SECURITY JOURNEY

Reporting

Secure Coding Report: Injection Vulnerabilities

Reducing Risk in Minutes

The AppSec Dilemma

Application security faces a dilemma. Cybercrime is on the rise at an unprecedented level, with the cost of a data breach now averaging at $4.24m globally – the highest it has ever been reported in IBM's annual *Cost of a Data Breach*. This rise is affecting each industry and every area of IT. Across the cyber landscape, there are undoubtedly a variety of threats directed at web applications, but the OWASP Top 10 provides a solid breakdown on which vulnerabilities present the biggest risk. Their rankings paint a clear picture: Injection vulnerabilities have held the top spot for over 10 years, and they are still on the podium, an undeniable indication that they are endemic in Web Application Development.

In today's digital world, businesses can't afford to sit still, and pressure is mounting on developers to keep apps evolving in line with the agility of the enterprises they serve. In fact, Sourcegraph found that 51% of developers have 100x the volume of code to write compared to 10 years ago, while the same report identified that 92% feel pressured to release code faster having been tasked with driving business innovation. Yet what they're developing is not necessarily safe nor secure. Unfortunately, some developers have been taught that it is the job of the AppSec team – not the development team – to ensure the enterprise isn't at risk from any new application developments. This imbalance and siloed team approach to innovation creates a significant knowledge and process gap between the two and is where flaws then occur.

There needs to be a more security-first mindset at the development stage. However, there is a distinct lack of training and awareness of secure coding within the industry on a global basis. According to Gartner, 71% of CISOs claim their DevOps stakeholders view security as an impediment to fast development. If the pressure of time-to-market continues to mount, it is no surprise that DevOps will prioritize speed over safety just to meet their targets. Yet this is a false dichotomy – it does not have to be a choice between speed or security; both can occur simultaneously. Considering that according to Forrester, not one of the Top 40 university coding programs in the United States require secure coding training, it is hard to blame DevOps professionals for either not understanding or not prioritizing security within their day-to-day role. The same issue extends globally; we're seeing numerous initiatives being adopted by UK Research & Innovation (UKRI) to raise awareness of the concept of security by design, as well as the Institute of Coding spearheading a national initiative with leading UK Universities to encourage more security-focused teaching.

Security Concerns

Injection vulnerabilities have been at the top of OWASP Top 10 for 14 years+[1]

Financial services companies experienced a

185%

increase in the last 12 months in high-risk critical vulnerabilities.[2]

Demands on Developers

51%

of developers have 100x the volume of code vs. 10 years ago[3]

92%

feel pressured to release code faster[3]

Lack of Security Training

Secure coding is not a requirement in the top 40 university coding programs in the United States[4]

53%

of developers have no professional secure coding training[5]

1: IBM, 2: Bugcrowd, 3: Sourcegraph, 4:Forrester, 5:Ponemon

Here lies the dilemma: many within the industry do not even realize that this lack of knowledge and understanding of secure code is an issue, let alone one that has a **simple solution**.

In fact, there are areas within application security that are both critical to an organization's protection from a data breach and also incredibly simple to solve.

This research paper presents a central solution to the application security dilemma, as well as identifying developer best practice from analysis provided by **Derek Brink, VP and Research Fellow at Aberdeen Strategy and Research**, of approximately 140,000 HackEDU exercises. Every exercise has been analyzed with the aim to generate know-how of how to protect against **Injection vulnerabilities**, as well as helping educate on the importance of introducing security from the very start of application development. HackEDU believes short and continuous training exercises can help close the gap between security initiatives and developer knowledge. Yet the importance of these initiatives has not yet been realized and it is rarely communicated across the industry. This paper outlines the benefits of secure coding training, analyzes the means to implementing programs and demonstrates how your organization can avoid unnecessary costs while reducing risk in the ever-evolving cybersecurity landscape.

> "For businesses that recognize the value of security training for their teams but worry that developers are already under pressure with numerous time constraints, these findings highlight that learning to write secure code takes far less time than you may think."
>
> Derek Brink, VP and Research Fellow at Aberdeen Strategy and Research

Embracing DevSecOps to Enable Shifting-Left

For those in the world of IT, DevSecOps (an approach to culture, automation, and platform design that integrates security as a shared responsibility throughout the entire IT lifecycle) is a concept that's hard to avoid and it is often touted as the future of application development.

Shift-left is one of the possible features of DevSecOps, which means application security testing is performed earlier in the development process with the hopes of preventing vulnerabilities from reaching production code. Shift-left is becoming business-critical now that cybercrime is so widespread. Fortunately, Gartner has predicted that DevSecOps will reach mainstream adoption within the next two to five years, and that 90% of software development projects plan to incorporate DevSecOps practices by the end of this year.

