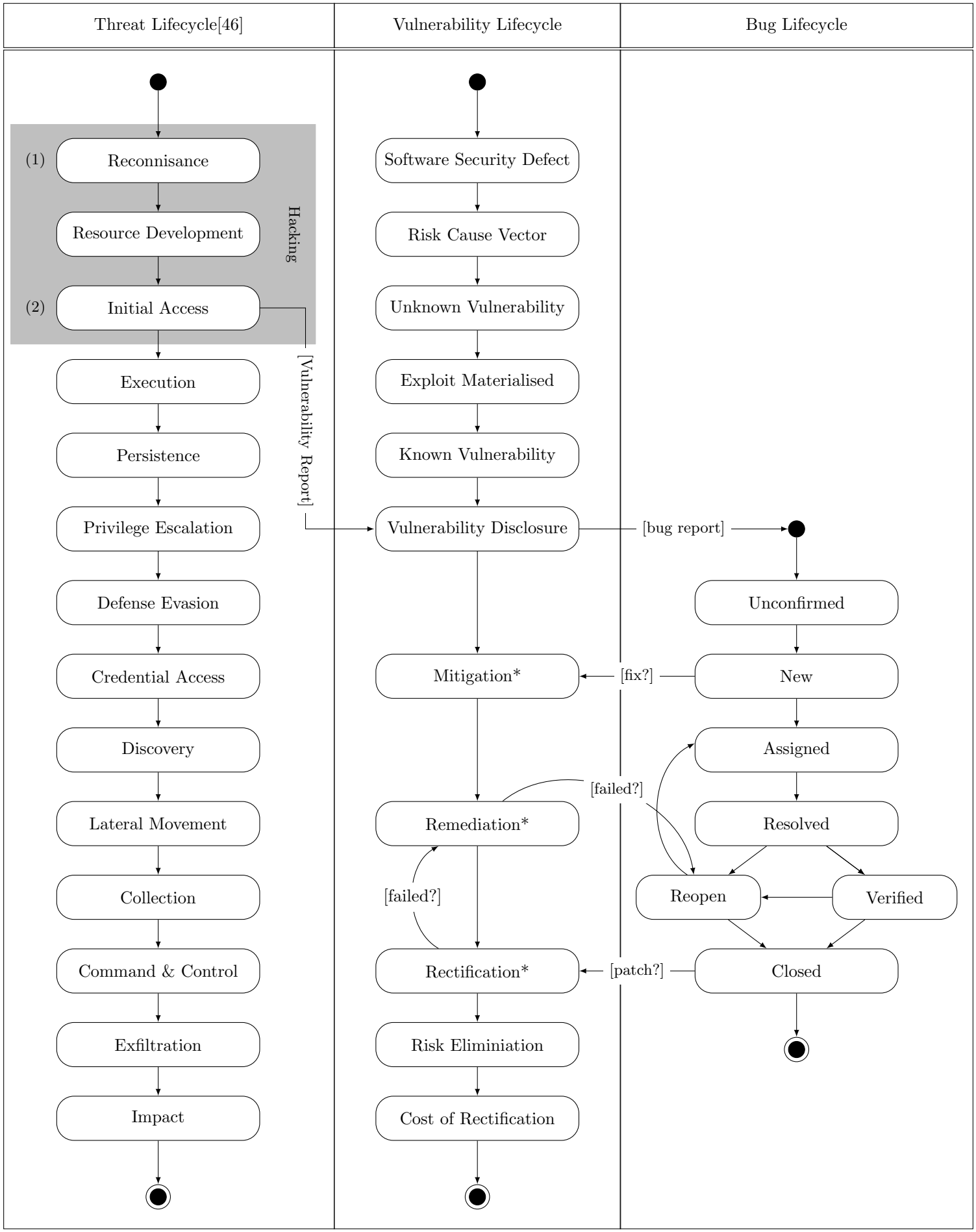

(1) Point of first network contact (2) Initial Attack Vector * Possible patch/hotfix release

Figure 14: The lifecycle of a vulnerability

Vulnerability management

The lifecycle of a vulnerability

Referring to Figure 14 on Page 72, in building a secure system, it is important to differentiate the relationship between a cyber threat, a vulnerability and a software bug. It is also a mistake to assume that bugs, vulnerabilities or threats are well defined. This is because it is impossible to prove that a system is safe from threats, vulnerabilities or bugs, however it is possible to clearly deliniate the difference between each so their risk can be reduced and risk planning can be put in place to ensure early detection and appropriate response.

1. **Threat**: A *threat* is a potential occurence of a security event that can have an undesirable effect on the system assets or resources. It is a danger that can lead to undersirable consequences.

2. **Vulnerability**: A *vulnerability* is a weakness that makes it possible for a threat to occur.

3. **Bug**: A *bug* is a software defect and/or fault that has been identified and has been entered into the bug tracking system so that it can be eliminated, resolved and closed.

When working with Threats, otherwise known as *Threat Intelligence* or *Threat Hunting*, one must recognise that the true source of a threat and the ability to close down it's entry point may only be done by reviewing logs and other forensic data to isolate the *Point of first network contact* and the *Initial Attack Vector*. One can only be sure they have identified these critical landmarks in the threat lifecycle if they have also identified hacking activity between these two points (see (1) and (2) respectively in Figure 14). Hacking activity may refer to scanning, password guessing, and/or some signs that a vulnerability has been used as part of the intial attack vector. Once the vulnerability associated with the initial attack vector has been identified, one must investigate and decide if this is a new vulnerability that must be reported to a CERT or to the vendor directly, or if it relates to an existing vuilnerability that has not yet been patched.

