Security maturity, complexity, and bug bounty program effectiveness: A deep dive

BY ELEANOR BARLOW · JUNE 10, 2025 · LAST UPDATED ON JANUARY 2, 2026

What you will learn

- **How security maturity, attack surface size, and asset complexity influence the effectiveness of your bug bounty program**, helping you prioritise testing and attract the right researchers.
- **Why a complete understanding of your organisation's assets and attack surface matters**, and how poor visibility and asset sprawl can weaken your bug bounty results and risk management.
- **How to use reconnaissance, complexity mapping, and tailored reward structures** to reduce asset sprawl, improve researcher engagement, and align payouts with effort and risk

There are three key elements that, when combined, support the planning of a bug bounty program to attract the right researchers. These three components are the attack surface, security maturity, and asset complexity. In this article, we explore each of these elements, how they impact one another, and their influence on bug bounty programs.

What defines an attack surface? And how does attack surface size impact bug bounty hunting?

The **attack surface** refers to the sum of any potential entry points within a system, network, or organization that could be exploited to gain unauthorized access. It is, in essence, the perimeter of what could be targeted. Usually, the larger the attack surface, the more variation, the more there is for a bug bounty hunter to hunt on.

Every company with a digital footprint holds assets that need to be itemized and monitored. These include anything from mobile and web applications, APIs, source code repositories, to third-party services, cloud infrastructure, and much more. For every asset, there are attributes such as integration points, data sensitivity levels, exposure levels, tech stacks, and authentication methods that need to be mapped and analyzed. This is to ensure that the attack surface is well understood, in order to manage the risk of a cyber attack effectively.

What is security maturity?

Asset **security maturity** refers to the ability to manage and respond to security threats and vulnerabilities. Typically, the higher the security maturity level, the more defenses in place, the harder it is to hack. As a result, successful bug bounty hunters need to be more skilled and invest more time, but can earn higher bounties.

What defines asset complexity, and how does it relate to security maturity and the attack surface?

Asset complexity refers to how secure or complex an asset is. Even if the asset security maturity is low, some assets are hard to hack and require attractive bounties and well-structured programs to keep skilled researchers engaged. For instance, a web application may be more accessible and widely understood, whereas a more complex messaging platform or ecosystem that requires deeper technology understanding would require more time, effort, and skills to effectively perform a security review.

Assessing each asset means analyzing the surface area, which can include endpoints and subdomains. It also means looking deeper into an asset's authentication flows, frameworks, languages, and platforms, and monitoring the rate of change, including feature pushes such as code deployments or new feature rollouts. Historical incidents and prior vulnerabilities must also be mapped to gain a clear view of asset complexity.

What is the impact on program owners and researchers alike?

When it comes to bug bounty, **program owners'** understanding of asset complexity and maturity provides insight into risk levels, coverage, and budget allocation.

For security researchers, understanding asset complexity and maturity can directly correlate to reward potential. It also means informed decisions are made regarding prioritizing risks and conducting triage.

What is the impact of asset sprawl on IT teams and researchers?

Asset sprawl occurs when organizations lose visibility and control over their assets, making it difficult to effectively manage changes. As companies develop, complex systems also grow, and when interdependent structures and hybrid environments expand, it can be challenging to perform continuously at scale.

> **"Untracked assets often go unmaintained, which is exactly what threat actors look for. The systems that seem least important can turn into hidden pathways to your most valuable data. It's like an intruder slipping through an old garage window instead of the front door. It draws less attention, but the consequences are the same."**
>
> **Inti De Ceukelaire, Intigriti**

What makes things harder is that security measures like asset discovery are often an afterthought. This has significant repercussions. Complex systems and processes can obscure elements such as lateral movement and persistence techniques. As the assets are not monitored, this lack of visibility can mean that Indicators of Compromise (IOCs) can be missed.

If you do have insights that your asset list is incomplete, then setting more inclusive scopes (like wildcards, etc.) will allow you to discover assets that were not known. If you don't have full visibility, you

can use the power of the crowd to help discover what you don't know, but your program needs to be configured to support that.

The first step to form a robust baseline for any organization is **asset discovery**.

"This involves identifying all the assets within an organization's network, including hardware, software, and any connected devices. Understanding what assets exist and their roles within the infrastructure is crucial for effective vulnerability management."- ***How to optimize your vulnerability management process.***

Incomplete view of full attack surface?

Due to the dynamic nature of modern environments, Configuration Management Databases (CMDBs) are becoming more difficult to maintain. Elements such as cloud instances and containers need to be factored in. Manual updates are time-intensive, and poor integration and a lack of tools can lead to incomplete entries. Alongside this, a lack of direction and, with it, a lack of ownership, instills a blame culture where responsibilities are not clear.

When assets are not logged or accurately tracked, they fall outside the defined scope of the program, meaning researchers can't test them. This not only limits the effectiveness of bug bounty programs but also introduces compliance risks, as unmonitored assets may fail to meet regulatory standards. Since patching priorities often rely on CMDB data, critical vulnerabilities in overlooked systems may remain unpatched, undermining both the program's value and weakening the organization's overall security posture.

Fragmented asset inventories and shadow IT

Fragmented asset inventories across multiple departments or environments can severely destabilize the effectiveness of bug bounty programs by creating an unclear or inaccurate scope. This can frustrate researchers and waste time and resources. The researchers' focus may also be misaligned to analyze low-value elements, which can lead to submission duplication, while high-risk elements go untouched.

Shadow IT refers to assets that were created without formal IT or security approval. This can include developer environments, rogue APIs, and unregistered cloud instances. These elements can bypass inventory controls and, therefore, not be included in scanning or bug bounty scopes.

How can reconnaissance identify and then reduce asset sprawl?

Reconnaissance, also known as recon, is a **method or process used to identify, validate, and organize** assets, with the purpose of understanding, managing, and reducing asset sprawl. The process involves discovering and identifying unmanaged or even unknown assets, suggesting organizational approaches to gain control, and aligning infrastructure.

In finance, reconciliation is the term used for the same approach to align records and prevent duplication and disorganized tracking. In both cases, it is essential to reduce complexity and risk.

“Bug bounty hunters who spend time performing recon are almost always rewarded well for their efforts, as they often come across forgotten assets. Skipping this process may leave certain in-scope applications or functionalities in web apps untested, resulting in decreased chances of finding vulnerabilities” - blackbird-eu

A combination of attack surface management, manual and automated tagging and classification, recon workflows, and asset discovery tools can help map and reduce the level of asset sprawl.

Complexity mapping and rewarding efforts

Complexity mapping denotes the complexity of assets, like identifying whether an asset is a hypervisor or a standard web application. For the more complex applications, you need to ensure you attract the right skills and resources by ensuring the rewards offered are aligned with the efforts.

Bug bounty tables are tiered, so that complexity matches the input of work dedicated by the researcher. For instance, more complex applications, elements with higher business criticality, or applications with higher security maturity, must have the appropriate bounties mapped to them to attract the right skills and time spent hunting for vulnerabilities. Read more about bug discovery and payout here.

If you are uncertain about any elements discussed in this article, contact the team to learn how to work with a global pool of researchers who use different tools, perspectives, and capabilities to identify vulnerabilities your internal team might miss due to limited scope, bias, or budget.

AUTHOR

Eleanor Barlow

Eleanor Barlow is a London-based Senior Cyber Security Technical Writer at Intigriti, with 9+ years’ experience reporting on and writing for the cyber and tech sector. She specializes in data-driven content on cybersecurity and bug bounty intelligence, helping organizations benefit from the latest trends and insights.