


+



INTO THE HACKER'S MIND

Learn how pentesters assess web applications and APIs, and how to best prepare for your next pentest





Why Pentesting? Framing the Assessment



Why Pentesting?

Framing the Assessment

What does it mean to be a “hacker” in today’s environment? I would start by asking a follow-up question: “*What kind?*”

A hacker’s mindset can be applied in nearly any cybersecurity job. In addition to becoming full-time penetration testers or offensive security engineers, those with the skills to attack applications can leverage their talents in security operations, forensics, vulnerability management, or any flavor of security engineering. They may choose to specialize in attacking a certain type of infrastructure—whether that’s ICS, mobile, cloud, physical offices, web applications. And they may specialize in the *method* of attack—from social engineering to exploit development to cloud configuration reviews.

Whenever something contains sensitive data, a hacker’s mindset should always come into play.

This white paper will help you understand how pentesters think, how they look at your application, and the steps behind an attack. With these insights, you can better prepare for an upcoming pentest engagement, making it a much more focused and insightful view into the security of your web applications and their integrated infrastructure.

Bug Bounty versus Pentesting

Both are synonymous with hacking, so how do we distinguish between the two in this white paper?

An established bug bounty program lends itself to *depth*. Security researchers are rewarded for the severity of the bugs they find. In a well-established and competitive bug bounty program, it is advantageous for security researchers to find complex vulnerabilities as quickly as possible. This, in turn, is advantageous for companies who want to reduce risk as soon as it is presented.

Bug bounty programs can serve SaaS companies that continuously push code because programs provide ongoing coverage of a constantly changing attack surface. (Although this comes with nuance, as the more a program ages, the more you will need to expand the pool of researchers and reward.)

Bug bounty programs may also come with a litany of “vulnerability debris” – common misconfigurations that gain popularity through hacking collateral and Twitter. Think of the insurgence of cross-site scripting, subdomain takeovers, or DMARC reports. Certain classes of vulnerabilities may become trendy and easily identifiable through automation, which results in quick finds with reasonable payouts. Although these vulnerabilities present some risk, the value of these reports diminish over time and will quickly find themselves on the “out of scope” list for beleaguered bug bounty programs.

Thus, a hacker's mindset for bug bounty rewards either quick, repeatable vulnerabilities discovered through automation, or deep, complex vulnerabilities that may pay a sizable award. It's a bit like gambling for both security researchers and bug bounty programs.

On the other hand, a pentester's mindset requires *coverage* and *context*. Pentests serve a crucial function for businesses that must meet compliance, legal requirements, and contractual agreements. Decades of iterative research and collaboration among security researchers has contributed to concrete methodologies for attacking applications for pentests. [Cobalt Core](#) pentests, for example, follow the [OWASP methodology](#). When pentesters follow these methodologies appropriately, pentest engagements guarantee due diligence for both pentesters and companies.

Since pentests are time-boxed and serve a discrete function for companies, pentesting lends itself to a more collaborative approach with companies and with other pentesters. Both pentesters and the companies hiring them are working together to identify risk that is unique to the business.

This collaboration begins with proper scoping and an exceptional brief.

+



Reviewing the Application



Reviewing the Application

It starts with the brief.

Think of a brief as a pentester's map. All maps require a general framing of the environment, though the exact borders may be unknown. Then come the details—where the roads are, the mountains, the rivers. It provides a sense of navigation and direction.

It takes time and nuance for a pentester to understand the depths of an attack surface. A well-laid out brief reduces the amount of reconnaissance and gives more time for thoroughly attacking the application. It should also provide critical information about user roles and permissions, business functions, and areas of focus. A pentester cannot test access controls and business logic unless both the company and the pentesters know what each user role is supposed to do.

Black box testing: Tempting but unproductive.

It may be tempting to run black-box testing against an application, in which limited or no information is provided to pentesters. Black-box testing essentially challenges pentesters to “find what they can” and exploit it, which means pentesters must rely on heavy-duty reconnaissance to attack the application. This approach is limited for a number of reasons:

1.

