Technical

From phishing to developers: What are the new attack vectors?

Published September 22 2021 · 3 min. read

"Shift Left and Extend Right" is the primary driver for digital transformation, but it is also an attackers' paradise. Developers are getting more and more responsibility and power from design to code to cloud, but this has left security gaps unaddressed. Attackers have identified this blind spot and are targeting the developers themselves – and their identities. Software supply chain attacks are running rampant: in just one example, members of the PHP team identified malicious code commits to their interpreter using legitimate developer identities that had been compromised, along with the git.php.net server.

Today's attackers understand that their level of access will be much greater if they are able to gain legitimate access to the source code – and more.

Segregation of Duties has been a fundamental security principle for nearly as long as we've had security principles. But the lines have become blurred. By breaking down the walls between Dev and Ops, developer accounts have gained the ability to do far more – and cause more damage. With modern DevOps practices, Developers today have access to far more than just source code.

In order to make changes to production settings only a few years ago, an attacker would target system or admin accounts on individual servers. The targets evolved into administrative identities for cloud-based services, such as AWS, Azure, and GCP. But with the rise of Infrastructure as Code, there is a new target in order to access production settings: Developer accounts. Having said that, the developer account is the weakest link between application code and Infrastructure as Code and no one today is looking across application and infrastructure risks in one platform.

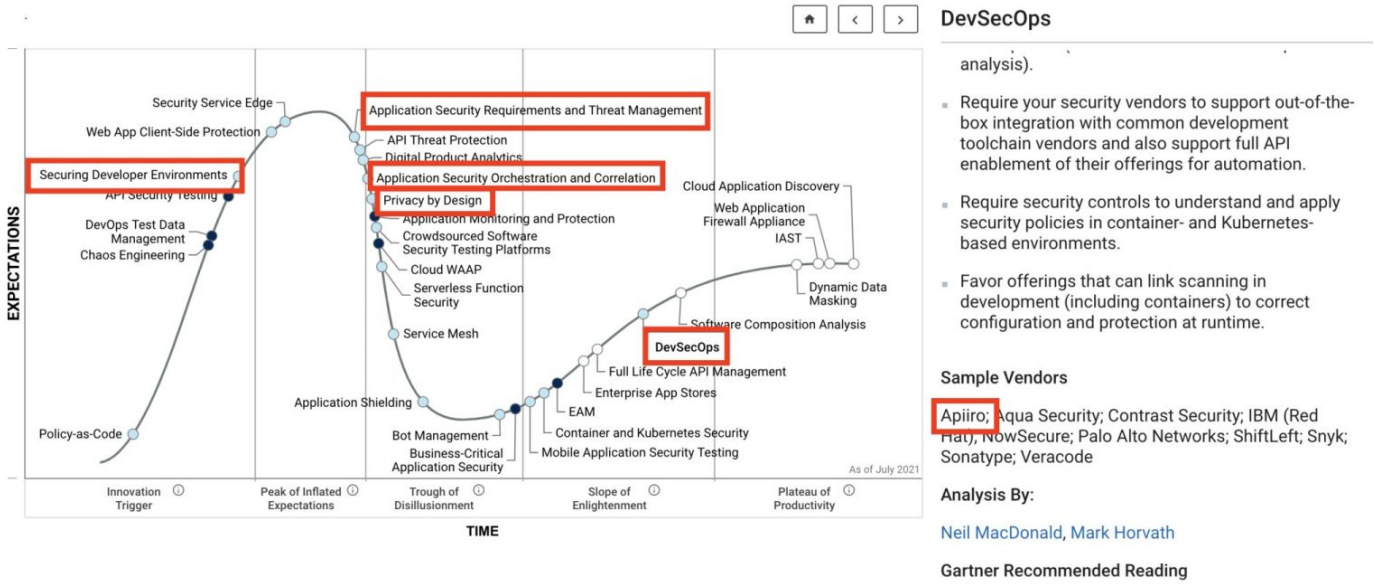
A single compromised developer account can lead to catastrophic damage in the hands of an attacker – and those accounts are often not properly secured. Unlike system admins of old who would follow a strict process to manage, access, and audit shared admin accounts and individual sysadmin identities, development accounts have very different profiles. Developers:

- Are not as well-trained in security
- Have other pressing priorities
- Will not go through an extensive series of security processes, such as using VPNs and jump servers in order to do their job

Software Bills of Materials (SBOM) are important but don't go nearly far enough. They're also ineffective when it comes to malicious code inserted into existing components.

What Must be Done

"Securing Development Environments" is in the "Innovation Trigger" stage of Gartner's Hype Cycle for Application Security, 2021.



It's easy to understand why. *Securing development environments* is not a straightforward task that can be performed in a silo and with existing tools. To be effective, it requires deep visibility across people, processes, and tools that spans the entire SDLC.

There has been no Privileged Identity/Access Management market specifically for development accounts and environments. And that is exactly what is needed. It is essential that organizations apply User and Entity Behavior Analytics (UEBA) concepts to developer identities, in order to identify abnormal activities, including compromised accounts and malicious insiders. But the UEBA engine must be significantly more contextual than simple anomaly detection. Consider the following:

- A developer commits code at 5am on a Saturday, which this particular developer has not done before
- The commit was made from an unusual location (for that developer)
- Back-end code was updated that changes an authorization control on the API gateway using Infrastructure as Code
- The developer is a front-end web developer

In order to have a full picture of the risk, it is necessary to understand the person involved, his/her peer group of contributors, the code itself (including the context, logic, and functionality), and analyze spatio-temporal behavioral information, including complex relationships between developers and other contributors in the organisation. We need to "connect the dots" in a way that gives us an accurate assessment of the risk.

"Generic" UEBA is Not Enough

In the PHP attack, there were clear indicators of abnormal activity, including:

- The commit did not match the contributor's recent activity in regards to the intensity of commits
- The commit time deviated from the expected activity day and time of the user
- The contributor deviated from the patterns common by his peers in the repository
- A major discrepancy between commit's information and analyzed code
- The added code was significantly different than predicted by the model for this repository

The key takeaway is that generic UEBA algorithms and tools would never have been able to detect this abnormality. Truly recognizing the discrepancies requires a deep understanding of the application, repositories, code, and the developers' activity patterns.

Developer accounts are a critical new attack vector, but with a comprehensive approach to developer security that spans from design to code to cloud, we also have the tools to fight back.