But how is a culture of DevSecOps achieved? At the foundation, it means ensuring security practices are part of the entire software development lifecycle (SDLC), rather than being an afterthought (or only introduced once vulnerabilities have been identified and as a reactive remediation). However, for developers to prioritize secure coding, they need consistent hands-on training, regular education and the practical know-how to do so. Secure coding practices are important to keep applications safe, but too many organizations simply aren't sure how to train their developers. What's more, while DevSecOps may be the desired end-goal, it is not enough to implement a broad security awareness training program. Instead, time should be dedicated towards education of secure coding practices rather than just awareness of the issue. To resonate with developers and support their secure coding success, role-based training should also be encouraged in a way they are familiar with (e.g., playing with code to find the solution). Understanding what your developers want, how they learn, and the vulnerabilities that will be most critical to them is an essential part of successfully shifting-left.

The Injection Vulnerability Conundrum

There are a variety of application security training programs that together can form a strong foundation for secure coding best practice and support the move to DevSecOps. And with 90 – 95% of confirmed data breaches involving attacks on web applications according to Verizon's Data Breach Investigations Report (DBIR), continuous training and awareness around how to protect these applications can make a significant impact. However, this paper looks specifically at Injection flaws.

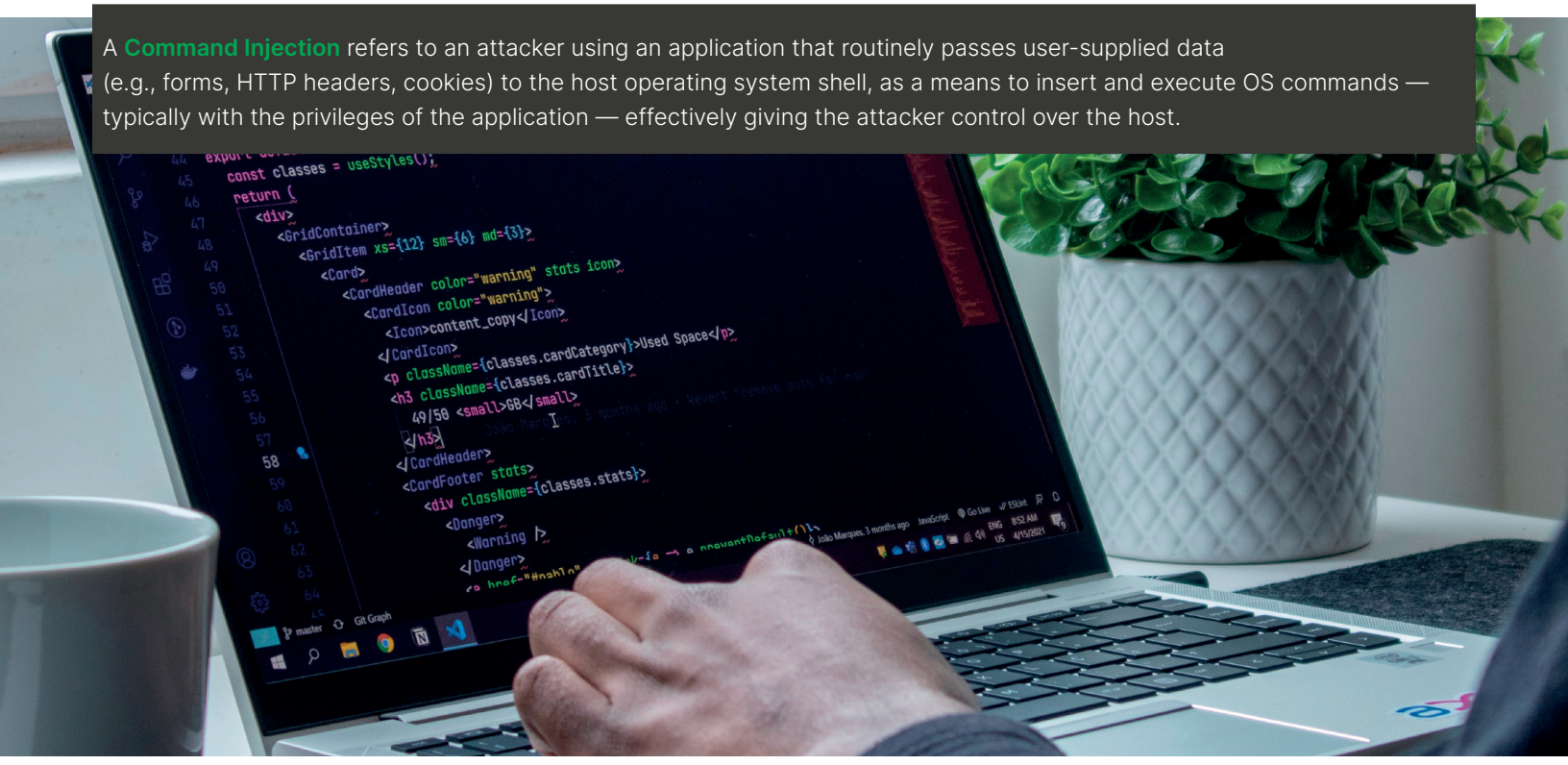
This category of vulnerabilities spent over 10 years at the very top of the OWASP Top 10 and now remains in the top three most critical flaws that developers need to be aware of. Findings from HackEDU analysis with Derek Brink highlight that the likelihood of applications having an Injection flaw ranges from 0% to 19.09%, with a median of 3.37%. These kinds of application vulnerabilities are some of the most prolific on the web and can be devastating to any organization, but are particularly threatening to organizations that store sensitive and / or regulated information about their customers and workforce. Latest industry reports have found that 35% of educational institutions and 32% of government organizations were vulnerable to SQL Injections (SQLi) last year.

