



COMMON SECURITY PITFALLS

A Software Engineer's Perspective
Lochie Ashcroft

\$ whoami

Computer Science

- Strong fundamentals
- Did well in CTFs during uni
- Prefer designing (defence) over attacking

IBM Security - R&D Lab

- Identity and Access Management (Verify SaaS)
- Software Engineer (Intern → Mid)

Stake - Investment Platform

- Software Engineer (Product, Platform)
- Site Reliability Engineer



**What this talk is
not**

ATT&CK Matrix for Enterprise

layout: flat ▾ show sub-techniques hide sub-techniques

Reconnaissance 11 techniques	Resource Development 8 techniques	Initial Access 11 techniques	Execution 17 techniques	Persistence 23 techniques	Privilege Escalation 14 techniques	Defense Evasion 47 techniques	Credential Access 17 techniques	Discovery 34 techniques	Lateral Movement 9 techniques	Collection 17 techniques	Command and Control 18 techniques	Exfiltration 9 techniques	Impact 15 techniques
Active Scanning (3)	Acquire Access	Content Injection	Cloud Administration Command	Account Manipulation (7)	Abuse Elevation Control Mechanism (4)	Abuse Elevation Control Mechanism (4)	Adversary-In-the-Middle (4)	Account Discovery (4)	Exploitation of Remote Services	Adversary-In-the-Middle (4)	Application Layer Protocol (3)	Automated Exfiltration (1)	Account Access Removal
Gather Victim Host Information (4)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (12)	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Brute Force (4)	Application Window Discovery	Internal Spearphishing	Archive Collected Data (3)	Communication Through Removable Media	Data Transfer Size Limits	Data Destruction (1)
Gather Victim Identity Information (2)	Compromise Accounts (2)	Exploit Public-Facing Application	Container Administration Command	Boot or Logon Autostart Execution (14)	Account Manipulation (7)	Account Manipulation (7)	Credentials from Password Stores (4)	Browser Information Discovery	Lateral Tool Transfer	Content Injection	Content Injection	Exfiltration Over Alternative Protocol (2)	Data Encrypted for Impact
Gather Victim Network Information (4)	Compromise Infrastructure (8)	External Remote Services	Deploy Container	Boot or Logon Initialization Scripts (3)	Boot or Logon Initialization Scripts (3)	Build Image on Host	Debugger Evasion	Cloud Infrastructure Discovery	Remote Service Session Hijacking (2)	Automated Collection	Data Encoding (2)	Exfiltration Over C2 Channel	Data Manipulation (2)
Gather Victim Org Information (4)	Establish Accounts (2)	Hardware Additions	ESXi Administration Command	Cloud Application Integration	Event Triggered Execution (14)	Delay Execution	Deobfuscate/Decode Files or Information	Cloud Service Dashboard	Remote Services (8)	Browser Session Hijacking	Data Obfuscation (3)	Exfiltration Over Other Network Medium (1)	Defacement (2)
Phishing for Information (4)	Obtain Capabilities (7)	Replication Through Removable Media	Exploitation for Client Execution	Compromise Host Software Binary	Create or Modify System Process (5)	Deploy Container	Domain or Tenant Policy Modification (2)	Cloud Service Discovery	Replication Through Removable Media	Clipboard Data	Dynamic Resolution (3)	Exfiltration Over Physical Medium (1)	Disk Wipe (2)
Search Closed Sources (2)	Stage Capabilities (6)	Supply Chain Compromise (3)	Input Injection	Create Account (3)	Domain or Tenant Policy Modification (2)	Direct Volume Access	Domain or Tenant Policy Modification (2)	Container and Resource Discovery	Software Deployment Tools	Data from Cloud Storage	Encrypted Channel (2)	Exfiltration Over Web Service (4)	Email Bombing
Search Open Technical Databases (3)	Trusted Relationship	Valid Accounts (4)	Inter-Process Communication (2)	Create or Modify System Process (3)	Event Triggered Execution (14)	Escape to Host	Event Triggered Execution (14)	Debugger Evasion	Taint Shared Content	Data from Configuration Repository (2)	Fallback Channels	Scheduled Transfer	Endpoint Denial of Service (4)
Search Open Websites/Domains (3)	Wi-Fi Networks	Poisoned Pipeline Execution	Native API	Exclusive Control	Exploitation for Privilege Escalation	Email Spoofing	Multi-Factor Authentication Interception	Device Driver Discovery	File and Directory Discovery	Data from Information Repositories (6)	Hide Infrastructure	Transfer Data to Cloud Account	Financial Theft
Search Threat Vendor Data		Scheduled Task/Job (5)	Poisoned Pipeline Execution	External Remote Services	Hijack Execution Flow (12)	Execution Guardrails (2)	Multi-Factor Authentication Request Generation	Domain Trust Discovery	Group Policy Discovery	Data from Local System	Ingress Tool Transfer		Firmware Corruption
Search Victim-Owned Websites		Serverless Execution	Shared Modules	Implant Internal Image	Process Injection (12)	Exploitation for Defense Evasion	Network Sniffing	File and Directory Permissions Modification (2)	Local Storage Discovery	Data from Network Shared Drive	Non-Application Layer Protocol		Network Denial of Service (2)
		Software Deployment Tools	System Services (3)	Modify Authentication Process (6)	Scheduled Task/Job (5)	File and Directory Permissions Modification (2)	OS Credential Dumping (4)	Hide Artifacts (14)	Log Enumeration	Data from Removable Media	Non-Standard Port		Resource Hijacking (4)
		User Execution (3)	User Execution (3)	Modify Registry	Valid Accounts (4)	Hijack Execution Flow (12)	Steal Application Access Token	Impair Defenses (12)	Network Service Discovery	Data Staged (2)	Protocol Tunneling		Service Stop
		Windows Management Instrumentation	Power Settings	Office Application Startup (6)	Pre-OS Boot (3)	Indicator Removal (10)	Steal or Forge Authentication Certificates	Impersonation	Network Share Discovery	Email Collection (3)	Proxy (4)		System Shutdown/Reboot
			Power Settings	Pre-OS Boot (3)	Scheduled Task/Job (3)	Indirect Command Execution	Steal Web Session Cookie	Permissions Groups Discovery (3)	Password Policy Discovery	Input Capture (4)	Remote Access Tools (2)		
			Power Settings	Scheduled Task/Job (3)	Server Software Component (6)	Masquerading (12)	Unsecured Credentials (8)	Process Discovery	Peripheral Device Discovery	Screen Capture	Web Service (3)		
			Power Settings	Software Extensions (2)	Traffic Signaling (2)	Modify Authentication Process (4)	Modify Cloud Compute Infrastructure (3)	Query Registry	System Information Discovery	Video Capture			
			Power Settings	Valid Accounts (4)	Valid Accounts (4)	Modify Cloud Resource Hierarchy	Modify Registry	Remote System Discovery	System Location Discovery (1)				
			Power Settings			Modify Registry	Modify System Image (2)	Software Discovery (2)	System Network Configuration Discovery (2)				
			Power Settings			Obfuscated Files or Information (17)	Network Boundary Bridging (1)	System Network Connections Discovery	System Owner/User Discovery				
			Power Settings			Plist File Modification	Pre-OS Boot (3)	System Service Discovery	System Time Discovery				
			Power Settings			Process Injection (12)	Reflective Code Loading	Virtual Machine Discovery	Virtualization/Sandbox Evasion (3)				



