts by anticipating threats and developing goals that support the organization's continuity of operations. If you do not have established goals, how do you know when you are done and whether your efforts were actually successful? Goals are established so everyone knows the ultimate objectives. Establishing goals is important for any task, but especially for business continuity and disaster recovery plans. The definition of the goals helps direct the proper allocation of resources and tasks, supports the development of necessary strategies, and assists in financial justification of the plans and program overall. Once the goals are set, they provide a guide to the development of the actual plans themselves. Anyone who has been involved in large projects that entail many small, complex details knows that at times it is easy to get off track and not actually accomplish the major goals of the project. Goals are established to keep everyone on track and to ensure that the efforts pay off in the end.

Great—we have established that goals are important. But the goal could be, "Keep the company in business if an earthquake hits." That's a good goal, but it is not overly useful without more clarity and direction. To be useful, a goal must contain certain key information, such as the following:

• Authority In times of crisis, it is important to know who is in charge. Teamwork is important in these situations, and almost every team does much better with an established and trusted leader. Such leaders must know that they are expected to step up to the plate in a time of crisis and understand what type of direction they should provide to the rest of the employees. Everyone else must recognize the authority of these leaders and respond accordingly. Clear-cut authority will aid in reducing confusion and increasing cooperation.

• Priorities It is extremely important to know what is critical versus what is merely nice to have. Different departments provide different functionality for an organization. The critical departments must be singled out from the departments that provide functionality that the organization can live without for a week or two. It is necessary to know which department must come online first, which second, and so on. That way, the efforts are made in the most useful, effective, and focused manner. Along with the priorities of departments, the priorities of systems, information, and programs must be established. It may be necessary to ensure that the database is up and running before working to bring the web servers online. The general priorities must be set by management with the help of the different departments and IT staff.

• Implementation and testing It is great to write down very profound ideas and

develop plans, but unless they are actually carried out and tested, they may not add up to a hill of beans. Once a disaster recovery plan is developed, it actually has to be put into action. It needs to be documented and stored in places that are easily accessible in times of crisis. The people who are assigned specific tasks need to be taught and informed how to fulfill those tasks, and dry runs must be done to walk people through different situations. The exercises should take place at least once a year, and the entire program should be continually updated and improved.

NOTE We address various types of tests, such as walkthrough, tabletop, simulation, parallel, and full interruption, later in this chapter.

According to the U.S. Federal Emergency Management Agency (FEMA), 90 percent of small businesses that experience a disaster and are unable to restore operations within five days will fail within the following year. Not being able to bounce back quickly or effectively by setting up shop somewhere else can make a company lose business and, more importantly, its reputation. In such a competitive world, customers have a lot of options. If one company is not prepared to bounce back after a disruption or disaster, customers may go to another vendor and stay there.

The biggest effect of an incident, especially one that is poorly managed or that was preventable, is on an organization's reputation or brand. This can result in a considerable and even irreparable loss of trust by customers and clients. On the other hand, handling an incident well, or preventing great damage through smart, preemptive measures, can enhance the reputation of, or trust in, an organization.

The disaster recovery plan (DRP) should address in detail all of the topics we have covered so far. The actual format of the DRP will depend on the environment, the goals of the plan, priorities, and identified threats. After each of those items is examined and documented, the topics of the plan can be divided into the necessary categories.

Response

The first question the DRP should answer is, "What constitutes a disaster that would trigger this plan?" Every leader within an organization (and, ideally, everyone else too) should know the answer. Otherwise, precious time is lost notifying people who should've

self-activated as soon as the incident occurred, a delay that could cost lives or assets.
Examples of clear-cut disasters that would trigger a response are loss of power exceeding
ten minutes, flooding in the facility, or terrorist attack against or near the site.
Every DRP is different, but most follow a familiar sequence of events:
1. Declaration of disaster
2. Activation of the DR team
3. Internal communications (ongoing from here on out)
4. Protection of human safety (e.g., evacuation)
5. Damage assessment
6. Execution of appropriate system-specific DRPs (each system and network should have its own DRP)
7. Recovery of mission-critical business processes/functions
8. Recovery of all other business processes/functions

Personnel
The DRP needs to define several different teams that should be properly trained and
available if a disaster hits. Which types of teams an organization needs depends upon the
organization. The following are some examples of teams that an organization may need
to construct:

The DR coordinator should have an understanding of the needs of the organization and the types of teams that need to be developed and trained. Employees should be
assigned to the specific teams based on their knowledge and skill set. Each team needs

- Damage assessment team
- Recovery team
- Relocation team
- Restoration team
- Salvage team
- Security team

to have a designated leader, who will direct the members and their activities. These team
leaders will be responsible not only for ensuring that their team’s objectives are met but
also for communicating with each other to make sure each team is working in parallel
phases.
The purpose of the recovery team should be to get whatever systems are still operable
back up and running as quickly as possible to reduce business disruptions. Think

of
them as the medics whose job is to stabilize casualties until they can be transported to
the hospital. In this case, of course, there is no hospital for information systems, but there
may be a recovery site. Getting equipment and people there in an orderly fashion should
be the job of the relocation team. The restoration team should be responsible for getting
the alternate site into a working and functioning environment, and the salvage team
should be responsible for starting the recovery of the original site. Both teams must know
how to do many tasks, such as install operating systems, configure workstations and
servers, string wire and cabling, set up the network and configure networking services,
and install equipment and applications. Both teams must also know how to restore
data from backup facilities and how to do so in a secure manner, one that ensures the
availability, integrity, and confidentiality of the system and data.
The DRP must outline the specific teams, their responsibilities, and notification
procedures. The plan must indicate the methods that should be used to contact team
leaders during business hours and after business hours.

Communications
The purpose of the emergency communications plan that is part of the overall DRP is to
ensure that everyone knows what to do at all times and that the DR team remains
synchronized and coordinated. This all starts with the DR plan itself. As stated previously,
copies of the DRP need to be kept in one or more locations other than the primary site,
so that if the primary site is destroyed or negatively affected, the plan is still available to
the teams. It is also critical that different formats of the plan be available to the teams,
including both electronic and paper versions. An electronic version of the plan is not very
useful if you don’t have any electricity to run a computer.
In addition to having copies of the recovery documents located at their offices and
homes, key individuals should have easily accessible versions of critical procedures and
call tree information. One simple way to accomplish the latter is to publish a call tree
on cards that can be affixed to personnel badges or kept in a wallet. In an emergency
situation, valuable minutes are better spent responding to an incident than looking for a
document or having to wait for a laptop to power up. Of course, the call tree is only as

effective as it is accurate and up to date, so verifying it periodically is imperative.
One limitation of call trees is that they are point to point, which means they're typically good for getting the word out, but not so much for coordinating activities.
Group text messages work better, but only in the context of fairly small and static groups.
Many organizations have group chat solutions, but if those rely on the organization's servers, they may be unavailable during a disaster. It is a good idea, then, to establish

a communications platform that is completely independent of the organizational infrastructure. Solutions like Slack and Mattermost offer a free service that is typically sufficient to keep most organizations connected in emergencies. The catch, of course, is that everyone needs to have the appropriate client installed on their personal devices and know when and how to connect. Training and exercises are the keys to successful execution of any plan, and the communications plan is no exception.
NOTE An organization may need to solidify communications channels and relationships with government officials and emergency response groups. The goal of this activity is to solidify proper protocol in case of a city- or region-wide disaster. During the BIA phase, the DR team should contact local authorities to elicit information about the risks of its geographical location and how to access emergency zones. If the organization has to perform DR, it may need to contact many of these emergency response groups.

PACE Communications Plans
The U.S. armed forces routinely develop Primary, Alternate, Contingency, and Emergency (PACE) communications plans. The PACE plan outlines the different capabilities that exist and aligns them into these four categories based on their ability to meet defined information exchange requirements. Each category is defined here:

The PACE plan includes redundant communications capabilities and specifies the order in which the organization will employ the capabilities when communication outages occur.

- Primary The normal or expected capability that is used to achieve the objective.
- Alternate A fully satisfactory capability that can be used to achieve the objective with minimal impact to the operation or exercise. This capability is used when the Primary capability is unavailable.
- Contingency A workable capability that can be used to achieve the objective.

This capability may not be as fast or easy as the Primary or Alternate but is capable of achieving the objective with an acceptable amount of time and effort. This capability is used when the Primary and the Alternate capabilities are unavailable.
• Emergency This is the last-resort capability and typically may involve significantly more time and effort than any of the other capabilities. This capability should be used only when the Primary, Alternate, and Contingency capabilities are unavailable.

Assessment
A role, or a team, needs to be created to carry out a damage assessment once a disaster has
taken place. The assessment procedures should be properly documented in the DRP and
include the following steps:

• Determine the cause of the disaster.
• Determine the potential for further damage.
• Identify the affected business functions and areas.
• Identify the level of functionality for the critical resources.
• Identify the resources that must be replaced immediately.
• Estimate how long it will take to bring critical functions back online.
After the damage assessment team collects and assesses this information, the DR coordinator identifies which teams need to be called to action and which system-specific
DRPs need to be executed (and in what order). The DRP should specify activation criteria for the different teams and system-specific DRPs. After the damage assessment, if
one or more of the situations outlined in the criteria have taken place, then the DR team
is moved into restoration mode.
Different organizations have different activation criteria because business drivers and
critical functions vary from organization to organization. The criteria may comprise
some or all of the following elements:

• Danger to human life
• Danger to state or national security
• Damage to facility
• Damage to critical systems
• Estimated value of downtime that will be experienced

Restoration
Once the damage assessment is completed, various teams are activated, which signals the
organization’s entry into the restoration phase. Each team has its own tasks—for example,
the facilities team prepares the offsite facility (if needed), the network team rebuilds the

network and systems, and the relocation team starts organizing the staff to move into a
new facility.
The restoration process needs to be well organized to get the organization up and
running as soon as possible. This is much easier to state in a book than to carry out
in reality, which is why written procedures are critical. The critical functions and their
resources would already have been identified during the BIA, as discussed earlier in this
chapter (with a simplistic example provided in Table 23-1). These are the functions that
the teams need to work together on restoring first.