SQLi involves an attacker using the standard data input interface of a SQL database to insert a database query, which can potentially compromise the privacy, integrity or availability of the database, or execute other database admin operations. It sits alongside Cross-Site Scripting (XSS), XML External Entities (XXE) and Command Injection as one of the oldest and easily mitigated web application vulnerabilities out there.

Cross-Site Scripting (XSS) refers to an attacker using a trusted web application to send malicious code to an end-user's browser, where it can potentially hijack the user's session, access sensitive data (e.g., passwords, account numbers, payment card data, personally identifiable information), and achieve a successful account takeover.

In an **XML External Entity (XXE)** attack, an attacker uses a web application's eXtensible Markup Language (XML) interface to insert and process XML communications with the application's trusted data sources / recipients (referred to as "external entities") — typically resulting in unauthorized access to data, or as a means to execute commands on remote systems.

A **Command Injection** refers to an attacker using an application that routinely passes user-supplied data (e.g., forms, HTTP headers, cookies) to the host operating system shell, as a means to insert and execute OS commands — typically with the privileges of the application — effectively giving the attacker control over the host.



Prevention is Better than Cure

It is not only easy to mitigate Injection vulnerabilities with continuous secure coding training, but also it is neither a time intensive nor expensive process to do so. The cost benefits are threefold:

1 Secure applications equate to cost avoidance.

If coded securely, an application is protected from the risks and repercussions from a successful exploit. Of course, it is possible to remediate security flaws within the software code after it has been in production, but this process is highly problematic. While it takes months to identify the issue (research shows an average of 254 days to discover an incident related to web application exploits), it is also extremely time consuming and expensive to do so. This falls in line with Boehm's law: 'the cost of finding and fixing a defect grows exponentially with time.' If this cost is avoidable, wouldn't you avoid it?

2 Penetration testing is an expensive alternative.

While penetration testing is a valuable solution, it should not be used as the first opportunity to enhance security. With developers trained in secure coding, penetration testing can become a useful assessment of a security approach already implemented. However, if applications are not securely coded and a penetration test is used as a first line of defense, it will likely flag a number of flaws that need remediating. To ensure these flaws are addressed, numerous follow-up penetration tests will be required. Why not reduce the ongoing investment in pen testing by learning to do it right the first time?

3 Data breaches can be extremely costly.

A data breach is likely to have multiple layers of financial consequences. This could be a ransom payment (63% of organizations still pay, despite international cybersecurity advice not to), but it is also the cost of recovery; the days or months dedicated to restoring systems rather than driving revenue; the money spent on external MSSP support to bolster protection; and the reputational damage and corresponding decline in sales once customers are aware of the issue. Investing in a continuous approach to secure coding today will be futureproofing your tomorrow.

HackEDU Injection Vulnerability Training: Our Findings

To uncover and share the true value of secure coding training and proactively avoiding web application flaws rather than remedying them later down the line, Derek Brink analyzed six HackEDU courses in the Injection vulnerabilities space. Looking at nearly 140,000 exercises taken by developers on this subject across the past year, the aim was to identify how a selection of HackEDU's exercises quickly and successfully trained developers to write more secure code, and therefore reduce the likelihood of a data breach within their organizations.

140,000
exercises analyzed

Only
45%
of the developers were
100%
successful in their first attempt to pass

93%
were able to find and fix SQLi after less than
10 minutes
of training

Brink's analysis uncovered that across the six Injection vulnerability courses, 45% of the developers were 100% successful in their first attempt to pass. This left over 50% of developers demonstrating the need for additional time in secure coding training – both contextually on why security matters, and practically with a hands-on approach to securing their code against Injection flaws.

