Making a Career in Application Security (AppSec)

Secure the code. Find bugs before attackers do.

AUTHOR	Babashaheer
VERSION	1.0
DATE	April 2026
SERIES	Cybersecurity Career Series — Document 06 of 09
AUDIENCE	Developers and security professionals moving into AppSec

Contents

Digitally signed by Baba Shaheer Gutappa
Date: 2026.04.20 12:31:00 +01'00'

1. What Is Application Security?

Every website, mobile app, API and piece of software has security vulnerabilities in it. Some are minor. Some allow an attacker to steal the personal data of millions of people. Application security — AppSec — is the discipline of finding and fixing those vulnerabilities before attackers do, and building systems so that fewer vulnerabilities are introduced in the first place.

AppSec sits at the intersection of software development and security. It is not purely a hacking job, and it is not purely a development job. The best AppSec professionals understand both worlds — they can read code, understand how a web framework works, explain a vulnerability to a developer clearly, and test whether a security fix actually resolved the issue.

The majority of data breaches involve a web application vulnerability.
Verizon's 2023 DBIR found that web application attacks were involved in 26% of all breaches. AppSec professionals are the people whose job is to prevent those breaches from happening.

Level	Typical UK Salary	Common Roles
Junior	£32,000 – £50,000	Junior AppSec Engineer, Security Analyst (Application), Developer with security focus
Mid-level	£55,000 – £80,000	AppSec Engineer, Security Champion, Secure Code Reviewer, Penetration Tester (Web)
Senior	£80,000 – £110,000+	Senior AppSec Engineer, Application Security Architect, Head of Product Security
Lead / Principal	£100,000 – £140,000+	Principal AppSec Engineer, VP of Product Security, CISO (product-focused org)

Table 1: UK salary ranges for application security roles (CW Jobs, 2024)

2. AppSec vs Other Cybersecurity Roles

AppSec overlaps significantly with web application penetration testing — but they are not the same thing. A penetration tester is called in periodically to test a live system. An AppSec engineer is embedded in the development process to prevent vulnerabilities from being built in the first place.

	AppSec Engineer	Web App Penetration Tester	Secure Developer
Core question	How do we build this securely?	Can I break into this?	Am I writing this safely?
Timing	Throughout the SDLC — continuous	Periodic — quarterly or annual	During coding — daily
Primary output	Secure design, SAST findings, security requirements	Pentest report with findings	Secure code, no vulnerabilities
Works with	Development teams, architects	Client security team	Own team and AppSec engineers
Tools	SAST tools, threat modelling, code review	Burp Suite, OWASP ZAP, manual testing	Linting, SCA, IDE security plugins
Suits people who	Enjoy coding and want to add security depth	Enjoy offensive testing	Enjoy writing code and care about quality

Table 2: AppSec engineer vs web app pentester vs secure developer

3. Security in the Software Development Lifecycle

The most important shift AppSec has driven in the last decade is moving security left — meaning earlier in the development process. Fixing a vulnerability after a product is live costs roughly 30 times more than fixing it during design. The Secure Software Development Lifecycle (SSDLC) integrates security at every phase of how software is built.