Many organizations create templates during the DR plan development stage. These
templates are used by the different teams to step them through the necessary phases
and to document their findings. For example, if one step could not be completed until
new systems were purchased, this should be indicated on the template. If a step is
partially completed, this should be documented so the team does not forget to go back
and finish that step when the necessary part arrives. These templates keep the teams on
task and also quickly tell the team leaders about the progress, obstacles, and potential
recovery time.
NOTE Examples of possible templates can be found in NIST Special
Publication 800-34, Revision 1, Contingency Planning Guide for Federal
Information Systems, which is available online at https://csrc.nist.gov/
publications/detail/sp/800-34/rev-1/final.

An organization is not out of an emergency state until it is back in operation at the
original primary site or at a new site that was constructed to replace the primary original
one, because the organization is always vulnerable while operating in a backup facility.
Many logistical issues need to be considered as to when an organization should return
from the alternate site to the primary one. The following lists a few of these issues:

- Ensuring the safety of employees
- Ensuring an adequate environment is provided (power, facility infrastructure, water, HVAC)
- Ensuring that the necessary equipment and supplies are present and in working order
- Ensuring proper communications and connectivity methods are working

• Properly testing the new environment

Once the coordinator, management, and salvage team sign off on the readiness of the primary site, the salvage team should carry out the following steps:

• Back up data from the alternate site and restore it within the primary site.
• Carefully terminate contingency operations.
• Securely transport equipment and personnel to the primary site.

PART VII

The least critical functions should be moved back first, so if there are issues in network configurations or connectivity, or important steps were not carried out, the critical operations of the organization are not negatively affected. Why go through the trouble of moving the most critical systems and operations to a safe and stable alternate site, only to return them to a main site that is untested? Let the less critical departments act as the

Incident

Time zero

canary in the coal mine. If they survive, then move the more critical components of the organization to the main site.

Overall recovery objective:
Back to normal as quickly as possible

Timeline

Damage assessment
Within minutes to hours:
staff and visitors accounted for;
casualties dealt with; damage
assessment; call appropriate
teams into action

Recovery
Within minutes to days:
contact staff, customers,
suppliers, etc.; recovery of
critical-business activities;
rebuild lost work in progress

Reconstitution
Within weeks to months:

damage repair/replacement;
relocation to permanent place of
work; resume normal working;
recovery of cost from insurers

Training and Awareness
Training your DR team on the execution of a DRP is critical for at least three reasons.
First, it allows you to validate that the plan will actually work. If your DR team is doing
a walkthrough exercise in response to a fictitious scenario, you'll find out very quickly
whether the plan would work or not. If it doesn't work in a training event when the stress
level and stakes are low, then there is no chance it would work in a real emergency.
Another reason to train is to ensure that everyone knows what they're supposed to do,
when, where, and how. Disasters are stressful, messy affairs and key people may not be
thinking clearly. It is important for them to have a familiar routine to fall back on. In a
perfect world, you would train often enough for your team to develop "muscle memory"
that allows them to automatically do the right things without even thinking.
Lastly, training can help establish that you are exercising due care. This could keep
you out of legal trouble in the aftermath of a disaster, particularly if people end up
getting hurt. A good plan and evidence of a trained workforce can go a long way to
reduce liability if regulators or other investigators come knocking. As always, consult
your attorneys to ensure you are meeting all applicable legal and regulatory obligations.
When thinking of training and "muscle memory," you should also consider everyone
else in the organization that is not part of the DR team. You want all your staff to have
an awareness of the major things they need to do to support DR. This is why many of
us conduct fire drills in our facilities: to ensure everyone knows how to get out of the

building and where to assemble if we ever face this particular kind of disaster. There
are many types of DR awareness events you can run, but you should at least consider
three types of responses that everyone should be aware of: evacuations (e.g., for fires or
explosives), shelter-in-place (e.g., for tornadoes or active shooters), and remain-at-home

(e.g., for overnight flooding).

Lessons Learned

As mentioned on the first page of this chapter, no battle plan ever survived first contact with the enemy. When you try to execute your DRP in a real disaster, you will find the need to disregard parts of it, make on-the-fly changes to others, and faithfully execute the rest. This is why you should incorporate lessons learned from any actual disasters and actual responses. The DR team should perform a "postmortem" on the response and ensure that necessary changes are made to plans, contracts, personnel, processes, and procedures.

Military organizations collect lessons learned in two steps. The first steps, called a hotwash, is a hasty one that happens right after the event is concluded (i.e., restoration is completed). The term comes from the military practice of dousing rifles with very hot water immediately after an engagement to quickly get the worst grit and debris off their weapons. The reason you want to conduct a hotwash right away is that memories will be freshest right after restoring the systems. The idea is not necessarily to figure out how to fix anything, but rather to quickly list as many things that went well or poorly as possible before participants start to forget them.

The second event at which lessons learned are collected in the military is much more deliberate. An after-action review (AAR) happens several days after completion of the DR and allows participants to think things through and start formulating possible ways to do better in the future. The AAR facilitator, ideally armed with the notes from the hotwash, presents each issue that was recorded (good or bad), a brief discussion of it, and then opens the floor for recommendations. Keep in mind that since you're dealing with things that went well or poorly, sometimes the group recommendation will be to "sustain" the issue or, in other words, keep doing things the same way in the future. More frequently, however, there are at least minor tweaks that can improve future performance.

Testing Disaster Recovery Plans

PART VII

The disaster recovery plan should be tested regularly because environments continually

change. Interestingly, many organizations are moving away from the concept of “testing,”
because a test naturally leads to a pass or fail score, and in the end, that type of score is
not very productive. Instead, many organizations are adopting the concept of “exercises,”
which appear less stressful, better focused, and ultimately more productive to
the participants. Each time the DRP is exercised or tested, improvements and efficiencies are
generally uncovered, yielding better and better results over time. The responsibility of
establishing periodic exercises and the maintenance of the plan should be assigned to a
specific person or persons who will have overall ownership responsibilities for the disaster
recovery initiatives within the organization.

The maintenance of the DRP should be incorporated into change management
procedures. That way, any changes in the environment are reflected in the plan itself.
Tests and disaster recovery exercises should be performed at least once a year. An
organization should have no real confidence in a developed plan until it has actually
been tested. Exercises prepare personnel for what they may face and provide a controlled
environment to learn the tasks expected of them. These exercises also point out issues to
the planning team and management that may not have been previously thought about
and addressed as part of the planning process. The exercises, in the end, demonstrate
whether an organization can actually recover after a disaster.
The exercise should have a predetermined scenario that the organization may indeed
be faced with one day. Specific parameters and a scope of the exercise must be worked
out before sounding the alarms. The team of testers must agree upon what exactly is
getting tested and how to properly determine success or failure. The team must agree
upon the timing and duration of the exercise, who will participate in the exercise, who
will receive which assignments, and what steps should be taken. Also, the team needs
to determine whether hardware, software, personnel, procedures, and communications
lines are going to be tested and whether it is all or a subset of these resources that will
be included in the event. If the test will include moving some equipment to an alternate
site, then transportation, extra equipment, and alternate site readiness must be

addressed
and assessed.
Most organizations cannot afford to have these exercises interrupt production or productivity, so the exercises may need to take place in sections or at specific times,
which will require logistical planning. Written exercise plans should be developed that
will test for specific weaknesses in the overall DRP. The first exercises should not include
all employees, but rather a small representative sample of the organization. This allows
both the planners and the participants to refine the plan. It also allows each part of the
organization to learn its roles and responsibilities. Then, larger exercises can take place so
overall operations will not be negatively affected.
The people conducting these exercises should expect to encounter problems and mistakes. After all, identifying potential problems and mistakes is why they are conducting the exercises in the first place. An organization would rather have employees
make mistakes during an exercise so they can learn from them and perform their tasks
more effectively during a real disaster.
NOTE After a disaster, telephone service may not be available. For communications purposes, alternatives should be in place, such as mobile phones or hand-held radios.

A few different types of exercises and tests can be used, each with its own pros and
cons. The following sections explain the different types of assessment events.

Checklist Test
In this type of test, copies of the DRP are distributed to the different departments and
functional areas for review. This enables each functional manager to review the plan

and indicate if anything has been left out or if some approaches should be modified or
deleted. This method ensures that nothing is taken for granted or omitted, as might
be the case in a single-department review. Once the departments have reviewed their
copies and made suggestions, the planning team then integrates those changes into the
master plan.
NOTE The checklist test is also called the desk check test.

Structured Walkthrough Test
In this test, representatives from each department or functional area come together and

go over the plan to ensure its accuracy. The group reviews the objectives of the plan; discusses the scope and assumptions of the plan; reviews the organization's reporting structure; and evaluates the testing, maintenance, and training requirements described. This gives the people responsible for making sure a disaster recovery happens effectively and efficiently an opportunity to review what has been decided upon and what is expected of them.

