Need a deeper dive on this topic? **Ask-An-Expert**

# Decide Where to Do Manual Penetration-Testing: Production or Dev/Test

Ask-An-Expert Writeups | Penetration Testing and Red Teaming | By Jake Williams, IANS Faculty

## What You Will Learn

- How the decision to do manual pen-testing on production vs. dev/test environments depends on many factors, including the types of vulnerabilities tested, network architecture, stability of applications/underlying OS, similarity between the two environments and potential business impact.
- There is no one-size-fits-all answer. While all penetration-testing should be performed against production systems in an ideal world, the realities of business impacts dictate that some testing must be performed against development systems, with the results extrapolated to their production counterparts.

## The Challenge: Performing Manual Penetration Tests

The security team for a telecommunications company is interested in learning best practices around conducting manual penetration tests in production environments vs. development/test environments. Specifically, the team asks:

- Can IANS provide some recommendations?

## Production or Development? No Easy Answer

When performing penetration-testing, business units are often concerned about critical assets being impacted by the testing. For this reason, a case is often made that penetration testers should only be allowed to access development and/or test systems. While this strategy satisfies the goal of not impacting production systems, it carries its own separate set of risks.

The primary risk of only performing penetration-testing against development systems is they may

be configured differently than their counterparts in production. Two possible scenarios emerge. First, the penetration tester might discover vulnerabilities in the development system that are not present in the production system (perhaps due to a compensating control being deployed in production). Second, the production system might have a vulnerability that is not present in the development system.

The first scenario is a nuisance, while the second scenario can be catastrophic – particularly because of the false sense of security it promotes. If an organization believes a system is secure when in fact it is not, the security staff can end up making poor judgments in terms of deploying controls and investigating alerts.

## Factors to Consider

The decision of whether to perform most penetration-testing on production or development systems depends on a number of factors:

- How experienced are the penetration testers on your team?
- How confident are you the development system matches the configuration and software versions of the production system?
- How tightly managed is the change control process on the production systems?
- Are production systems used to control or monitor life safety equipment or perform other life safety-related functions?
- What is the maximum anticipated business impact if a production system is taken offline during testing?
- How likely is it the test being performed could adversely impact system operation and/or stability?
- How robust is your disaster recovery (DR) program?
- Are you confident you could quickly restore a business-critical system that was adversely impacted by testing?

While a full discussion of each of these points is beyond our scope, we will delve into a few of them in more detail.

## Will Tests Likely Affect System Operation/Stability?

Not all penetration tests are created equal. Some tests are more likely than others to cause issues with system availability and data integrity. For instance:

- **Denial-of-service (DoS) vulnerabilities.** In some cases, the only way to determine a DoS vulnerability is present via a remote scan is to attempt exploitation. Even if the expected outcome is remote code execution, any number of factors the penetration tester cannot account for before attempting the exploit can lead to an application crash.
- **SQL injection.** This is another class of vulnerability that can be problematic to test in production. Many organizations request penetration testers leave behind proof they were able to access the system. In most cases, the penetration tester simply plants a flag (commonly a text file). With SQL injection, however, dropping a text file is most often not possible. Consequently, penetration testers may be asked to illicitly insert new data or modify existing data in the database.

With total knowledge of the underlying system, this can be performed safely. However, penetration testers often lack perfect knowledge of the system and these updates can have catastrophic impacts. Even leaking data via SQL injection can cause issues in production systems. A very common class of SQL injection vulnerability is blind SQL injection. The penetration tester can only ask binary (yes/no) questions of the database. This results in a very large number of queries relative to the amount of data being leaked. These queries also often involve joins and unions, which can further impact database performance.

## Can the Network Architecture Handle Testing Traffic?

Another issue for consideration is the network infrastructure where the development and production systems are deployed. For instance, a port scan is very unlikely to cause an adverse system impact on most endpoints. This is equally true of networking equipment. If only a single host is being scanned, we should not expect any impairment of network equipment.

However, if a penetration tester attempts to scan all TCP ports across thousands of systems, that amount of traffic could easily overwhelm the capacity of a small office/home office (SOHO) network switch. Likewise, a penetration tester performing scans over a site-to-site virtual private network (VPN) could easily overwhelm the capacity of a smaller VPN concentrator.

Neither of these are hypothetical; we have observed both situations in corporate environments. Neither of these, however, are likely when testing a development system. First, the development system has fixed IP address ranges, so penetration testers are less likely to be bulk scanning. Second, even if network infrastructure for the development system is overwhelmed, the impact should be limited in properly architected networks.

## How Stable Are the Applications and Underlying OS?

Another factor at play is the stability of the underlying applications and operating systems. Some poorly architected devices can actually encounter issues from just a port scan (this is particularly common with older embedded devices).

However, instability of the underlying application should not necessarily be used to justify not testing production systems. IF attackers gain access to the network, they will not have the same scoping limitations. The organization can be better prepared by understanding these potential impacts during a more controlled test.

## How Close Are Production and Development Environments?

Now, let's consider the likelihood development systems match the exact configurations deployed in production. In traditional development models, this is extremely unlikely. Systems administrators dealing with issues in production often make configuration changes during troubleshooting that are not reverted once the problem is resolved. This leads to a drift between the effective configuration and the documented configuration.

In development shops using containers and microservices for deployment, however, it is much more likely the deployed and the development configurations will be identical, since very few (if any) configuration changes need to be made. Rather than installing the application on an existing OS, the deployment involves merely cloning a container with all necessary configuration files, libraries and other dependencies. Applications using this model are prime candidates for effective penetration-testing in the development environment.

# What's the Potential Business Impact?

Finally, we should consider the state of the organization's DR program, as well as the maximum impact of a production system outage. Although these are two separate issues, they are closely related. Anytime a system is unavailable, there is obviously some production impact.

But the duration of that impact is tightly related to the state of the organization's DR program. Even in mature DR programs, not all systems are created equal. Some systems and services can be restored to operation simply by reverting to the latest known good backup. In other cases, restoring from the latest backup would result in an unacceptable loss of transactional data (data that must be rebuilt or recovered from other sources).

***Avoid Transactional Data Systems***

We strongly advise against fuzzing or performing attacks that pose a risk of data corruption on systems that use transactional data. A corrupted record in a single table can have cascading effects that may only be discovered weeks or months after the test concludes.

# Testing Should Vary System by System

In an ideal world, all penetration-testing would be performed against production systems. However, the realities of business impacts dictate that some testing must be performed against development systems, with the results extrapolated to their production counterparts.

You should consider the factors above when determining whether testing development systems is more appropriate than testing production, but realize you are unlikely to come to a single answer for this question. Instead, the answer will vary system to system because the risks for each are different.

# Further Reading

I Am Not a Robot: Manual Pen-Testing Tips and Tricks, April 13, 2018

Penetration Test Preparation Checklist, May 7, 2015

Adversarial Simulations - Evolving Penetration Testing, April 18, 2017

Top 10 Ways Penetration Testers Break Into Organizations, Aug. 22, 2017