Black-box testing will always be part of a full pentest engagement. Pentesters conduct intensive reconnaissance and will test for enumeration, authentication, and privilege escalation, despite what information is provided in the brief. Anything exposed on the Internet that could be used to attack the application in-scope will be included as part of the pentesting methodology.

2.

Black-box testing ignores some of the most pressing risks for companies in 2022: insider threat, abuse of exposed credentials, and inadvertent human error. See the [Verizon 2022 DBIR](#) for more information.

3.

Black-box testing forgoes a critical function of pentesting: in-depth reviews of business logic and access control. In essence, a bug bounty program can function as a continuous black-box test when only an in-scope domain is included. However, a pentest gives companies the opportunity to convey more sensitive information about the application and its functions, which is covered under an NDA. Within this context, you can provide architecture diagrams, tech stack information, and credentials that can be leveraged to attack more sensitive parts of the infrastructure.

White-box or gray-box testing gives you an opportunity to capture risk for a wider attack surface, which is reflected in the scope.

There is no pentest without scope.

When beginning an engagement, the brief provides critical context. Pentesters always begin by assessing the scope. The scope listed in a brief serves as a contractual obligation, a compliance requirement, and a legal agreement. If you're a small business that's procuring a pentest when specifically requested by a prospect, you are providing evidence that the infrastructure containing that sensitive data has been tested and is secure. A scope is also a written agreement between the company and the pentesters that the pentesters will only attack what is listed—and nothing more.

This is why it's essential to dive into the infrastructure before listing the scope. Even if you will not be providing architecture information to pentesters, it is well advised to consult with your engineering teams to ensure you are adequately documenting the assets that should be tested.

In addition, pay attention to the nuances of how you list scope, which will impact what pentesters can review.

It's rarely just a web app.

An Internet-facing web application is always hosted on a server. For SaaS companies in particular, these servers are typically hosted in the cloud, like AWS. The web application may likely leverage functions that require an API or web hooks. There may also be an associated mobile application.

Consider a web application for the company Squirrels, Inc. that is hosted on AWS and utilizes a simple API. Here's what a pentester will cover depending on what's listed in scope:

Listed in Scope	What This Includes	How the Pentester Will Investigate
squirrels.com	squirrels.com/*	<ul style="list-style-type: none"> • Crawling for endpoints
.squirrels.com	squirrels.com/ *.squirrels.com/*	<ul style="list-style-type: none"> • Crawling for endpoints • Subdomain enumeration
.squirrels.com api.squirrels.com	squirrels.com/ *.squirrels.com/* api.squirrels.com	<ul style="list-style-type: none"> • Crawling for endpoints • Subdomain enumeration • API reconnaissance
.squirrels.com api.squirrels.com External network	squirrels.com/ *.squirrels.com/* Api.squirrels.com External network	<ul style="list-style-type: none"> • Crawling for endpoints • Subdomain enumeration • API reconnaissance • Nmap scans and cloud configuration review



It's more than just an asterisk.