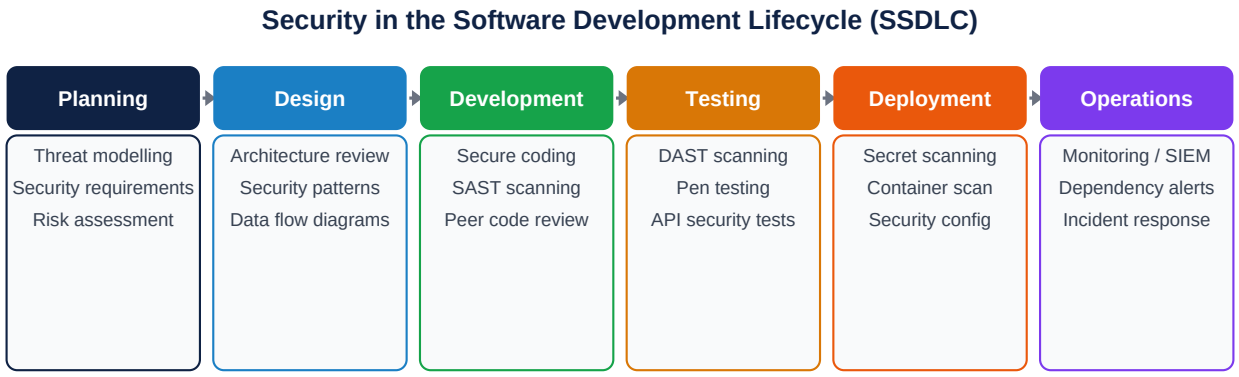


Figure 1: Security activities integrated at every phase of the Software Development Lifecycle (SSDLC)

4. The OWASP Top 10 — What Every AppSec Professional Must Know

The Open Web Application Security Project (OWASP) Top 10 is the most widely referenced list of critical web application security risks. Published every few years, it is based on data from real-world breaches and vulnerability assessments. Every AppSec professional, developer and security tester should know all ten inside out — what the vulnerability is, how it is exploited and how to prevent it.

OWASP Top 10 — 2021 Edition

A01	Critical	**Broken Access Control**	Users access data or functions beyond their permissions. Most
A02	Critical	**Cryptographic Failures**	Sensitive data exposed due to weak or missing encryption in transit
A03	Critical	**Injection (SQLi, XSS)**	Untrusted input interpreted as code. SQL injection, command
A04	High	**Insecure Design**	Security flaws in the architecture itself, not just the implementation.
A05	High	**Security Misconfiguration**	Default credentials, verbose errors, unneeded features, missing
A06	High	**Vulnerable Components**	Third-party libraries and frameworks with known CVEs used without
A07	Critical	**Auth & Session Failures**	Weak passwords, missing MFA, poor session management,
A08	High	**Software & Data Integrity**	Insecure update mechanisms, unsigned code, insecure
A09	Medium	**Logging & Monitoring Failures**	Insufficient logging means attacks go undetected for months.
A10	High	**Server-Side Request Forgery**	Server makes requests on behalf of attacker — bypasses firewalls,

Figure 2: OWASP Top 10 (2021). Critical = highest severity, most commonly exploited vulnerabilities.

Deep dive: The most important three

A01 — Broken Access Control

An application shows different data to different users — an admin sees everything, a regular user sees only their own records. Broken access control means those boundaries are not enforced properly. The simplest example: if changing the URL from */account/1234* to */account/1235* shows another user's data, access control is broken. This is now the single most common web vulnerability and is involved in the majority of significant data breaches.

A03 — Injection

Injection vulnerabilities occur when user-supplied input is interpreted as code rather than data. SQL injection — where an attacker manipulates a database query — is the classic example. If a login form passes the username directly into a SQL query without sanitisation, an attacker can enter SQL syntax that changes the query's logic completely.

sql_injection_example.py

```
# VULNERABLE — string concatenation, never do this
query = "SELECT * FROM users WHERE username='" + username + "'"

# Attacker input: ' OR '1'='1
# Resulting query: SELECT * FROM users WHERE username='' OR '1'='1'
# Result: returns ALL users — authentication bypassed

# SAFE — parameterised query (prepared statement)
query = "SELECT * FROM users WHERE username = ?"
```

Code 1: SQL injection — vulnerable vs safe parameterised query pattern

5. Threat Modelling

Threat modelling is a structured process for identifying security risks in a system before it is built. It is one of the highest-value AppSec activities because it catches design-level vulnerabilities — the kind that are very expensive to fix once the system is live. The most widely used methodology is STRIDE.