The group walks through different scenarios of the plan from beginning to end to make sure nothing was left out. This also raises the awareness of team members about the recovery procedures.

Tabletop Exercises

PART VII

Tabletop exercises (TTXs) may or may not happen at a tabletop, but they do not involve a technical control infrastructure. TTXs can happen at an executive level (e.g., C-suite) or at a team level (e.g., SOC), or anywhere in between. The idea is usually to test procedures and ensure they actually do what they're intended to and that everyone knows their role in responding to a disaster. TTXs require relatively few resources apart from deliberate planning by qualified individuals and the undisturbed time and attention of the participants.

After determining the goals of the exercise and vetting them with the senior leadership of the organization, the planning team develops a scenario that touches on the important aspects of the response plan. The idea is normally not to cover every contingency, but to ensure the DR team is able to respond to the likeliest and/or most dangerous scenarios.

As they develop the exercise, the planning team considers branches and sequels at every point in the scenario. A branch is a point in which the participants may choose one of multiple approaches to respond. If the branches are not carefully managed and controlled, the TTX could wander into uncharted and unproductive directions. Conversely, a sequel is a follow-on to a given action in the response. For instance, as part of the response, the strategic communications team may issue statements to the news media. A sequel to that could involve a media outlet challenging the statement, which in turn would require a response by the team. Like branches, sequels must be used carefully to keep the

exercise on course. Senior leadership support and good scenario development are critical
ingredients to attract and engage the right participants. Like any contest, a TTX is only
as good as the folks who show up to play.

EXAM TIP Tabletop exercises are also called read-through exercises.

Simulation Test

This type of test takes a lot more planning and people. In this situation, all employees who participate in operational and support functions, or their representatives, come
together to practice executing the disaster recovery plan based on a specific scenario. The
scenario is used to test the reaction of each operational and support representative. Again,
this is done to ensure specific steps were not left out and that certain threats were not
overlooked. It raises the awareness of the people involved.
The exercise includes only those materials that will be available in an actual disaster,
to portray a more realistic environment. The simulation test continues up to the point
of actual relocation to an offsite facility and actual shipment of replacement equipment.

Parallel Test

In a parallel test, some systems are moved to the alternate site and processing takes place.
The results are compared with the regular processing that is done at the original site. This
ensures that the specific systems can actually perform adequately at the alternate offsite
facility and points out any tweaking or reconfiguring that is necessary.

Full-Interruption Test

This type of test is the most intrusive to regular operations and business productivity.
The original site is actually shut down, and processing takes place at the alternate
site. The recovery team fulfills its obligations in preparing the systems and environment
for the alternate site. All processing is done only on devices at the alternate offsite facility.
This is a full-blown exercise that takes a lot of planning and coordination, but it
can reveal many holes in the plan that need to be fixed before an actual disaster hits.
Full-interruption tests should be performed only after all other types of tests have been
successful. They are the riskiest type and can impact the business in very

serious and
devastating ways if not managed properly; therefore, senior management approval needs
to be obtained prior to performing full-interruption tests.
The type of organization and its goals will dictate what approach to the training
exercise is most effective. Each organization may have a different approach and unique
aspects. If detailed planning methods and processes are going to be taught, then specific
training may be required rather than general training that provides an overview.
Higherquality training will result in an increase in employee interest and commitment.
During and after each type of test, a record of the significant events should be
documented and reported to management so it is aware of all outcomes of the test.

Other Types of Training
Other types of training that employees need in addition to disaster recovery training
include first aid and cardiac pulmonary resuscitation (CPR), how to properly use a fire
extinguisher, evacuation routes and crowd control methods, emergency communications
procedures, and how to properly shut down equipment in different types of disasters.

The more technical employees may need training on how to redistribute network
resources and how to use different telecommunications lines if the main one goes down.
They may need to know about redundant power supplies and be trained and tested on
the procedures for moving critical systems from one power supply to the next.

Business Continuity
When a disaster strikes, ensuring that the organization is able to continue its operations
requires more than simply restoring data from backups. Also necessary are the
detailed procedures that outline the activities to keep the critical systems
available and ensure that operations and processing are not interrupted.
Business continuity planning defines what should
take place during and after an incident. Actions that are required to take place for emergency
response, continuity of operations, and dealing with major outages must be documented
and readily available to the operations staff. There should be at least two instances of these
documents: the original that is kept on-site and a copy that is at an offsite location.
BC plans should not be trusted until they have been tested. Organizations should
carry out exercises to ensure that the staff fully understands their

responsibilities and
how to carry them out. We already covered the various types of exercises that can be used
to test plans and staff earlier in this chapter when we discussed DR. Another issue to
consider is how to keep these plans up to date. As our dynamic, networked environments
change, so must our plans on how to rescue them when necessary.
Although in the security industry “contingency planning” and “business continuity
planning (BCP)” are commonly used interchangeably, it is important that you
understand the actual difference for the CISSP exam. BCP addresses how to keep the
organization in business after a major disruption takes place. It is about the survivability
of the organization and making sure that critical functions can still take place even after a
disaster. Contingency plans address how to deal with small incidents that do not qualify
as disasters, as in power outages, server failures, a down communication link to the
Internet, or the corruption of software. Organizations must be ready to deal with both
large and small issues that they may encounter.
EXAM TIP BCP is broad in scope and deals with survival of the organization.
Contingency plans are narrow in scope and deal with specific issues.

BCP Life Cycle
Remember that most organizations aren’t static, but change, often rapidly, as do
the conditions under which they must operate. Thus, BCP should be considered a life cycle in order
to deal with the constant and inevitable change that will affect it.
Understanding and

PART VII

As a security professional you will most likely not be in charge of BCP, but you should
most certainly be an active participant in developing the BCP. You will also be involved
in BC exercises and may even be a lead in those that focus on information systems.
To effectively participate in BC planning and exercises, you should be familiar with the
BCP life cycle, how to ensure continuous availability of critical information systems,
and the particular requirements of the end-user environments. We look at these in the
following sections.

maintaining each step of the BCP life cycle is critical to ensuring that the BC

plan remains
useful to the organization. The BCP life cycle is outlined in Figure 23-7.
Note that this life cycle has two modes: normal management (shown in the top
half
of Figure 23-7) and incident management (shown in the bottom half). In the
normal
mode, the focus of the BC team is on ensuring preparedness. Obviously, we want
to start
Normal operations
Define the business
continuity concept
Operate and maintain
the continuity plans
and solutions

Assess current
environment

Manage the business
continuity life cycle

Test and exercise
the continuity plans
and solutions

Select and define
continuity strategies

Train users
and continuity
personnel

Design continuity
plans and solutions
Implement continuity
plans and solutions

Incident occurs
Return to normal
operations at
primary site

Employ highavailability systems
and services

Conduct resumption
at alternate site
Reconstitute
primary site

Conduct emergency
response

Activate

resumption
plan?
Yes

Figure 23-7

BCP life cycle

Operate from
alternate site

No
Ready for
reconstitution?
Yes

No

with a clearly defined concept for what business continuity means for the organization.
What are the critical business functions that must continue to operate regardless of what
incident happens? What are the minimum levels of performance that are acceptable for
these functions?
Once we define the BC concept, we can take a look at the current environment and consider the strategies that would allow continuity of operations under a variety
of conditions. It is important to consider that, unlike DR planning, not every type of
incident covered in BCP involves loss of IT capabilities. Many organizations suffered
tremendously in 2020 because their BCP didn’t account for a global pandemic in which
many (or even all) staff members would have to work from home for extended periods
of time. Information systems are certainly an important part of the continuity strategies,
plans, and solutions, but the scope of the BCP is much broader than that of the DRP.
The BC plan is only useful if the organization in general, and the BC team in particular, knows how to execute the plan. This requires periodic training, tests, and
exercises to ensure that both the plan and the staff are able to keep the business going no
matter what comes their way. As we find gaps and opportunities for improvement, we get
to redefine our BCP concept and start another run through the cycle. This continuous
improvement is key to being able to switch into incident management mode (at the bottom of Figure 23-7) when needed and execute the BC plan (and, potentially,

the DR
plan) to keep the business going.

Information Systems Availability
Our main job as CISSPs in the BCP life cycle is to ensure the continuous availability
of organizational information systems. To this end, we should ensure the BCP includes
backup solutions for the following:

- Network and computer equipment
- Voice and data communications resources
- Human resources
- Transportation of equipment and personnel
- Environment issues (HVAC)
- Data and personnel security issues
- Supplies (paper, forms, cabling, and so on)
- Documentation

PART VII

The BCP team must understand the organization's current technical environment. This
means the planners have to know the intimate details of the network, communications
technologies, computers, network equipment, and software requirements that are
necessary to get the critical functions up and running. What is surprising to some people
is that many organizations do not totally understand how their network is configured
and how it actually works, because the network may have been established 10 to 15 years
ago and has kept growing and changing under different administrators and personnel.