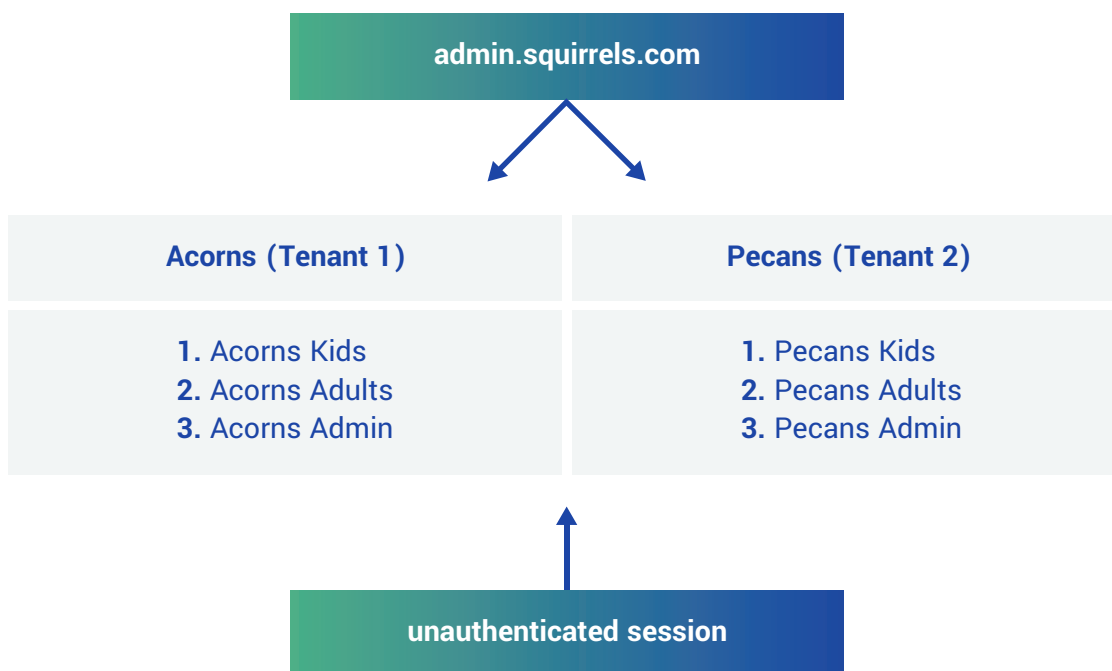
A web application can either be a single tenant (everyone accesses the same environment) or multi-tenant (there are different environments and users can only access certain environments with certain restricted data).

Suppose that Squirrels Inc. has two customers: Acorns and Pecans. They both log into the same website portal: `signin.squirrels.com`, but when the session starts, they see different environments and different data.

Pentesters want to ensure that Acorns users cannot see the data for Pecans, and vice versa. That's why a pentest brief must include credentials for both Acorns and Pecans users.

But it gets deeper than that. Suppose that Squirrels Inc. is a SaaS environment with three user roles for each customer: Squirrels Kids, Squirrels Adults, and Squirrels Admins. Each user has different functionality. Squirrels Kids shouldn't view information that's for Squirrels Adults, and Squirrels Kids and Squirrels Adults shouldn't have the same permission as Squirrels Admins. They also shouldn't have access to any user functions for any user role for other tenants. And then on top of that, an unauthenticated user shouldn't have access to anything that is behind authentication.

So a pentest brief for a multi-tenant environment for Squirrels Inc needs to include at least six credentials. Of course, there is always a seventh admin—that of the admin for the SaaS platform. If you are using a staging environment, it could be worthwhile to also provide credentials for the admin account. This type of highly-sensitive credential should not be provided for a production environment since it exposes restricted customer data.



Put together, multiple credentials and an explanation of user roles and permissions allows a pentester to review the following types of vulnerabilities in depth:

- IDOR
- Privilege Escalation
- Mass Assignment
- Session Management
- Directory Traversal
- OAuth Weaknesses

User roles that are outlined in a brief can be incorporated into a Burp Suite extension like [Auth Matrix](#), which allows a pentester to automate authorization tests when crawling through the application.

But what about the API?

Let's say that the Squirrels, Inc. platform has an API that allows its customers to connect with their systems. For a pentest engagement, simply listing `api.squirrels.com` is not enough. To ensure complete coverage of the API, a proper brief should include the full API endpoints that can be imported into Postman. Otherwise, testing the API is like shooting an arrow in the dark. A pentester may encounter some API endpoints while reviewing the web application, but the full functionality is almost impossible to enumerate.

An API shares some similar vulnerabilities with its web application, like injection attacks and issues with authorization. But there are also [vulnerabilities](#) that are unique to APIs. Issues with JWT or mass assignment, for example, would need to be covered more in depth.

