4.1 Discovery

Reports that say that something hasn’t happened are always interesting to me, because as we know, there are known knowns; there are things we know
we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns
– the ones we don’t know we don’t know. – Donald Rumsfeld

Aside from the simplest applications, software development is difficult, complex, and prone to error. As a result, the likelihood that any given software-
based product or component is free of vulnerabilities is extremely low. For vendors, this implies the need to create a response capability to handle
vulnerability reports, whether those reports come from sources internal or external to the vendor.

Why Look for Vulnerabilities?

Ultimately, we can’t fix vulnerabilities we don’t know about. While software engineering best-practices, code audits, testing (including fuzzing), and
application security testing are important parts of the development lifecycle, security research is important for rooting out hidden vulnerabilities. Some
organizations may have in-house expertise to find and identify security vulnerabilities, but most will not. For some vendors, encouraging independent
vulnerability finders may be the only way to stay on top of the latest trends in vulnerability research and exploitation techniques.

Many organizations hire application security testers or code auditors to look for vulnerabilities. While such testing is certainly important and commendable,
it is important to understand that absence of evidence is not always evidence of absence. Rumsfeld’s point about unknown unknowns applies here. A
clean audit or pen test report should not be taken as evidence that the software is free of vulnerabilities. All software-based systems have problems we’re
not even aware of and so we don’t even know to look for them. Because such vulnerabilities may exist and can be exploited without warning, vendors and
deployers should establish their VR capability in preparation for this eventuality.

Avoid Unnecessary Risk in Finding Vulnerabilities

Finders should exercise an appropriate degree of care when performing vulnerability research. This will help to alleviate legal concerns and limit the
potential for damage to others. Vulnerability research should of course be performed on equipment that the finder is authorized to use for the purpose. If the research is performed on
behalf of an organization such as a private security firm or university, permission should be obtained before attempting research on organization-owned
equipment.

Likewise, organizations should make the rules and process for obtaining permission very clear and easy to find. For example, a form or email address
provided on an intranet page might be sufficient. Employees hired specifically to find vulnerabilities should be briefed on necessary rules and provided with
concrete permission as part of the on-boarding process. Failure to adequately document permissible scope and authority for vulnerability testing can lead
to frustration and other negative consequences with various legal ramifications.

Operational Risk

In general, the software or devices tested should not be production systems that support or have access to real data or users. When possible, dedicated,
controlled testing environments should be established. Such a testing environment often consists of virtual machines (VMs) in a virtual network firewalled
off from any production network. Even as a finder in a controlled testing scenario, you should keep in mind the potential for unintended consequences (i.e.,
the unknown unknowns). Always try to limit the potential for unintended negative impact of testing, even within your controlled environment. If the impact
cannot be constrained to a controlled environment with relatively known consequences, do not attempt to test your exploit and instead report your findings
directly to the vendor.

Safety Risk

Safety-critical systems have been defined as ”systems whose failure could result in loss of life, significant property damage, or damage to the environment
[1].” A high degree of caution is both appropriate and necessary when testing the security of safety-critical systems, such as medical devices, industrial
equipment, or vehicles. A proof of concept exploit to demonstrate a vulnerability on a traditional computer might cause a calculator to pop up on the
screen. A proof of concept exploit on a car might cause it to behave erratically, potentially leading to injury or death. Testing or demonstrating safety-critical
systems outside a controlled environment, or when there is any chance of harming unwitting bystanders is unacceptable under any circumstances.

Legal Risk
Depending on the circumstances, finders may be subject to a non-disclosure agreement (NDA) regarding any vulnerabilities found. This is often the case when vulnerability testing is performed on behalf the vendor whether directly as an employee, or under contract as part of a consulting firm or as a freelance consultant. Finders should be aware of this possibility and consider the legal implications of any relevant NDAs before reporting a vulnerability to any third party.

That said, vendors are strongly encouraged to avoid requiring NDAs of reporters if at all possible. Many finders prefer to avoid the legal entanglements that NDAs entail and will be discouraged from reporting vulnerabilities when an NDA is involved. This can leave vendors unaware of potential threats to their products and services and in turn, their users.

Additionally, in some environments, such as medical devices, healthcare, education, or financial information systems, there may be legal consequences to accessing real data (under HIPAA [2], FERPA [3], COPPA [4], and similar laws, industry standards such as PCI DSS [5], etc.), so we again reiterate the need to perform research only in controlled test environments, preferably with fake data.

For more information on the legal implications of vulnerability disclosure, we refer you to the EFF's Coders' Rights Project Vulnerability Reporting FAQ [6].

References