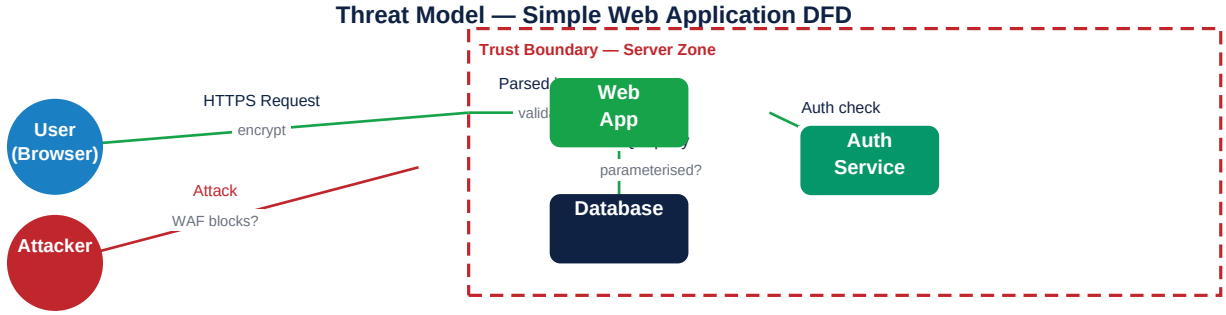


Figure 3: Simple Data Flow Diagram (DFD) for a web application. Threat modelling maps threats to each element and flow.

STRIDE Threat	What It Means	Example Attack	Mitigation
Spoofing	Pretending to be someone you are not	Login with stolen credentials	Authentication, MFA, certificate pinning
Tampering	Modifying data in transit or at rest	Man-in-the-middle modifying API response	Integrity checks, digital signatures, HTTPS
Repudiation	Denying an action took place	Attacker deletes logs after accessing data	Audit logging, tamper-evident log storage
Information Disclosure	Exposing data to unauthorised parties	SQL injection exposing database contents	Encryption, access controls, input validation
Denial of Service	Making a system unavailable	Flooding an API with requests	Rate limiting, WAF, DDoS mitigation
Elevation of Privilege	Gaining more access than authorised	Exploiting a broken access control to become admin	Least privilege, authorisation checks, RBAC

Table 3: STRIDE threat modelling framework — six categories of threats

The four questions of threat modelling (Microsoft):

1. What are we building? 2. What can go wrong?
3. What are we going to do about it? 4. Did we do a good job?

6. Static Analysis (SAST) — Finding Bugs in Code

Static Application Security Testing (SAST) analyses source code — or compiled code — without executing it. SAST tools scan every line of code looking for known vulnerability patterns: SQL queries built with string concatenation, user input passed to dangerous functions, hardcoded credentials, and so on. The advantage of SAST is that it finds bugs before the code runs. The disadvantage is false positives — it reports potential issues that may not be real vulnerabilities in context. AppSec engineers triage these findings.

SAST Tool	Languages	Best For	Cost
Semgrep	Python, JS, Java, Go, Ruby, many more	Custom rule writing. Fast. Excellent CI/CD integration. Community rules for common vulnerabilities.	Free (OSS) / Commercial
SonarQube	25+ languages	Enterprise SAST with dashboards, issue tracking and quality gates. Widely used in large dev teams.	Free (Community) / Commercial
Checkmarx SAST	30+ languages	Enterprise-grade SAST with deep data flow analysis. Heavy but thorough.	Commercial
Snyk Code	JS, Python, Java, C#, Go, Ruby	Developer-friendly SAST with IDE plugins. Good for shift-left embedding in developer workflow.	Free tier / Commercial
Bandit	Python only	Lightweight Python security linter. Perfect for quick Python codebase scans.	Free
ESLint security plugins	JavaScript / TypeScript	Catch common JS security issues during development. Integrates with any JS project.	Free
CodeQL	C, C++, Java, Python, JS, Go, Swift	GitHub's analysis engine. Write queries to find complex vulnerability patterns.	Free (GitHub)

Table 4: Common SAST tools, languages supported and use cases

SAST finds what the code could do wrong. DAST finds what the running app actually does wrong. Neither alone is sufficient — a mature AppSec programme uses both.

7. Dynamic Analysis (DAST) — Testing the Running Application

Dynamic Application Security Testing (DAST) tests a running application from the outside — just as an attacker would. It sends malformed input, attempts injections, probes authentication, and checks security headers. Unlike SAST, DAST does not need access to source code. It is also more likely to find vulnerabilities that only manifest at runtime, such as authentication flaws or business logic issues.