Why should I care about the external network?

When the external network is included in scope, it gives the pentester a greater opportunity to explore the cloud configurations that serve the web application and the API. Exposed token keys in GitHub, for example, can be tested against the infrastructure that's exposed to the Internet. Similarly, a misconfigured S3 bucket may serve not only images to the web application, but sensitive data that can be pulled by simply having an AWS account. Including an external network in the scope gives pentesters a chance to run a deep dive NMAP scan against the target and check for cloud misconfigurations that could leak information.

Proper scope and a well-written brief is half the battle.

When talking about pentest coverage, it's critical to recognize time. Pentest engagements are time-boxed, which means time is limited when covering the full methodology. Some vulnerabilities may apply to the entire application, such as vulnerable Javascript dependencies that expose risk of cross-site scripting, or a lack of parameter filtering that allows for SQL injection across the application. However, some vulnerabilities are unique to specific endpoints. The more users, endpoints, and parameters there are, the more time it takes to investigate the application. In other words, the depth of coverage will be vastly different if it's one pentester tackling one thousand endpoints than three pentesters.

Wrapping it up

As you can see, a pentest requires much more than just listing a website. Spending the time to fully prepare for a pentest before it begins will support your organization in diving deeper into areas of risk. It also makes the engagement much more motivating and challenging for pentesters.





Attacking the Application

Attacking the Application

Setting up the environment.

Identifying the scope and writing the brief is critical to launching a pentest, but you would be surprised how many people forget the final step: ensuring there's something to test.

Many pentests are run on staging environments that are separate from the production instance, which contains sensitive or restricted data. These staging environments may be used specifically for pentesting, or may be borrowed from developers. You need to prepare the scope by addressing some important questions:

1. Is this environment ready for a large amount of traffic?
2. Is this environment accessible via any IP or is it restricted by certain IPs? If restricted, have the pentester IPs been whitelisted?
3. Is the staging environment actually working?
Have you tried accessing it?
4. Do all the pentester credentials work?
5. Have you informed your incident monitoring team that a pentest is about to start?
6. Will there be someone available to unblock pentesters if they are locked out of accounts or the IP has been blocked?

Once the clock starts ticking for a pentest engagement, it cannot be paused. Delays in testing because an environment is not loading or credentials aren't working is a lose-lose for you and the pentesters. Take the time to ensure that everything is functioning properly.

Expect high traffic, then targeted attacks.

The first few days of a pentest engagement pushes a high volume of traffic to the application and network. Pentesters will crawl the application, conduct enumeration, and run active scans to identify parameters for injection and file upload, as well as automated vulnerabilities like HTTP misconfigurations, server-side request forgeries and SQL injection opportunities. This volume of traffic could potentially take down a staging environment that doesn't have proper load balancing, so it's essential to ensure that the environment for testing is prepared for the volume and type of traffic. Most malicious traffic that a pentester sends through early scans will be filtered through a proper WAF, and the remaining reported vulnerabilities will be triaged and further investigated later in the pentest.



Open-source resources

Most manual testing requires Burp Suite, which comes in either the free Community edition or the paid Professional version. However, there are free tools that most pentesters will use during an engagement:



[SSL Labs](#) (from Qualys)



[CSP Evaluator](#) (from Google)



[Security Headers](#)



[DMARC Check](#) (from Mimecast)



[Clickjacker.io](#) POC



[SQLMap](#), run through command line or as a Burp Extension



[NMAP](#), an extensive network scanner that can run superficial port scans all the way to identifying exploitable CVEs

Manual attacks

Armed with a proper scope, a detailed brief, and a strong Internet connection, a pentester during an engagement can be unstoppable. The funnest attacks for a pentester are the ones that are specific to the company's business logic and tech stack. This is where Burp Suite becomes essential to manage crafted payloads and manipulate server responses. Manual attacks could vary from complex injection attacks to access control manipulations to file upload vulnerabilities. It all depends on how the application functions and its use cases.