Outsourcing
Part of the planned response to a disaster may be to outsource some of the affected
activities to another organization. Organizations do outsource activities—help-desk
services, manufacturing, legal advice—all the time, so why not important functions
affected by a disaster? Some companies specialize in disaster response and continuity
planning and can act as expert consultants.
That is all well and good. However, be aware that your organization is still
ultimately responsible for the continuity of a product or service that is outsourced.
Clients and customers will expect the organization to ensure continuity of its
products and services, either by itself or by having chosen the right outside vendors
to provide the products and services. If outside vendors are brought in, the

active
participation of key in-house managers in their work is still essential. They still need
to supervise the work of the outside vendors.
This same concern applies to normal, third-party suppliers of goods and services to the organization. Any BCP should take them into account as well. Note that the
process for evaluating an outsourced company for BCP is like that for evaluating the
organization itself. The organization must make sure that the outsourced company is financially viable and has its own solid BCP.
The organization can take the following steps to better ensure the continuity of its outsourcing:

• Make the ability of such companies to reliably assure continuity of products and services part of any work proposals.
• Make sure that business continuity planning is included in contracts with such companies, and that their responsibilities and levels of service are clearly spelled out.
• Draw up realistic and reasonable service levels that the outsourced firm will meet during an incident.
• If possible, have the outsourcing companies take part in BCP awareness programs, training, and testing.
The goal is to make the supply of goods and services from outsources as resilient
as possible in the wake of a disaster.

New devices are added, new computers are added, new software packages are added, VoIP may have been integrated, and the DMZ may have been split up into three DMZs,
with an extranet for the organization’s partners. Maybe a company bought and merged
with another company and network. Over ten years, a number of technology refreshes
most likely have taken place, and the individuals who are maintaining the environment
now likely are not the same people who built it ten years ago. Many IT departments
experience extensive employee turnover every five years. And most organizational network

schematics are notoriously out of date because everyone is busy with their current tasks
(or will come up with new tasks just to get out of having to update the schematic).
So the BCP team has to make sure that if the networked environment is partially or
totally destroyed, the recovery team has the knowledge and skill to properly rebuild it.
NOTE Many organizations use VoIP, which means that if the network goes down, network and voice capability are unavailable. The BCP team should

address the possible need of redundant voice systems.

The BCP team needs to incorporate into the BCP several things that are commonly overlooked, such as hardware replacements, software products, documentation, environmental needs, and human resources.

Hardware Backups

PART VII

The BCP needs to identify the equipment required to keep the critical functions up and running. This may include servers, user workstations, routers, switches, tape backup devices, and more. The needed inventory may seem simple enough, but as they say, the devil is in the details. If the recovery team is planning to use images to rebuild newly purchased servers and workstations because the original ones were destroyed, for example, will the images work on the new computers? Using images instead of building systems from scratch can be a time-saving task, unless the team finds out that the replacement equipment is a newer version and thus the images cannot be used. The BCP should plan for the recovery team to use the organization's current images, but also have a manual process of how to build each critical system from scratch with the necessary configurations.
The BCP also needs to be based on accurate estimates of how long it will take for new equipment to arrive. For example, if the organization has identified Dell as its equipment replacement supplier, how long will it take this vendor to send 20 servers and 30 workstations to the offsite facility? After a disaster hits, the organization could be in its offsite facility only to find that its equipment will take three weeks to be delivered. So, the SLA for the identified vendors needs to be investigated to make sure the organization is not further damaged by delays. Once the parameters of the SLA are understood, the BCP team must make a decision between depending upon the vendor and purchasing redundant systems and storing them as backups in case the primary equipment is destroyed.
As described earlier, when potential organizational risks are identified, it is better to take preventive steps to reduce the potential damage. After the calculation of the MTD values, the team will know how long the organization can operate without a specific device. This data should be used to make the decision on whether the organization

should depend on the vendor's SLA or make readily available a hot-swappable redundant
system. If the organization will lose $50,000 per hour if a particular server goes down,
then the team should elect to implement redundant systems and technology.
If an organization is using any legacy computers and hardware and a disaster hits
tomorrow, where would it find replacements for this legacy equipment? The BCP

team should identify legacy devices and understand the risk the organization is facing
if replacements are unavailable. This finding has caused many organizations to move
from legacy systems to commercial off-the-shelf (COTS) products to ensure that timely
replacement is possible.

Software Backups
Most organizations' IT departments have their array of software disks and licensing
information here or there—or possibly in one centralized location. If the facility were
destroyed and the IT department's current environment had to be rebuilt, how would
it gain access to these software packages? The BCP team should make sure to have an
inventory of the necessary software required for mission-critical functions and have
backup copies at an offsite facility. Hardware is usually not worth much to an
organization without the software required to run on it. The software that needs to be backed
up can be in the form of applications, utilities, databases, and operating systems. The
business continuity plan must have provisions to back up and protect these items along
with hardware and data.
It is common for organizations to work with software developers to create customized
software programs. For example, in the banking world, individual financial institutions
need software that enables their bank tellers to interact with accounts, hold account
information in databases and mainframes, provide online banking, carry out data
replication, and perform a thousand other types of bank-like functionalities. This
specialized type of software is developed and available through a handful of software
vendors that specialize in this market. When bank A purchases this type of software for
all of its branches, the software has to be specially customized for its environment and

needs. Once this banking software is installed, the whole organization depends upon it
for its minute-by-minute activities.
When bank A receives the specialized and customized banking software from the software vendor, bank A does not receive the source code. Instead, the software vendor
provides bank A with a compiled version. Now, what if this software vendor goes out of
business because of a disaster or bankruptcy? Then bank A will require a new vendor to
maintain and update this banking software; thus, the new vendor will need access to the
source code.
The protection mechanism that bank A should implement is called software escrow, in
which a third party holds the source code, backups of the compiled code, manuals, and
other supporting materials. A contract between the software vendor, customer, and third
party outlines who can do what, and when, with the source code. This contract usually
states that the customer can have access to the source code only if and when the vendor
goes out of business, is unable to carry out stated responsibilities, or is in breach of the
original contract. If any of these activities takes place, then the customer is protected
because it can still gain access to the source code and other materials through the thirdparty escrow agent.
Many organizations have been crippled by not implementing software escrow. They paid a software vendor to develop specialized software, and when the software vendor
went belly up, the organizations did not have access to the code that their systems ran on.

End-User Environment
Because the end users are usually the worker bees of an organization, they must be provided a functioning environment as soon as possible after a disaster hits. This means
that the BCP team must understand the current operational and technical functioning
environment and examine critical pieces so they can replicate them.
In most situations, after a disaster, only a skeleton crew is put back to work. The
BCP committee has previously identified the most critical functions of the organization
during the analysis stage, and the employees who carry out those functions must be put
back to work first. So the recovery process for the user environment should be laid out

in different stages. The first stage is to get the most critical departments back online, the next stage is to get the second most important back online, and so on.

The BCP team needs to identify user requirements, such as whether users can work on stand-alone PCs or need to be connected in a network to fulfill specific tasks. For example, in a financial institution, users who work on stand-alone PCs might be able to accomplish some small tasks like filling out account forms, word processing, and accounting tasks, but they might need to be connected to a host system to update customer profiles and to interact with the database.

The BCP team also needs to identify how current automated tasks can be carried out manually if that becomes necessary. If the network is going to be down for 12 hours, could the necessary tasks be accomplished through traditional pen-and-paper methods?

If the Internet connection is going to be down for five hours, could the necessary communications take place through phone calls? Instead of transmitting data through the internal mail system, could couriers be used to run information back and forth?

Today, we are extremely dependent upon technology, but we often take for granted that it will always be there for us to use. It is up to the BCP team to realize that technology may be unavailable for a period of time and to come up with solutions for those situations.

EXAM TIP As a CISSP, your role in business continuity planning is most likely to be that of an active participant, not to lead it. BCP questions in the exam will be written with this in mind.

Chapter Review

PART VII

There are four key take-aways in this chapter. The first is that you need to be able to identify and implement strategies that will enable your organization to recover from any disaster, supporting your organization’s continuity of operations. Leveraging these strategies, you develop a detailed plan that includes the specific processes that the organization (and particularly the IT and security teams) will execute to recover from specific types of disasters. Thirdly, you have to know how to train your DR team to execute the plan flawlessly, even in the chaos of an actual disaster. This includes ensuring that everyone in the organization is aware of their role in the recovery efforts. Finally, the DRP is the cornerstone of the BCP, so you will be called upon to participate in broader business

continuity planning and exercises, even if you are not in charge of that effort.

Quick Review
• Disaster recovery (DR) is the set of practices that enables an organization to minimize loss of, and restore, mission-critical technology infrastructure after a
catastrophic incident.
• Business continuity (BC) is the set of practices that enables an organization to
continue performing its critical functions through and after any disruptive event.
• The recovery time objective (RTO) is the maximum time period within which a mission-critical system must be restored to a designated service level after a disaster to avoid unacceptable consequences associated with a break in business continuity.
• The work recovery time (WRT) is the maximum amount of time available for certifying the functionality and integrity of restored systems and data so they can
be put back into production.
• The recovery point objective (RPO) is the acceptable amount of data loss measured in time.
• The four commonly used data backup strategies are direct-attached storage, network-attached storage, cloud storage, and offline media.
• Electronic vaulting makes copies of files as they are modified and periodically
transmits them to an offsite backup site.
• Remote journaling moves transaction logs to an offsite facility for database recovery, where only the reapplication of a series of changes to individual records
is required to resynchronize the database.
• Offsite backup locations can supply hot, warm, or cold sites.
• A hot site is fully configured with hardware, software, and environmental needs.
It can usually be up and running in a matter of hours. It is the most expensive option, but some organizations cannot be out of business longer than a day without very detrimental results.
• A warm site may have some computers, but it does have some peripheral devices, such as disk drives, controllers, and tape drives. This option is less expensive than
a hot site, but takes more effort and time to become operational.
• A cold site is just a building with power, raised floors, and utilities. No devices
are available. This is the cheapest of the three options, but can take weeks to get
up and operational.
• In a reciprocal agreement, one organization agrees to allow another organization
to use its facilities in case of a disaster, and vice versa. Reciprocal agreements
are very tricky to implement and may be unenforceable. However, they offer

a relatively cheap offsite option and are sometimes the only choice.
• A redundant (or mirrored) site is equipped and configured exactly like the primary site and is completely synchronized, ready to become the primary site at a moment’s notice.