DAST Approach	How It Works	What It Finds	Tool Examples
Automated scanning	Tool sends thousands of test cases to the running app automatically	Common vulnerabilities: XSS, SQLi, open redirects, missing security headers	OWASP ZAP, Burp Suite Professional, Nikto
Manual testing	Security engineer manually tests the application with knowledge of its functionality	Business logic flaws, complex access control bypasses, chained vulnerabilities	Burp Suite, manual HTTP manipulation
API security testing	Specifically targets REST, GraphQL and SOAP APIs — often missed by web scanners	Mass assignment, broken object-level auth, rate limit bypass	Postman + security checks, OWASP ZAP, Burp Suite
Authenticated scanning	DAST with valid user credentials — tests deeper application functionality	IDOR, privilege escalation, authenticated injection points	Burp Suite with session handling, ZAP with authentication
IAST (Interactive)	Agent inside the running app observes internally as the app is exercised	Very accurate — low false positives — sees exactly what code paths are triggered	Contrast Security, Seeker

Table 5: DAST approaches compared

8. DevSecOps — Security in the Pipeline

DevSecOps extends the DevOps philosophy — fast, automated, iterative software delivery — to include security at every step. Rather than a security review happening at the end of a project, security checks run automatically every time a developer pushes code. Vulnerabilities are caught in minutes, not months.

The DevSecOps pipeline

github_actions_security.yml

```
# .github/workflows/security.yml — example GitHub Actions pipeline

on: [push, pull_request]

jobs:
  sast:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Run Semgrep SAST
        run: semgrep --config=auto --error  # Fail build on HIGH findings

  sca:                          # Software Composition Analysis
    steps:
      - name: Snyk dependency scan
        run: snyk test --severity-threshold=high

  secrets:
    steps:
      - name: Scan for hardcoded secrets
```

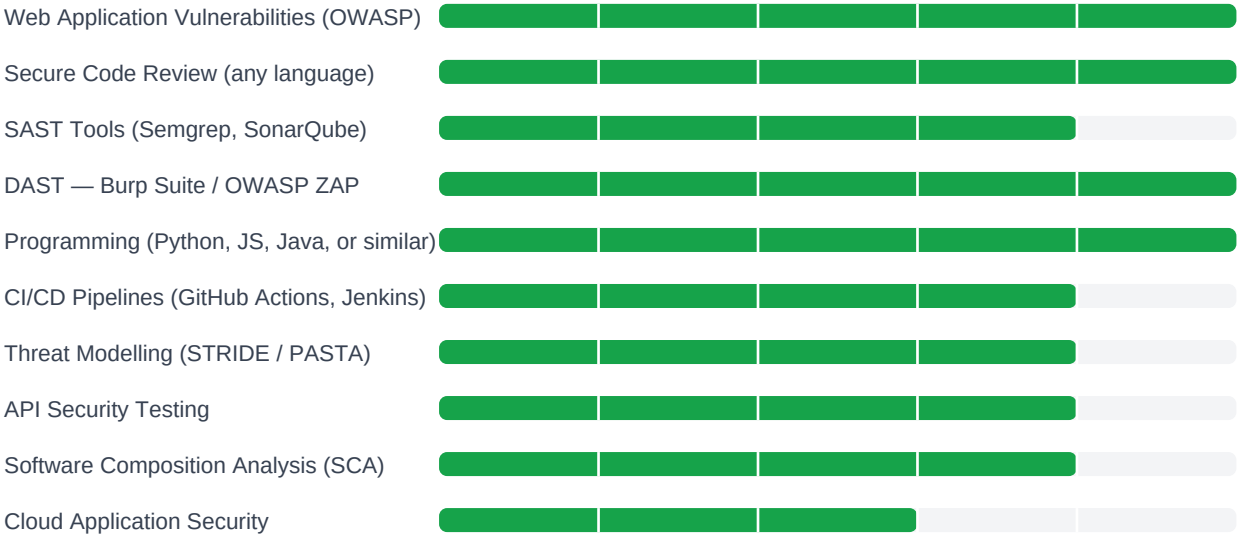
Code 2: Example GitHub Actions DevSecOps pipeline — SAST, SCA and secret scanning on every commit