Staying abreast of developments

It seems that every week there is another serious vulnerability that targets tech companies. Whether it's Log4j or the recent SSL vulnerability, a pentester must always stay abreast of the latest changes in CVEs and tech stacks. (This is the reason, after all, that the OWASP Top 10 is updated every few years.) Pentesters have a variety of resources for reviewing new vulnerabilities and testing their skills. These resources are also useful for any security engineer who wants to stay updated on the upcoming threats against their attack surface:

- PortSwigger
- Twitter
- HacktheBox
- TryHackMe
- Medium





Reporting Vulnerabilities

Reporting Vulnerabilities

All applications have vulnerabilities. This is an inevitable part of shipping code and running services. Some companies fear finding high-severity vulnerabilities because of a concern for reputation loss. I argue that companies should seek these vulnerabilities out instead—before bad actors find them.

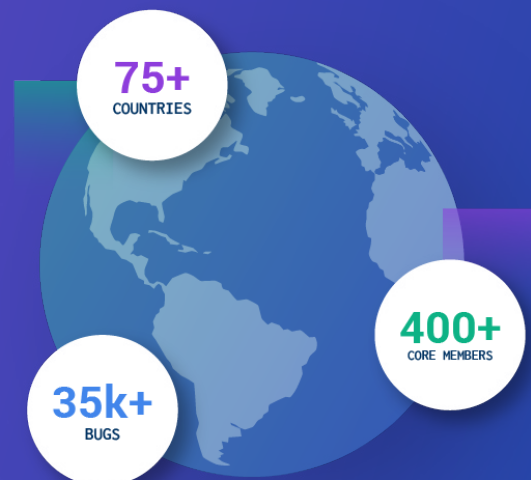
When delivering a pentest report to a prospect or auditor, the number and severity rating of vulnerability ratings are less important than the status of remediation. In fact, complex vulnerabilities may provide assurance that proper due diligence was conducted on the services.

A pentest report is a signal of trust between you and the people who use your services. You are assuring them that they are not passing sensitive data to a company that could lose or abuse it. I encourage you to take this assurance of trust seriously, while also having fun. Pentesters are here to help you and share the same goals as you: keeping All Things Secure. Enjoy the ride.



Your professional pentester community

Cobalt Core is a community of skilled, vetted pentesting professionals ready to work with you.



“Pendo is a complicated product. It takes time to wrap your mind around how it works. But the quality of the results we got from Cobalt was greater than what I had seen in comparable pentests. I felt like they were digging deep, and that’s not something I’ve always seen in the past. Where previously I might have expected two consultants to be assigned to a project, Cobalt brought five pentesters, each with different skills that complemented each other.”

| Chuck Kesler | CISO at [pendo](#)

Cobalt Core Stats

- 400+ highly vetted, certified pentesters
- 8.5 years of security experience on average

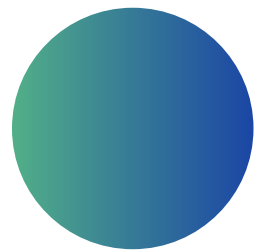
Certifications include:

OSCP, PWSP, CERP, CEH, CPISI, ISO27001L, CISSP, eWPT, MCSA

[SCHEDULE A DEMO](#)

[LEARN ABOUT THE COBALT CORE](#)





Vanessa Sauter

Cobalt Core Pentester

Vanessa Sauter is a cybersecurity professional who specializes in advising companies on security solutions architecture, product security, and risk assessments. She has years of penetration testing and vulnerability management experience, and has spoken about vulnerability management at BSides SF, BSides Athens, and DeveloperWeek. She is currently a security engineer at a leading tech start-up based in San Francisco. Previously, Sauter worked at Bugcrowd, where she led security solutions architecture. She holds a bachelor's degree from Columbia University.