• High availability (HA) is a combination of technologies and processes that work
together to ensure that some specific thing is up and running most of the time.
• Quality of service (QoS) defines minimum acceptable performance characteristics
of a particular service, such as response time, CPU utilization, or network bandwidth utilization.
• Fault tolerance is the capability of a technology to continue to operate as expected
even if something unexpected takes place (a fault).
• Resilience means that the system continues to function, albeit in a degraded fashion, when a fault is encountered.
• When returning to the original site after a disaster, the least critical organizational
units should go back first.
• Disaster recovery plans can be tested through checklist tests, structured walkthroughs, tabletop exercises, simulation tests, parallel tests, or fullinterruption tests.
• Business continuity planning addresses how to keep the organization in business
after a major disruption takes place, but it is important to note that the scope is
much broader than that of disaster recovery.
• The BCP life cycle includes developing the BC concept; assessing the current environment; implementing continuity strategies, plans, and solutions; training the staff; and testing, exercising, and maintaining the plans and solutions.
• An important part of the business continuity plan is to communicate its requirements and procedures to all employees.

Questions
Please remember that these questions are formatted and asked in a certain way for a
reason. Keep in mind that the CISSP exam is asking questions at a conceptual level.
Questions may not always have the perfect answer, and the candidate is advised against
always looking for the perfect answer. Instead, the candidate should look for the best
answer in the list.
1. Which best describes a hot-site facility versus a warm- or cold-site facility?
A. A site that has disk drives, controllers, and tape drives
B. A site that has all necessary PCs, servers, and telecommunications
D. A mobile site that can be brought to the organization’s parking lot

2. Which of the following describes a cold site?

A. Fully equipped and operational in a few hours
B. Partially equipped with data processing equipment
C. Expensive and fully configured
D. Provides environmental measures but no equipment

PART VII

C. A site that has wiring, central air-conditioning, and raised flooring

3. Which is the best description of remote journaling?
A. Backing up bulk data to an offsite facility
B. Backing up transaction logs to an offsite facility
C. Capturing and saving transactions to two mirrored servers in-house
D. Capturing and saving transactions to different media types

4. Which of the following does not describe a reciprocal agreement?
A. The agreement is enforceable.
B. It is a cheap solution.
C. It may be able to be implemented right after a disaster.
D. It could overwhelm a current data processing site.

5. If a system is fault tolerant, what would you expect it to do?
A. Continue to operate as expected even if something unexpected takes place
B. Continue to function in a degraded fashion
C. Tolerate outages caused by known faults
D. Raise an alarm, but tolerate an outage caused by any fault

6. Which of the following approaches to testing your disaster recovery plan would
be least desirable if you had to maintain high availability of over 99.999 percent?
A. Checklist test
B. Parallel test
C. Full-interruption test
D. Structured walkthrough test

Use the following scenario to answer Questions 7–10. You are the CISO of a small research
and development (R&D) company and realize that you don’t have a disaster recovery
plan (DRP). The projects your organization handles are extremely sensitive and, despite
having a very limited budget, you have to bring the risk of project data being lost as close
to zero as you can. Recovery time is not as critical because you bill your work based on
monthly deliverables and have some leeway at your disposal. Because of the sensitivity
of your work, remote working is frowned upon and you keep your research data on local
servers (including Exchange for e-mail, Mattermost for group chat, and Apache

for web)
at your headquarters (and only) site.
7. Which recovery site strategy would be best for you to consider?
A. Reciprocal agreement
B. Hot site
C. Warm site
D. Cold site

8. Which of the following recovery site characteristics would be best for your organization?
A. As close to headquarters as possible within budgetary constraints
B. 100 miles away from headquarters, on a different power grid
C. 15 miles away from headquarters on a different power grid
D. As far away from headquarters as possible

9. Which data backup storage strategy would you want to implement?
A. Direct-attached storage
B. Network-attached storage
C. Offline media
D. Cloud storage

10. Which of the following would be the best way to communicate with all members of the organization in the event of a disaster that takes out your site?
A. Internal Mattermost channel
B. External Slack channel
C. Exchange e-mail
D. Call trees

Answers
1. B. A hot site is a facility that is fully equipped and properly configured so that it
can be up and running within hours to get an organization back into production. Answer B gives the best definition of a fully functional environment.
2. D. A cold site only provides environmental measures—wiring, HVAC, raised floors—basically a shell of a building and no more.
3. B. Remote journaling is a technology used to transmit data to an offsite facility,
but this usually only includes moving the journal or transaction logs to the offsite
facility, not the actual files.

5. A. Fault tolerance is the capability of a technology to continue to operate as
expected even if something unexpected takes place (a fault), with no degradations
or outages.
6. C. A full-interruption test is the most intrusive to regular operations and business
productivity. The original site is actually shut down, and processing takes place
at the alternate site. This is almost guaranteed to exceed your allowed downtime

unless everything went extremely well.

PART VII

4. A. A reciprocal agreement is not enforceable, meaning that the organization that
agreed to let the damaged organization work out of its facility can decide not to allow this to take place. A reciprocal agreement is a better secondary backup option if the original plan falls through.

↟CISSP All-in-One Exam Guide

7. D. Because you are working on a tight budget and have the luxury of recovery time, you want to consider the least expensive option. A reciprocal agreement would be ideal except for the sensitivity of your data, which could not be shared
with a similar organization (that could, presumably, be a competitor at some point). The next option (cost-wise) is a cold site, which would work in the given
scenario.
8. C. An ideal recovery site would be on a different power grid to minimize the risk
that power will be out on both sites, but close enough for employees to commute. This second point is important because, due to the sensitivity of your work, your
organization has a low tolerance for remote work.
9. C. Since your data is critical enough that you have to bring the risk of it being
lost as close to zero as you can, you would want to use offline media such as tape
backups, optical discs, or even external drives that are disconnected after each backup (and potentially removed offsite). This is the slowest and most expensive approach, but is also the most resistant to attacks.
10. B. If your site is taken out, you would lose both Exchange and Mattermost since
those servers are hosted locally. Call trees only work well for initial notification,
leaving an externally hosted Slack channel as the best option. This would require
your staff to be aware of this means of communication and have accounts created before the disaster.

↟PART VIII

Software Development
Security
Chapter 24
Chapter 25

Software Development
Secure Software

↟This page intentionally left blank

CHAPTER

Software Development

This chapter presents the following:
- Software development life cycle
- Development methodologies
- Operation and maintenance
- Maturity models

Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.
—John F. Woods

Software is usually developed with a strong focus on functionality, not security. In many cases, security controls are bolted on as an afterthought (if at all). To get the best of both worlds, security and functionality have to be designed and integrated at each phase of the software development life cycle. Security should be interwoven into the core of a software product and provide protection at the necessary layers. This is a better approach than trying to develop a front end or wrapper that may reduce the overall functionality and leave security holes when the software has to be integrated into a production environment.

Before we get too deep into secure software development, however, we have to develop a shared understanding of how code is developed in the first place. In this chapter we will cover the complex world of software development so that we can understand the bad things that can happen when security is not interwoven into products properly (discussed in Chapter 25).

Software Development Life Cycle

The life cycle of software development deals with putting repeatable and predictable processes in place that help ensure functionality, cost, quality, and delivery schedule requirements are met. So instead of winging it and just starting to develop code for a project, how can we make sure we build the best software product possible?

Several software development life cycle (SDLC) models have been developed over the years, which we will cover later in this section, but the crux of each model deals with the

following phases:

• Requirements gathering Determining why to create this software, what the software will do, and for whom the software will be created
• Design Encapsulating into a functional design how the software will accomplish the requirements
• Development Programming software code to meet specifications laid out in the design phase and integrating that code with existing systems and/or libraries
• Testing Verifying and validating software to ensure that the software works as planned and that goals are met
• Operations and maintenance Deploying the software and then ensuring that it is properly configured, patched, and monitored
EXAM TIP You don’t need to memorize the phases of the SDLC. We discuss them here so you understand all the tasks that go into developing software and how to integrate security throughout the whole cycle.

In the following sections we will cover the different phases that make up an SDLC
model and some specific items about each phase that are important to understand.

Software Development Roles
The specific roles within a software development team will vary based on the methodology being used, the maturity of the organization, and the size of the project (to name just a few parameters). Typically, however, a team has at least the
following roles:

• Project manager (PM) This role has overall responsibility for the software development project, particularly with regard to cost, schedule, performance, and risk.
• Team leads It is rare for software projects to be tackled by a single team, so we usually divide them up and assign a good developer to lead each part.
• Architect Sometimes called a tech lead, this role figures out what technologies to use internally or when interfacing with external systems.
• Software engineer The people who actually write the programming code are oftentimes specialists in either frontends (e.g., user interfaces) or various
types of backends (e.g., business logic, databases). Engineers that can do all of this are called full-stack developers.
• Quality assurance (QA) Whether this is a single person or an entire team, this role implements and runs testing processes that detect software defects as early as possible.