Pipeline Stage	Security Activity	Tools
Pre-commit (developer machine)	IDE security plugins, pre-commit hooks run lightweight checks	Snyk IDE plugin, git-secrets, detect-secrets
Commit / PR	SAST scan, secret scanning, licence compliance check	Semgrep, TruffleHog, Snyk Code, GitHub Advanced Security
Build	Software Composition Analysis (SCA) — scan dependencies for CVEs	Snyk Open Source, OWASP Dependency-Check, Dependabot
Test	DAST against staging environment, API security tests	OWASP ZAP, Burp Suite Pro, Postman security tests
Container / IaC	Scan Docker images for CVEs, scan Terraform/Helm for misconfigurations	Trivy, Checkov, Grype, Docker Scout
Deploy	Security configuration checks, runtime protection enabled	OPA policy checks, RASP agents, WAF rules applied
Monitor	Runtime threat detection, dependency vulnerability alerts	Dependabot, Snyk Monitor, RASP, WAF logging

Table 6: Security activities integrated at each stage of the DevSecOps pipeline

9. Core Skills and Tools

Skills expected at mid-level AppSec engineer:



Tool	Category	What It Does	Cost
Burp Suite Pro	DAST / Manual Testing	The industry standard for web app security testing. HTTP proxy, scanner, repeater, intruder, collaborator.	Commercial (Community free)
OWASP ZAP	DAST	Open-source web scanner. Good for automated scanning in CI/CD pipelines. Active and passive scanning.	Free
Semgrep	SAST	Fast, customisable SAST with huge community rule library. Language-agnostic. Easy CI integration.	Free / Commercial
SonarQube	SAST	Quality and security platform for continuous code inspection. Dashboard, history, quality gates.	Free (Community) / Commercial
Snyk	SCA + SAST	Developer-first security — IDE plugins, CI integration, dependency and code scanning.	Free tier / Commercial
PortSwigger Web Security Academy	Learning	Free structured learning platform for web vulnerabilities. Labs cover every OWASP Top 10 category.	Free
TruffleHog / detect-secrets	Secret Scanning	Scans code repositories for hardcoded API keys, passwords and secrets.	Free
OWASP Dependency-Check	SCA	Identifies known CVEs in project dependencies (Java, .NET, Python, JavaScript).	Free
Trivy	Container / IaC Security	Scans container images and IaC files for vulnerabilities and misconfigurations.	Free

Tool	Category	What It Does	Cost
Postman + security tests	API Testing	API testing platform. Add security test assertions to your API test collections.	Free tier / Commercial
OWASP ASVS	Standard / Reference	Application Security Verification Standard — detailed security requirements for applications.	Free (document)

Table 7: Core tools for application security professionals

10. Career Paths in Application Security

AppSec offers career paths that suit both technical security specialists and developers who want to specialise in security. The two most common backgrounds are: security professionals who learn development, and developers who move into security.

Role	What You Do	Where You Work	UK Salary
Junior AppSec Engineer	Triage SAST/SCA findings, assist in code reviews, run DAST scans, write security test cases.	Tech companies, banks, consultancies	£32k–£50k
AppSec Engineer	Lead security reviews, embed in development teams, manage vulnerability programme, run threat modelling sessions.	Product companies, fintech, SaaS	£55k–£75k
Web App Penetration Tester	Focused on DAST — manually test web applications and APIs for vulnerabilities. Write pentest reports.	Consulting firms, MSSPs	£45k–£70k
DevSecOps Engineer	Build and maintain the security tooling pipeline — SAST, SCA, secret scanning, container scanning in CI/CD.	Tech-forward companies, cloud-native orgs	£60k–£85k
Security Champion	Developer embedded within a development team who has extra security training — first point of contact for secure coding questions.	Large enterprises with dev teams	£50k–£70k (dev salary with security bonus)
AppSec Architect	Design secure application architectures. Define security standards, patterns and frameworks for the engineering organisation.	Large enterprises, financial services	£80k–£110k
Head of Product Security	Own the security of all products. Manage AppSec team. Interface with CISO and product leadership.	Product companies, scaleups	£100k–£140k+

Table 8: Career paths in application security (Reed, 2024; CW Jobs, 2024)

11. The Learning Roadmap

AppSec is one of the most accessible specialisms for developers — if you already write code, you have the most valuable foundation. The roadmap below works for both developers adding security knowledge and security professionals adding development knowledge.