TOP 10

The Ten Most Critical Web Application Security Risks

Introduction

Welcome to the 8th installment of the OWASP Top Ten!

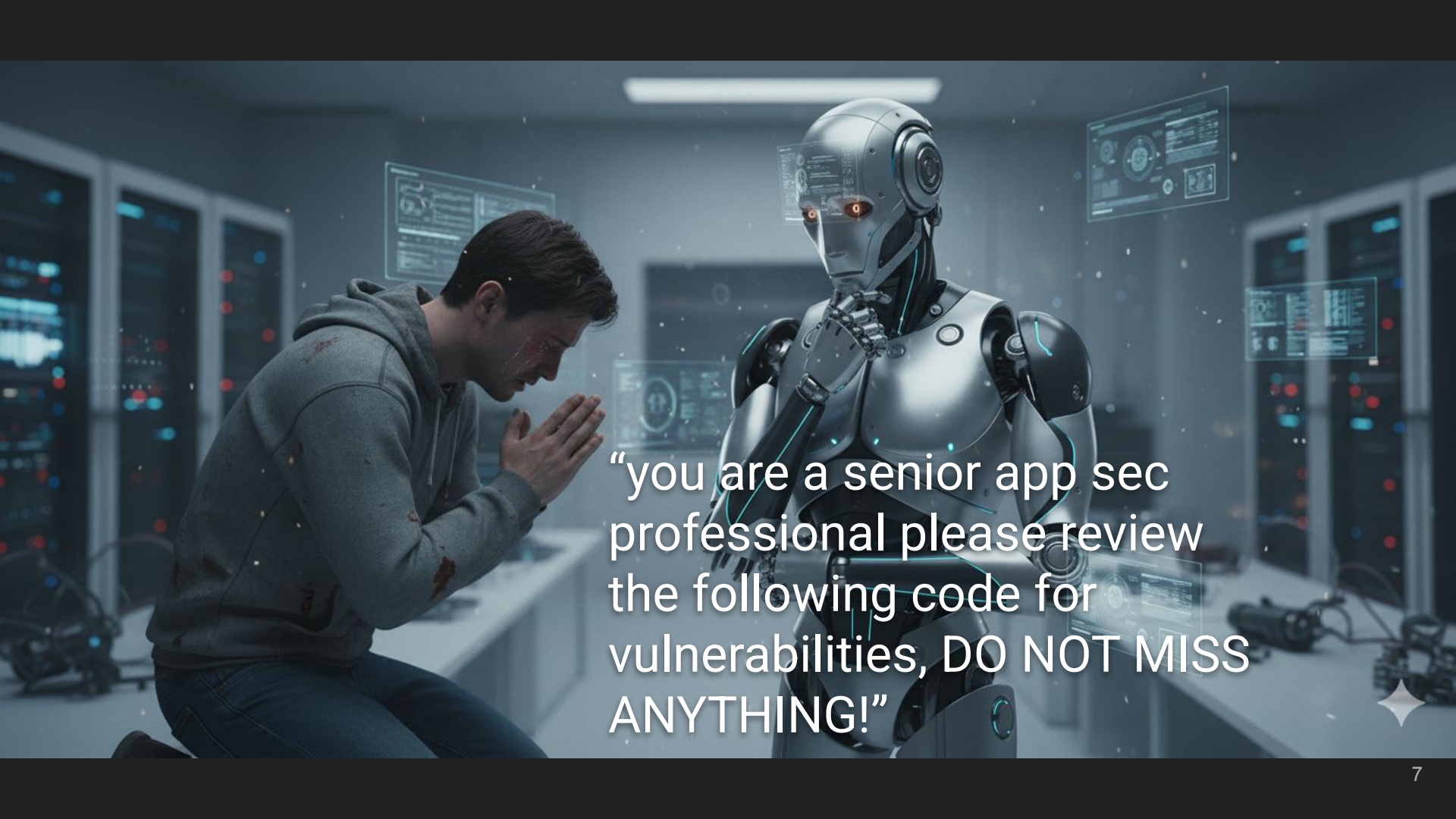
A huge thank you to everyone who contributed data and perspectives in the survey. Without you, this installment would not have been possible. **THANK YOU!**