Keep in mind that the discussion that follows covers phases that may happen repeatedly
and in limited scope depending on the development methodology being used. Before we get into the phases of the SDLC, let’s take a brief look at the glue that holds them
together: project management.

Project Management

PART VIII

Many developers know that good project management keeps the project moving in the right direction, allocates the necessary resources, provides the necessary leadership, and hopes for the best but plans for the worst. Project management processes should be put into place to make sure the software development project executes each life-cycle phase properly. Project management is an important part of product development, and security management is an important part of project management.

The project manager draws up a security plan at the beginning of a development project and integrates it into the functional plan to ensure that security is not overlooked.

This plan will probably be broad and should refer to documented references for more detailed information. The references could include computer standards (RFCs, IEEE standards, and best practices), documents developed in previous projects, security policies, accreditation statements, incident-handling plans, and national or international guidelines. This helps ensure that the plan stays on target.

The security plan should have a life cycle of its own. It will need to be added to, subtracted from, and explained in more detail as the project continues. Keeping the security plan up to date for future reference is important, because losing track of actions, activities, and decisions is very easy once a large and complex project gets underway.

The security plan and project management activities could be scrutinized later, particularly if a vulnerability causes losses to a third party, so we should document security-related decisions. Being able to demonstrate that security was fully considered in each phase of the SDLC can prove that the team exercised due care and this, in turn, can mitigate future liabilities. To this end, the documentation must accurately reflect how the product was built and how it is supposed to operate once implemented into an environment.

If a software product is being developed for a specific customer, it is common for a Statement of Work (SOW) to be developed, which describes the product and customer requirements. A detailed SOW helps to ensure that all stakeholders understand these requirements and don’t make any undocumented assumptions.

Sticking to what is outlined in the SOW is important so that scope creep does not take place. If the scope of a project continually extends (creeps) in an uncontrollable manner, the project may never end, not meet its goals, run out of funding, or all of the foregoing. If the customer wants to modify its requirements, it is important that the SOW is updated and funding is properly reviewed.

A work breakdown structure (WBS) is a project management tool used to define and group a project's individual work elements in an organized manner. It is a deliberate decomposition of the project into tasks and subtasks that result in clearly defined deliverables. The SDLC should be illustrated in a WBS format, so that each phase is properly addressed.

Requirements Gathering Phase

This is the phase in which everyone involved in the software development project attempts to understand why the project is needed and what the scope of the project entails. Typically, either a specific customer needs a new application or a demand for the product exists in the market. During this phase, the software development team examines the software's requirements and proposed functionality, engages in brainstorming sessions, and reviews obvious restrictions.

A conceptual definition of the project should be initiated and developed to ensure everyone is on the right page and that this is a proper product to develop. This phase could include evaluating products currently on the market and identifying any demands not being met by current vendors. This definition could also be a direct request for a specific product from a current or future customer.

Typically, the following tasks should be accomplished in this phase:

- Requirements gathering (including security ones)
- Security risk assessment
- Privacy risk assessment
- Risk-level acceptance

The security requirements of the product should be defined in the categories of availability, integrity, and confidentiality. What type of security is required for the software product and to what degree? Some of these requirements may come from applicable external regulations. For example, if the application will deal with payment cards, PCI DSS will dictate some requirements, such as encryption for card

information.
An initial security risk assessment should be carried out to identify the potential threats and their associated consequences. This process usually involves asking many, many questions to elicit and document the laundry list of vulnerabilities and threats, the probability of these vulnerabilities being exploited, and the outcome if one of these threats actually becomes real and a compromise takes place. The questions vary from product to product—such as its intended purpose, the expected environment it will be implemented in, the personnel involved, and the types of businesses that would purchase and use the product.
The sensitivity level of the data that many software products store and process has only increased in importance over the years. After a privacy risk assessment, a privacy impact rating can be assigned, which indicates the sensitivity level of the data that will be processed or accessible. Some software vendors incorporate the following privacy impact ratings in their software development assessment processes:

• P1, High Privacy Risk The feature, product, or service stores or transfers personally identifiable information (PII), monitors the user with an ongoing transfer of anonymous data, changes settings or file type associations, or installs software.
• P2, Moderate Privacy Risk The sole behavior that affects privacy in the feature, product, or service is a one-time, user-initiated, anonymous data transfer (e.g., the user clicks a link and is directed to a website).

• P3, Low Privacy Risk No behaviors exist within the feature, product, or service that affect privacy. No anonymous or personal data is transferred, no PII is stored on the machine, no settings are changed on the user’s behalf, and no software is installed.
The software vendor can develop its own privacy impact ratings and their associated definitions. As of this writing there are several formal approaches to conducting a privacy risk assessment, but none stands out as “the” standardized approach to defining a methodology for an assessment or these rating types, but as privacy increases in importance, we might see more standardization in these ratings and associated

metrics.
The team tasked with documenting the requirements must understand the criteria for risk-level acceptance to make sure that mitigation efforts satisfy these criteria. Which risks are acceptable will depend on the results of the security and privacy risk assessments.
The evaluated threats and vulnerabilities are used to estimate the cost/benefit ratios of the different security countermeasures. The level of each security attribute should be focused upon so that a clear direction on security controls can begin to take shape and can be integrated into the design and development phases.
The end state of the requirements gathering phase is typically a document called the Software (or System) Requirements Specification (SRS), which describes what the software will do and how it will perform. These two high-level objectives are also known as functional and nonfunctional requirements. A functional requirement describes a feature of the software system, such as reporting product inventories or processing customer orders. A nonfunctional requirement describes performance standards, such as the minimum number of simultaneous user sessions or the maximum response time for a query. Nonfunctional requirements also include security requirements, such as what data must be encrypted and what the acceptable cryptosystems are. The SRS, in a way, is a checklist that the software development team will use to develop the software and the customer will use to accept it.
The Unified Modeling Language (UML) is a common language used to graphically describe all aspects of software development. We will revisit it throughout the different phases, but in terms of software requirements, it allows us to capture both functional and nonfunctional requirements with use case diagrams (UCDs). We already saw these in Chapter 18 when we discussed testing of technical controls. If you look back to Figure 18-3, each use case (shown as verb phrases inside ovals) represents a high-level functional requirement. The associations can capture nonfunctional requirements through special labels, or these requirements can be spelled out in an accompanying use case description.

Design Phase

PART VIII

Once the requirements are formally documented, the software development team can begin figuring out how they will go about satisfying them. This is the phase

that starts
to map theory to reality. The theory encompasses all the requirements that were identified in the previous phase, and the design outlines how the product is actually going to
accomplish these requirements.

Some organizations skip the design phase, but this can cause major delays and redevelopment efforts down the road because a broad vision of the product needs to
be understood before looking strictly at the details. Instead, software development
teams should develop written plans for how they will build software that satisfies each
requirement. This plan usually comprises three different but interrelated models:

• Informational model Dictates the type of information to be processed and how it will move around the software system
• Functional model Outlines the tasks and functions the application needs to carry out and how they are sequenced and synchronized
• Behavioral model Explains the states the application will be in during and after specific transitions take place

For example, consider an antimalware software application. Its informational model
would dictate how it processes information, such as virus signatures, modified system
files, checksums on critical files, and virus activity. Its functional model would dictate
how it scans a hard drive, checks e-mail for known virus signatures, monitors critical
system files, and updates itself. Its behavioral model would indicate that when the system
starts up, the antimalware software application will scan the hard drive and memory
segments. The computer coming online would be the event that changes the state of the
application. If it finds a virus, the application would change state and deal with the virus
appropriately. Each state must be accounted for to ensure that the product does not go
into an insecure state and act in an unpredictable way.

The data from the informational, functional, and behavioral models is incorporated
into the software design document, which includes the data, architectural, and procedural
design, as shown in Figure 24-1.

Functional
model

Behavioral
model

Design

Data
design
Architectural
design

Informational
model
Procedural
design

Code
Program
modules

Test
Validated
software

Figure 24-1

Information from three models can go into the design.

From a security point of view, the following items should also be accomplished in the
design phase:

- Attack surface analysis
- Threat modeling

An attack surface is what is available to be used by an attacker against the product itself.
As an analogy, if you were wearing a suit of armor and it covered only half of your body,
the other half would be your vulnerable attack surface. Before you went into battle, you
would want to reduce this attack surface by covering your body with as much protective
armor as possible. The same can be said about software. The software development team
should reduce the attack surface as much as possible because the greater the attack surface
of software, the more avenues for the attacker; and hence, the greater the likelihood of
a successful compromise.
The aim of an attack surface analysis is to identify and reduce the amount of code
and functionality accessible to untrusted users. The basic strategies of attack surface
reduction are to reduce the amount of code running, reduce entry points

available to
untrusted users, reduce privilege levels as much as possible, and eliminate unnecessary
services. Attack surface analysis is generally carried out through specialized tools to
enumerate different parts of a product and aggregate their findings into a numeral value.
Attack surface analyzers scrutinize files, Registry keys, memory data, session information,
processes, and services details. A sample attack surface report is shown in Figure 24-2.