1 — Learn web fundamentals deeply
Understand HTTP/S — requests, responses, cookies, sessions, headers. Know how a browser makes requests and how a server responds. Understand what happens when you submit a login form. Without this, web security concepts are difficult to follow. 2–3 weeks.

2 — Work through PortSwigger Web Security Academy
The best free resource for learning web application security. Structured labs covering every OWASP Top 10 category with hands-on vulnerable applications. Complete every lab. This is what professionals actually use to learn. 8–12 weeks ongoing.

3 — Learn to use Burp Suite
Install Burp Suite Community. Set it up as a proxy for your browser. Work through every PortSwigger lab using Burp. Understand the repeater, intruder, decoder and scanner modules. This is the tool every AppSec professional uses daily. 3–4 weeks.

4 — Get comfortable with at least one programming language
Python for scripting and automation. JavaScript for front-end vulnerability understanding. Java or C# for enterprise application context. You must be able to read code in the language your target applications are written in. Ongoing.

5 — Learn SAST — run tools on real code
Install Semgrep. Run it against open-source codebases. Triage the findings — understand why each one is (or is not) a real vulnerability. Contribute custom rules. This hands-on SAST experience is directly transferable to a job. 2–3 weeks.

6 — Learn threat modelling
Read the OWASP Threat Modelling Cheat Sheet. Watch Adam Shostack's free threat modelling videos. Practice by threat modelling a simple web application you understand. This is a skill that requires practice, not just reading. 2–3 weeks.

7 — Build a DevSecOps lab
Set up a GitHub repository. Add GitHub Actions CI/CD. Integrate Semgrep, Snyk and TruffleHog into your pipeline. Watch it catch intentional vulnerabilities you introduce. Document the setup. This becomes a portfolio piece. 2–3 weeks.

8 — Get certified and apply
eWPT (eLearnSecurity Web Penetration Tester) or BSCP (Burp Suite Certified Practitioner — PortSwigger's own cert) for practical web testing skill. GWEB (GIAC) for broader AppSec. CompTIA Security+ as a baseline if needed.

Timeline: From developer to AppSec engineer: 9–15 months.
From security background adding AppSec: 6–9 months.
PortSwigger Web Security Academy is the single most valuable resource in this field.

12. Certifications That Matter

AppSec certifications split into two categories: those focused on web application penetration testing (offensive) and those focused on AppSec programme management and secure development (defensive). A well-rounded AppSec professional benefits from both.

Certification	Level	Provider	Focus	Why It Matters
BSCP — Burp Suite Certified Practitioner	Mid	PortSwigger	Practical web application security testing using Burp Suite	Highly regarded. Issued by the makers of the industry's primary testing tool. Demonstrates real hands-on ability.
eWPT — Web Penetration Testing	Beginner–Mid	INE / eLearnSecurity	Web application penetration testing — practical exam format	Practical, affordable. Good first AppSec cert demonstrating hands-on testing ability.
GWEB — GIAC Web Application Defender	Mid	GIAC / SANS	Defending web applications — AppSec programme, OWASP, secure development	Respected for AppSec programme management roles. More defensive than the other certs.
OSWE — Offensive Security Web Expert	Senior	Offensive Security	Advanced source code review and white-box web application exploitation	Very challenging. Highly respected for senior AppSec and secure code review roles.
CSSLP — Certified Secure Software Lifecycle Professional	Mid–Senior	ISC2	Secure software development lifecycle — all phases from design to deployment	Broad SSDLC cert. Valued in enterprise and regulated industry AppSec management roles.
CompTIA Security+	Beginner	CompTIA	Broad security foundations	Baseline requirement for many roles. Good starting point if coming from development.

Table 9: Application security certifications in order of progression

13. Case Study — A SQL Injection That Cost £2.1 Million

> This is a fictional case study based on real-world SQL injection breach patterns.
> SQL injection has been the most commonly exploited web vulnerability for over 20 years.

The Organisation

Prism Retail Group is a UK e-commerce company selling electronics. They process approximately 4,000 orders per day through their custom-built web application. Their customer database holds the personal and payment data of 620,000 registered customers. They had not conducted a web application security test in three years. Their development team of 12 had no AppSec engineer embedded.