The results showed that only 7% of developers were unable (at least as of year-end) to successfully demonstrate the required skills to find and fix Injection vulnerabilities, even after repeated exercise attempts. The other 93% of developers had successfully learned the necessary skills and were able to now remedy Injection vulnerabilities and protect their organizations from data breaches linked to this source. What's more, 93% of them were able to find and fix SQLi after less
than 10 minutes of training. While these results only highlight the secure coding solution for one vulnerability, and regular training is necessary across all vulnerabilities to ensure better application security, it demonstrates the ease at which a critical and pervasive flaw can be solved.

Derek Brink comments on the findings, "For businesses that recognize the value of security training for their teams but worry that developers are already under pressure with numerous time constraints, these findings highlight that learning to write secure code takes far
less time than you may think. In fact, the median time for each exercise on Injection vulnerabilities is only 4.1 minutes! If you can reduce the risk of one of the most common web application flaws in such a short timeframe, isn't it worth it?"

> "If 93% of your development organization can protect against an issue that is commonly the root cause of a data breach, with so little time commitment, wouldn't you at least **consider it**?"
>
> Derek Brink, VP and Research Fellow at Aberdeen Strategy and Research

How to “Get it Right” with Training Best Practice

The findings demonstrate that protecting web applications from one of the biggest vulnerabilities online is achievable, despite the ongoing pressure on developers to innovate fast. Learning to incorporate security early on in the SDLC is both simple and highly effective. However, how can organizations make sure they get it right when it comes to a secure coding training program? There are a number of best practice considerations that will enable your team to get the most out of a secure coding initiative.

Set measurable goals. To ensure a training program is successful, it is important to gather information that you can measure against later down the line. This could be the number of vulnerabilities that appear within a developer’s code both before and after the training, as well as their time spent fixing vulnerabilities and the number of vulnerabilities that they can detect and fix.

Collaborate with stakeholders. Communicating and working with stakeholders is key, because without their buy-in and support, chances of success are low.
Without sharing and championing the importance of secure coding training from the very start, you may constantly be defending its value and it will likely be undermined. It takes time to build a successful security culture, but without support of key stakeholders, it may be impossible.

Embrace learning science principles. This includes shorter, ‘bite-sized’ exercises, as many learn better when the information is more digestible. Alongside short exercises, it is key that secure coding is taught both ‘offensively’ (understanding how an exploit works) and ‘defensively’ (finding and fixing vulnerabilities).
This means that developers are provided the context and the ‘why’ behind the solutions they’re learning, as without an understanding of why this change matters, it becomes much harder to accept and integrate. With ‘defensive’ training, a hands-on, practical approach will help to engrain key learnings.

Create a continuous program (with incentives!) ‘One and done’ training simply does not work. In order to keep security front of mind throughout the year, repeating elements of the training will help developers retain knowledge. According to memory theory, after
31 days around 79% of learning is lost. Therefore, monthly competitions – with incentives and rewards – is a great way to ensure learnings aren’t seen once and forgotten forever.

Measure and optimize. As developers complete their training, collect feedback along the way and analyze the results to see what worked and what can be done better next time. Talk to the developer community and see what they want and need in future. Cybersecurity is ever changing and your approach to secure coding should mirror this.

Closing Thoughts

In an era where cybercrime is growing and evolving, the number of solutions and tools to protect businesses is overwhelming and global cybersecurity skills are at an all-time low. It has therefore never been more important to prioritize continuous training to help mitigate risk. It has also never been easier and less time intensive to train a team of developers to protect against one of the most prolific threats to application security. Derek Brink’s analysis proved that while over 50% of developers need more secure coding training, once a short training program is completed, 93% are able to reduce the risk of one vulnerability exposing their organization.

While this paper looked exclusively at Injection vulnerabilities as just one example within the broad range of secure coding training courses, numerous other risks and vulnerabilities will need addressing in order to secure web applications and continue with the shift-left to a world of DevSecOps. However, by beginning this journey to a more security-first culture with Injection vulnerability courses, the value of training will be quick and easy to see – both for the developers and for the company stakeholders that may be harboring doubts. Baking in security from the early on the SDLC will also save organizations significant costs in the long run, and a small upfront investment will be the key to saving company revenue and reputation moving forward.

There are various ways this secure coding training can be implemented. Yet with best practice adhered too, developers listened to, and communication prioritized, it is possible to make a palpable difference to enterprise security. Application security no longer needs to be a dilemma. Instead, developers can become a knowledgeable first line of defense and an indispensable component in your security strategy.

SECURITY JOURNEY

HackEDU's spring 2022 acquisition of Security Journey brings together two powerful platforms to provide application security education for developers and the entire SDLC team. The two officially became one in August 2022 and are now Security Journey. Two approaches, one path to build a security-first development culture.

www.SecurityJourney.com | info@SecurityJourney.com

SECURITY JOURNEY