PART VIII

Figure 24-2 Attack surface analysis result

Threat modeling, which we covered in detail in Chapter 9 in the context of risk management, is a systematic approach used to understand how different threats could be
realized and how a successful compromise could take place. As a hypothetical example,
if you were responsible for ensuring that the government building in which you work is
safe from terrorist attacks, you would run through scenarios that terrorists would most
likely carry out so that you fully understand how to protect the facility and the people
within it. You could think through how someone could bring a bomb into the building,
and then you would better understand the screening activities that need to take place at
each entry point. A scenario of someone running a car into the building would bring
up the idea of implementing bollards around the sensitive portions of the facility. The
scenario of terrorists entering sensitive locations in the facility (data center, CEO office)
would help illustrate the layers of physical access controls that should be implemented.
These same scenario-based exercises should take place during the design phase of software development. Just as you would think about how potential terrorists could enter
and exit a facility, the software development team should think through how potentially
malicious activities can happen at different input and output points of the software and
the types of compromises that can take place within the guts of the software itself.
It is common for software development teams to develop threat trees, as shown in Figure 24-3. A threat tree is a tool that allows the development team to understand all the

ways specific threats can be realized; thus, it helps them understand what type of security
controls they should implement in the software to mitigate the risks associated with each
threat type.

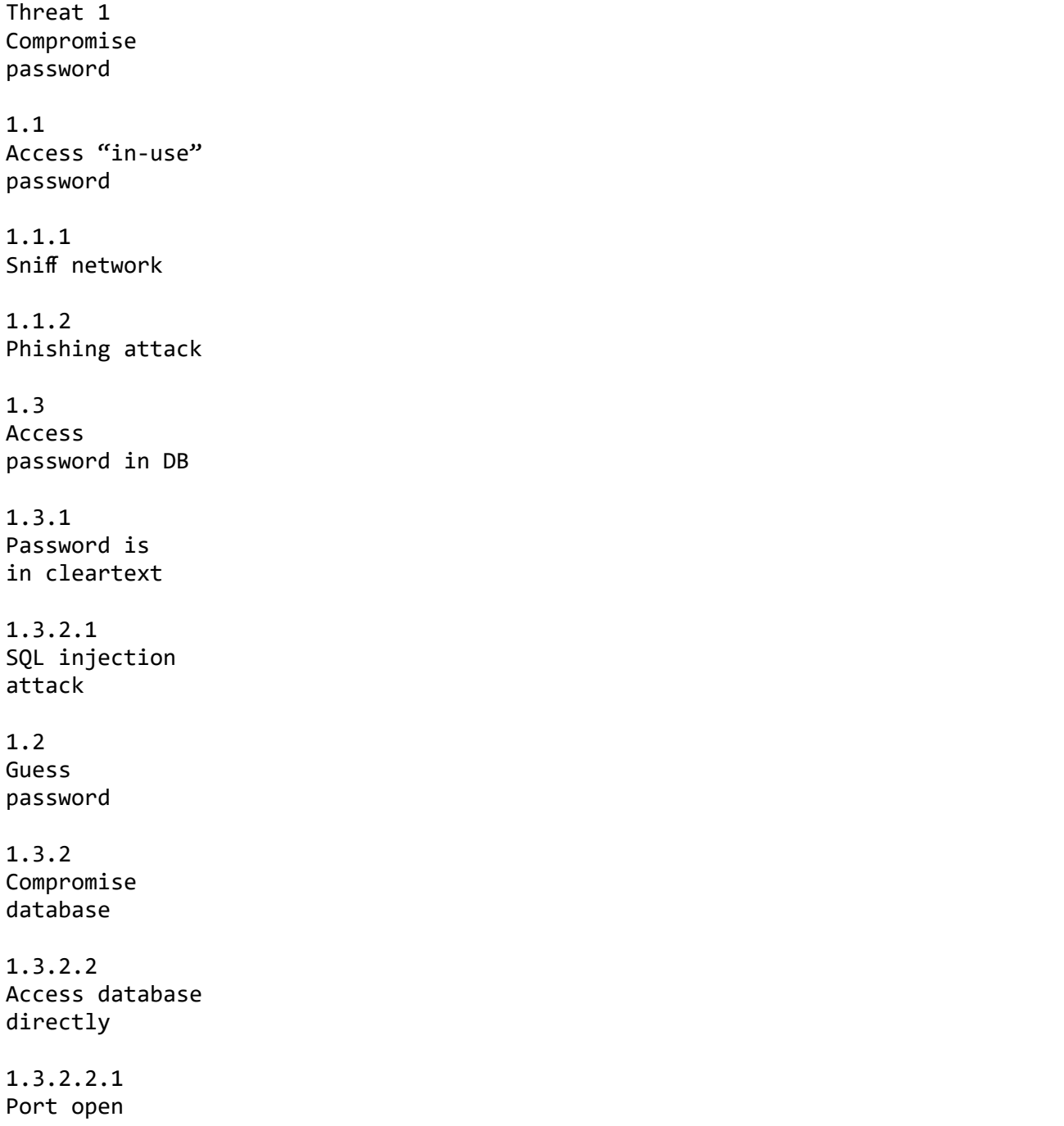

Figure 24-3 Threat tree used in threat modeling

password(s)

1.2.2
Brute-force
attack

1087
Figure 24-4
A simple flow
diagram for
threat modeling

There are many automated tools in the industry that software development teams can use to ensure that they address the various threat types during the design stage. One popular open-source solution is the Open Web Application Security Project (OWASP) Threat Dragon. This web-based tool enables the development team to describe threats visually using flow diagrams. Figure 24-4 shows a simple diagram of a three-tier web system showing its trust boundary and the four ways in which the tiers interact. The next step in building the threat model would be to consider how each of these four interactions could be exploited by a threat actor. For example, stolen credentials could allow an adversary to compromise the web server and, from there, issue queries to the database server that could compromise the integrity or availability of records stored there. For each threat identified through this process, the software development team would develop controls to mitigate it.

The decisions made during the design phase are pivotal steps to the development phase. Software design serves as a foundation and greatly affects software quality. If good
product design is not put into place in the beginning of the project, the following phases
will be much more challenging.

Development Phase

PART VIII

This is the phase where the programmers become deeply involved. The software design
that was created in the previous phase is broken down into defined deliverables, and
programmers develop code to meet the deliverable requirements.
There are many computer-aided software engineering (CASE) tools that programmers can use to generate code, test software, and carry out debugging activities. When these
types of activities are carried out through automated tools, development usually takes
place more quickly with fewer errors.
CASE refers to any type of software tool that supports automated development of software, which can come in the form of program editors, debuggers, code analyzers,
version-control mechanisms, and more. These tools aid in keeping detailed records
of requirements, design steps, programming activities, and testing. A CASE tool is
designed to support one or more software engineering tasks in the process of developing
software. Many vendors can get their products to the market faster because they are
“computer aided.”

In the next chapter we will delve into the abyss of “secure coding,” but let’s take a quick
peek at it here to illustrate its importance in the development phase. As stated previously,
most vulnerabilities that corporations, organizations, and individuals have to worry
about reside within the programming code itself. When programmers do not follow strict and secure methods of creating programming code, the effects can be widespread
and the results can be devastating. But programming securely is not an easy task. The list
of errors that can lead to serious vulnerabilities in software is long.
The MITRE organization’s Common Weakness Enumeration (CWE) initiative (https://cwe.mitre.org/top25) describes “a demonstrative list of the most common and
impactful issues experienced over the previous two calendar years.” Table 24-1

shows the
most recent list.
Rank

Name

1

Out-of-bounds Write

2

Improper Neutralization of Input During Web Page Generation (“Cross-site Scripting”)

3

Out-of-bounds Read

4

Improper Input Validation

5

Improper Neutralization of Special Elements used in an OS Command (“OS Command Injection”)

6

Improper Neutralization of Special Elements used in an SQL Command (“SQL Injection”)

7

Use After Free

8

Improper Limitation of a Pathname to a Restricted Directory (“Path Traversal”)

9

Cross-Site Request Forgery (CSRF)

10

Unrestricted Upload of File with Dangerous Type

11

Missing Authentication for Critical Function

12

Integer Overflow or Wraparound

13

Deserialization of Untrusted Data

14

Improper Authentication

15

NULL Pointer Dereference

16

Use of Hard-coded Credentials

17

Improper Restriction of Operations within the Bounds of a Memory Buffer

18

Missing Authorization

19

Incorrect Default Permissions

20

Exposure of Sensitive Information to an Unauthorized Actor

21

Insufficiently Protected Credentials

22

Incorrect Permission Assignment for Critical Resource

23

Improper Restriction of XML External Entity Reference

24

Server-Side Request Forgery (SSRF)

25

Improper Neutralization of Special Elements used in a Command (“Command

Injection”)

Table 24-1

2021 CWE Top 25 Most Dangerous Software Weaknesses List

Many of these software issues are directly related to improper or faulty programming practices. Among other issues to address, the programmers need to check input lengths so buffer overflows cannot take place, inspect code to prevent the presence of covert channels, check for proper data types, make sure checkpoints cannot be bypassed by users, verify syntax, and verify checksums. The software development team should play out different attack scenarios to see how the code could be attacked or modified in an unauthorized fashion. Code reviews and debugging should be carried out by peer developers, and everything should be clearly documented.

A particularly important area of scrutiny is input validation because it can lead to serious vulnerabilities. Essentially, we should treat every single user input as malicious until proven otherwise. For example, if we don’t put limits on how many characters users can enter when providing, say, their names on a web form, they could cause a buffer overflow, which is a classic example of a technique used to exploit improper input validation. A buffer overflow (which is described in detail in Chapter 18) takes place when too much data is accepted as input to a specific process. The process’s memory buffer can be overflowed by shoving arbitrary data into various memory segments and inserting a carefully crafted set of malicious instructions at a specific memory address.