The discovery — a security researcher

Security researcher discovers an obvious vulnerability

A security researcher testing the Prism Retail website notices something unusual in the product search URL: prismretail.co.uk/search?q=laptop

They add a single quote to the query: prismretail.co.uk/search?q=laptop' The server returns a database error message: You have an error in your SQL syntax near '"' at line 1. This is a textbook sign of SQL injection — the input is going directly into a SQL query without sanitisation.

The exploitation

SQLMap automated extraction — full database dump

Using SQLMap (an open-source automated SQL injection tool), the researcher is able to extract the entire customer database in under 40 minutes. No authentication required. No rate limiting. No WAF to block the automated requests.

The database contains: 620,000 customer records — names, email addresses, delivery addresses, hashed passwords (MD5, unsalted — easily crackable). Also exposed: 18,400 stored payment card tokens (tokenised but stored

The responsible disclosure and breach notification

72-hour GDPR clock starts — ICO notification required

The researcher notifies Prism Retail via their contact page. The message sits in a general enquiries inbox for 11 hours before being seen by a developer. The vulnerability is confirmed and patched with a parameterised query within 4 hours of discovery.

However, the question of how long the vulnerability existed — and whether it had been previously exploited — cannot be answered. Server logs show insufficient detail. CloudTrail equivalent had not been enabled. The worst-case scope:

The cost breakdown

Cost Category	Details	Estimated Cost
Legal advice and ICO management	External lawyers, regulatory response, two ICO investigations.	£180,000
ICO fine	UK GDPR breach — inadequate security measures, failure to patch known vulnerability class.	£425,000

Cost Category	Details	Estimated Cost
Customer notification	Email and letter campaign to 620,000 customers with guidance on steps to take.	£95,000
Forensic investigation	Determine scope of breach and whether unauthorised access occurred prior to disclosure.	£120,000
Credit monitoring	Offered to high-risk customers — 6 months credit monitoring service.	£55,000
Emergency AppSec programme	Contracted external AppSec firm for full web application security assessment and code review.	£85,000
Brand and revenue impact	Customer trust declined. Revenue dropped 18% in the three months post-disclosure.	~£1,140,000
TOTAL		**~£2,100,000**

Table 10: Estimated cost breakdown of the Prism Retail SQLi breach

What an embedded AppSec engineer would have prevented

- A SAST scan (Semgrep, SonarQube) would have flagged the string-concatenation SQL queries on day one of code review
- A DAST scan (OWASP ZAP, Burp Suite) run against staging would have detected the SQLi in under 5 minutes
- An annual web application penetration test — the industry baseline — would have caught this within three years
- A secure coding training session covering parameterised queries would have prevented it at the developer level
- A WAF rule blocking SQLMap's user agent and payload patterns would have at minimum slowed the automated exploitation

An AppSec engineer costs approximately £55,000–£70,000 per year.
This breach cost £2,100,000.
30 years of AppSec engineering for the cost of one avoidable breach.

14. Breaking In — Getting Your First Role

AppSec is one of the most achievable cybersecurity career transitions for developers. Your coding background is genuinely valued — an AppSec engineer who can read the code they are reviewing is significantly more effective than one who cannot.

Build visible evidence of AppSec skill

- **PortSwigger Web Security Academy profile:** Complete all labs across all vulnerability categories. Your completion is visible on your profile — link it on your CV and LinkedIn
- **Bug bounty:** HackerOne and Bugcrowd list programmes with defined scope. Finding and responsibly disclosing even a low-severity vulnerability demonstrates real practical skill in a way no certificate can replicate
- **Write security-focused code reviews:** Take an open-source project on GitHub. Run Semgrep against it. Find a potential vulnerability. Write up your analysis and open a responsible disclosure issue. Document it
- **Build a DevSecOps pipeline:** Create a public GitHub repository. Integrate SAST, SCA and secret scanning into a CI/CD pipeline. Show the pipeline catching intentional vulnerabilities. This is a powerful portfolio piece for DevSecOps roles
- **Write about what you learn:** Blog posts explaining SQL injection, XSS, or IDOR in plain language with code examples demonstrate both technical understanding and the communication skill AppSec demands