Vulnerabilities progress through a well defined lifecycle that begins with their realization at the implementation stage of the functionality lifecycle (see Figure 7 on Page 19 and Figure 14 on Page 72):

1. **Security Defect created:** Software *security defect* realized and instantiated in development environment.

2. **Risk Cause Vector introduced:** *security defect* gets committed to a release canidate, causing a *risk cause vector* to be introduced to SDLC.

3. **Unknown Vulnerability materialised:** *risk cause vector* gets deployed to production and a known, suspected or unknown minus-days *vulnerability* is now in production. May go undetected for many years.

4. **Exploit Materialised:** First successful *exploit* script written. This marks Day 0 (A script can be code, or a written sequence of steps).

5. **Known Vulnerability discovery:** Discovery of a vulnerability by a 3rd party.

6. **Vulnerability Disclosure:** Disclosure of a now known vulnerability to the software/hardware vendor.

7. **Mitigation:** Emergency *mitigation**

8. **Remediation:** *remediation* attempts* - this stage may be repeated numerous times, as the root cause of a security defect is not always immediately apparent.

9. **Rectification:** Final root cause identified and ultimate *rectification**

10. **Risk Elimination:** Security defect and risk cause vector *eliminated* - finalization activities commence (eg, releasing an advisory, making announcements to markets, produce reports for government bodies, notify Individuals of PII dta loss).

11. **Cost of Impact & Rectification:** Final *risk event cost* reported to board as *cost of impact* and *cost of rectification*.

**Can involve a patch release.*

Threats on the other hand progress through a lifecycle that begins with a class of hacking known as network *Reconnisance*. We have used here the Mitre ATT&CK framework [46] as it is the most complete and has been peer reviewed over a significant period. Referring again to Figure 14 on Page 72:

1. **Reconnisance (TA0043):** The adversary is trying to gather information they can use to plan future operations.

2. **Resource Development (TA0042):** The adversary is trying to establish resources they can use to support operations.

3. **Initial Access (TA0001):** The adversary is trying to get into your network.
4. **Execution (TA0002):** The adversary is trying to run malicious code.
5. **Persistence (TA0003):** The adversary is trying to maintain their foothold.
6. **Privilege Escalation (TA0004):** The adversary is trying to gain higher-level permissions.
7. **Defense Evasion (TA0005):** The adversary is trying to avoid being detected.
8. **Credential Access (TA0006):** The adversary is trying to steal account names and passwords.
9. **Discovery (TA0007):** The adversary is trying to figure out your environment.
10. **Lateral Movement (TA0008):** The adversary is trying to move through your environment.
11. **Collection (TA0009):** The adversary is trying to gather data of interest to their goal.
12. **Command and Control (TA0011):** The adversary is trying to communicate with compromised systems to control them.
13. **Exfiltration (TA0010):** The adversary is trying to steal data.
14. **Impact (TA0040):** The adversary is trying to manipulate, interrupt, or destroy your systems and data.

More information about these stages of the Threat Lifecycle are available at the MITRE ATT&CK website (`https://attack.mitre.org/tactics/enterprise/`). It is important that the *Reconnissance*, *Resource Development* and *Initial Access* stages be identified as early as possible so that one can identify the point of first network contact, the associated 'hacking' and the intial attack vector so that the hole can be identified and closed. If you cannot find these things, you either haven't found the initial attack vector, or a vendor backdoor has been used by a state actor to access your systems.

Regardless of if you are dealing with a *threat* or a *vulnerability*, it is important at the first available opportunity that one estimates the cost of impact and the cost of rectification respectively. Once the process of eliminating a *threa* or *vulnerability* has concluded, an updated cost should be calculated and presented to the board or whomever else is ultimately responsible for signing off on these things.

When a vulnerability report occurs, it is important that the initial discloser, at least at a top level, be taken along for the ride. That is because they have a unique view of the vulnerability, and it cannot truly be marked as rectified until they agree that the vulnerability has been eliminated. Very little information needs to be provided other than one's acceptance to address a vulnerability, an update email estimating when it will be rectified, and update emails each time a remediation attempt is made. This is because most vulnerabilties relate to hidden functionality that does not form the function points of a system, and their review is required to be absolutely sure that the *Risk Cause Vector* has been eliminated.

Vulnerabilities are complex to fix, because they are hidden behaviour that is not part of the system's prescribed function points and the behaviour is usually hidden behind legitimate code that is intended to deliver other function points simultaneously. It is extrordinarilty difficult to change existing code so as to eliminate the abberant *vuilnerability* behaviour without changing the system's underlying function points. Because of this, it is not uncommon for many attempts to be made to remediate a vulnerability before it can finally be considered rectified. It is for this reason that the lifecycle of a vulnerability includes three separate steps to address a vulnerability: mitigation is an emergency undertaking designed to reduce or temporarily eliminate the severity of a vulnerability, sometimes at the cost of prescribed system functionality. A vulnerability then goes through several remediation attempts before it finally can be declare rectified and risk elimination steps such as estimating the final cost of rectification can be undertaken.

Note that vulnerabilities do not get 'triaged', test teams triage the underlying security defect. Triaging is a test team activity that is initiated at Vulnerability Disclosure, when a vulnerability is finally discovered and isolation of the associated security defect commences (see Test team role in handling security defects on Page 73).

There is also the matter of crisis management and incident response, which is another parallel process that is initiated at Vulnerability Disclosure. The three interrelated processes are illustrated in Figure Figure 14 on Page 72.