Buffer overflows can also lead to illicit escalation of privileges. Privilege escalation is the process of exploiting a process or configuration setting to gain access to resources that would normally not be available to the process or its user. For example, an attacker can compromise a regular user account and escalate its privileges to gain administrator or even system privileges on that computer. This type of attack usually exploits the complex interactions of user processes with device drivers and the underlying operating system.

A combination of input validation and configuring the system to run with least

privilege
can help mitigate the threat of escalation of privileges.
What is important to understand is that secure coding practices need to be integrated
into the development phase of the SDLC. Security has to be addressed at each phase of
the SDLC, with this phase being one of the most critical.

Testing Phase

PART VIII

Formal and informal testing should begin as soon as possible. Unit testing is concerned
with ensuring the quality of individual code modules or classes. Mature developers
develop the unit tests for their modules before they even start coding, or at least in parallel with the coding. This approach is known as test-driven development and tends to result
in much higher-quality code with significantly fewer vulnerabilities.
Unit tests are meant to simulate a range of inputs to which the code may be exposed.
These inputs range from the mundanely expected, to the accidentally unfortunate, to
the intentionally malicious. The idea is to ensure the code always behaves in an expected
and secure manner. Once a module and its unit tests are finished, the unit tests are run
(usually in an automated framework) on that code. The goal of this type of testing is to
isolate each part of the software and show that the individual parts are correct.
Unit testing usually continues throughout the development phase. A totally different
group of people should carry out the formal testing. Depending on the methodology and
the organization, this could be a QA, testing, audit, or even red team. This is an example

Separation of Duties
Different environmental types (development, testing, and production) should be properly separated, and functionality and operations should not overlap.
Developers should not have access to modify code used in production. The code should be
tested, submitted to a library, and then sent to the production environment.

of separation of duties. A programmer should not develop, test, and release software. The
more eyes that see the code, the greater the chance that flaws will be found before the
product is released.

No cookie-cutter recipe exists for security testing because the applications and products can be so diverse in functionality and security objectives. It is important to map security risks to test cases and code. The software development team can take a linear approach by identifying a vulnerability, providing the necessary test scenario, performing the test, and reviewing the code for how it deals with such a vulnerability. At this phase, tests are conducted in an environment that should mirror the production environment to ensure the code does not work only in the labs.

Security attacks and penetration tests usually take place during the testing phase to identify any missed vulnerabilities. Functionality, performance, and penetration resistance are evaluated. All the necessary functionality required of the product should be in a checklist to ensure each function is accounted for.

Security tests should be run to test against the vulnerabilities identified earlier in the project. Buffer overflows should be attempted, interfaces should be hit with unexpected inputs, denial-of-service (DoS) situations should be tested, unusual user activity should take place, and if a system crashes, the product should react by reverting to a secure state. The product should be tested in various environments with different applications, configurations, and hardware platforms. A product may respond fine when installed on a clean Windows 10 installation on a stand-alone PC, but it may throw unexpected errors when installed on a laptop that is remotely connected to a network and has a virtual private network (VPN) client installed.

Verification vs. Validation

Verification determines if the software product accurately represents and meets the specifications. After all, a product can be developed that does not match the original specifications, so this step ensures the specifications are being properly met. It answers the question, “Did we build the product right?”

Validation determines if the software product provides the necessary solution for the intended real-world problem. In large projects, it is easy to lose sight of the overall goal. This exercise ensures that the main goal of the project is met. It answers the question, “Did we build the right product?”

↟Chapter 24: Software Development

Testing Types

Software testers on the software development team should subject the software to various types of tests to discover the variety of potential flaws. The following are some of the most common testing approaches:

- Unit testing Testing individual components in a controlled environment where programmers validate data structure, logic, and boundary conditions
- Integration testing Verifying that components work together as outlined in the design specifications
- Acceptance testing Ensuring that the code meets customer requirements
- Regression testing After a change to a system takes place, retesting to ensure functionality, performance, and protection

A well-rounded security test encompasses both manual tests and automated tests. Automated tests help locate a wide range of flaws generally associated with careless or erroneous code implementations. Some automated testing environments run specific inputs in a scripted and repeatable manner. While these tests are the bread and butter of software testing, we sometimes want to simulate random and unpredictable inputs to supplement the scripted tests.

A manual test is used to analyze aspects of the program that require human intuition and can usually be judged using computing techniques. Testers also try to locate design flaws. These include logical errors, which may enable attackers to manipulate program flow by using shrewdly crafted program sequences to access greater privileges or bypass authentication mechanisms. Manual testing involves code auditing by security-centric programmers who try to modify the logical program structure using rogue inputs and reverse-engineering techniques. Manual tests simulate the live scenarios involved in realworld attacks. Some manual testing also involves the use of social engineering to analyze the human weakness that may lead to system compromise.

At this stage, issues found in testing procedures are relayed to the development team in problem reports. The problems are fixed and programs retested. This is a continual process until everyone is satisfied that the product is ready for production. If there is a specific customer, the customer would run through a range of tests before formally accepting the product; if it is a generic product, beta testing can be carried out by various potential customers and agencies. Then the product is formally released to the market or customer.

NOTE Sometimes developers include lines of code in a product that will allow them to do a few keystrokes and get right into the application. This allows them to bypass any security and access controls so they can

quickly access the application’s core components. This is referred to as a “back door” or “maintenance hook” and must be removed before the code goes into production.

Once the software code is developed and properly tested, it is released so that it can be implemented within the intended production environment. The software development team’s role is not finished at this point. Newly discovered problems and vulnerabilities are commonly

identified at this phase. For example, if a company developed a customized application for a specific customer, the customer could run into unforeseen issues when rolling out the product within its various networked environments. Interoperability issues might come to the surface, or some configurations may break critical functionality. The developers would need to make the necessary changes to the code, retest the code, and re-release the code.

Almost every software system requires the addition of new features over time. Frequently, these have to do with changing business processes or interoperability with other systems. This highlights the need for the operations and development teams to work particularly closely during the operations and maintenance (O&M) phase. The operations team, which is typically the IT department, is responsible for ensuring the reliable operation of all production systems. The development team is responsible for any changes to the software in development systems up until the time the software goes into production. Together, the operations and development teams address the transition from development to production as well as management of the system’s configuration.

Another facet of O&M is driven by the fact that new vulnerabilities are regularly discovered. While the developers may have carried out extensive security testing, it is close to impossible to identify all the security issues at one point and time. Zero-day vulnerabilities may be identified, coding errors may be uncovered, or the integration of the software with another piece of software may uncover security issues that have to be addressed. The development team must develop patches, hotfixes, and new releases to address these items.

In all likelihood, this is where you as a CISSP will interact the most with the SDLC.

Change Management

One of the key processes on which to focus for improvement involves how we deal with the inevitable changes. These can cause a lot of havoc if not managed properly and in a deliberate manner. We already discussed change management in general in Chapter 20, but it is particularly important during the lifetime of a software development project.

The need to change software arises for several reasons. During the development phase, a customer may alter requirements and ask that certain functionalities be added, removed, or modified. In production, changes may need to happen because of other changes in the environment, new requirements of a software product or system, or newly released patches or upgrades. These changes should be carefully analyzed, approved, and properly incorporated such that they do not affect any original functionality in an adverse way.

Change management is a systematic approach to deliberately regulating the changing nature of projects, including software development projects. It is a management process that takes into account not just the technical issues but also resources (like people and money), project life cycle, and even organizational climate. Many times, the hardest part of managing change is not the change itself, but the effects it has in the organization.

Many of us have been on the receiving end of a late-afternoon phone call in which we're told to change our plans because of a change in a project on which we weren't even working. An important part of change management is controlling change.

Change Control

Change control is the process of controlling the specific changes that take place during the life cycle of a system and documenting the necessary change control activities.

Whereas change management is the project manager's responsibility as an overarching

process, change control is what developers do to ensure the software doesn't break when they change it.

Change control involves a bunch of things to consider. The change must be approved, documented, and tested. Some tests may need to be rerun to ensure the change does not affect the product's capabilities. When a programmer makes a change to source code, she should do so on the test version of the code. Under no conditions should a programmer change the code that is already in production. After making changes to the code, the programmer should test the code and then deliver the new code to the librarian.

Production code should come only from the librarian and not from a programmer or directly from a test environment.

A process for controlling changes needs to be in place at the beginning of a project so that everyone knows how to deal with changes and knows what is expected of each entity when a change request is made. Some projects have been doomed from the start because proper change control was not put into place and enforced. Many times in development, the customer and vendor agree on the design of the product, the requirements, and the specifications. The customer is then required to sign a contract confirming this is the agreement and that if they want any further modifications, they will have to pay the vendor for that extra work. If this agreement is not put into place, then the customer can continually request changes, which requires the software development team to put in the extra hours to provide these changes, the result of which is that the vendor loses money, the product does not meet its completion deadline, and scope creep occurs.

Other reasons exist to have change control in place. These reasons deal with organizational policies, standard procedures, and expected results. If a software product is in the last phase of development and a change request comes in, the development team should know how to deal with it. Usually, the team leader must tell the project manager how much extra time will be required to complete the project if this change is incorporated and what steps need to be taken to ensure this change does not affect other components within the product. If these processes are not controlled, one part of a development team could implement the change without another part of the team being aware of it. This could break some of the other development team's software pieces. When the pieces of the product are integrated and some pieces turn out to be incompatible, some jobs may