Where to Look	Notes
CyberSecurityJobs.com	Filter by 'AppSec engineer', 'product security', 'secure developer', 'DevSecOps'
LinkedIn	Tech and product companies post AppSec roles heavily here. Connect with AppSec engineers at companies you admire
HackerOne / Bugcrowd careers pages	Companies running bug bounty programmes are security-conscious and often hire from the bounty community
Snyk / Semgrep / PortSwigger job boards	Security tooling companies themselves hire developer-minded AppSec engineers
Tech company security teams (Google, Meta, Monzo, Revolut)	Some of the best AppSec roles are at product companies — they pay well and the work is embedded in real development
Specialist consultancies (NCC Group, WithSecure, Trustwave)	Consulting AppSec roles offer breadth — testing many different applications and technology stacks

Table 11: Where to find application security roles

Interview questions to prepare for

- What is SQL injection and how would you test for it in a black-box scenario?
- Explain the difference between XSS stored, reflected and DOM-based.
- What is IDOR and how do you identify it during a code review?
- Walk me through your SAST triage process — how do you decide if a finding is a real vulnerability?

- A developer tells you that they sanitise user input before using it in a SQL query. What follow-up questions do you ask?
- What security headers should every web application return and what does each one do?
- How would you integrate security into a CI/CD pipeline from scratch?
- What is the OWASP ASVS and how would you use it in an AppSec programme?

The best AppSec interview answer:
"Here is my PortSwigger profile. Here is a bug I found on HackerOne."
Demonstrated practical skill consistently beats theoretical knowledge alone.

15. References

1. CW Jobs (2024) *Technology Salary Survey 2024.* Available at: https://www.cwjobs.co.uk/salary-checker [Accessed: 10 April 2026].
2. GIAC (2024) *GWEB — GIAC Web Application Defender.* Available at: https://www.giac.org/certifications/web-application-defender-gweb/ [Accessed: 10 April 2026].
3. INE / eLearnSecurity (2024) *eWPT — eLearnSecurity Web Application Penetration Tester.* Available at: https://ine.com/learning/certifications/internal/elearnsecurity-web-application-penetration-tester [Accessed: 11 April 2026].
4. ISC2 (2024) *CSSLP — Certified Secure Software Lifecycle Professional.* Available at: https://www.isc2.org/certifications/csslp [Accessed: 11 April 2026].
5. Microsoft (2024) *Threat Modelling — Microsoft Security Development Lifecycle.* Available at: https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling [Accessed: 11 April 2026].
6. Offensive Security (2024) *OSWE — Offensive Security Web Expert.* Available at: https://www.offensive-security.com/awae-oswe/ [Accessed: 12 April 2026].
7. OWASP Foundation (2021) *OWASP Top Ten 2021.* Available at: https://owasp.org/Top10/ [Accessed: 12 April 2026].
8. OWASP Foundation (2023) *OWASP Application Security Verification Standard (ASVS) v4.0.* Available at: https://owasp.org/www-project-application-security-verification-standard/ [Accessed: 12 April 2026].
9. OWASP Foundation (2024) *OWASP Threat Modelling Cheat Sheet.* Available at: https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html [Accessed: 13 April 2026].
10. PortSwigger (2024) *Web Security Academy — Free Online Web Security Training.* Available at: https://portswigger.net/web-security [Accessed: 13 April 2026].
11. PortSwigger (2024) *Burp Suite Certified Practitioner.* Available at: https://portswigger.net/web-security/certification [Accessed: 13 April 2026].
12. Reed (2024) *Cybersecurity Salary Guide UK 2024.* Available at: https://www.reed.co.uk/career-advice/cybersecurity-salary [Accessed: 13 April 2026].
13. Shostack, A. (2014) *Threat Modeling: Designing for Security.* Indianapolis: Wiley.
14. Verizon (2023) *Data Breach Investigations Report 2023.* Available at: https://www.verizon.com/business/resources/reports/dbir/ [Accessed: 14 April 2026].

Document prepared by **Babashaheer**. Version 1.0 — April 2026. Cybersecurity Career Series — Document 06 of 09.