Introducing the OWASP Top 10:2025

- [A01:2025 - Broken Access Control](#)
- [A02:2025 - Security Misconfiguration](#)
- [A03:2025 - Software Supply Chain Failures](#)
- [A04:2025 - Cryptographic Failures](#)
- [A05:2025 - Injection](#)
- [A06:2025 - Insecure Design](#)
- [A07:2025 - Authentication Failures](#)
- [A08:2025 - Software or Data Integrity Failures](#)
- [A09:2025 - Security Logging & Alerting Failures](#)
- [A10:2025 - Mishandling of Exceptional Conditions](#)

AI

?



“you are a senior app sec professional please review the following code for vulnerabilities, DO NOT MISS ANYTHING!”

What this talk is

Types of Engineers

Throughput Optimizer

Metric: Ticket velocity and sprint completion.

Philosophy: "Done is better than perfect."

Constraint: Avoiding blockers and minimizing scope creep.

Sustainability Optimizer

Metric: System uptime and long-term maintainability.

Philosophy: "Build it once, build it right."

Constraint: Balancing technical debt against feature requests.

Corporate Reality

Company Stage	Primary Driver	Security Posture
Early Stage	Survival & Product-Market Fit	" Security is Friction " Speed to market is the only metric that ensures the next round of funding.
Growth	Market Capture	" The Complexity Gap " Scaling features faster than the underlying security infrastructure can support.
Mature / Public	Risk & Compliance	" Defensive Investment " Security is often driven by audit requirements and legal liability rather than proactive engineering.

The Incentive Gap

Invisible Work

Feature wins are celebrated in All-Hands meetings; "the breach that never happened" is rarely noticed.

The Cost of Friction

If security tools add 20% more time to a deployment, the "Throughput Optimizer" will naturally find workarounds.

Asymmetric Risk

The individual developer often bears the "speed pressure", while the company bears the "security risk".

A photograph showing the aftermath of a disaster. In the foreground, a large, chaotic pile of rubble, including concrete blocks, rebar, and debris, dominates the left and center. To the right, a two-story building with large glass windows is partially destroyed, with its upper floor missing and its structure exposed. In the background, a tall, dark smokestack stands against a pale, hazy sky. The overall scene conveys a sense of devastation and the consequences of failure.

**“If you fail to plan,
you are planning to
fail”**

Benjamin Franklin

Design Document

- What's the problem
- Potential Solutions
- Assumptions
- Constraints
- Architecture
- Flows
- **Security**
- ...



Security - Threat Model

What you are protecting from whom

- Assets
- Attackers
- Entry Points
- Trust Boundaries



Security - Requirements

Determined by business and regulatory needs

- Authentication and authorization standards
- Data encryption in transit and at rest
- Logging and monitoring requirements
- Secure error handling







Defensive Programming - Super Simple Example

```
1  
2  def divide(x, y):  
3      return x / y
```

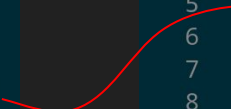
Defensive Programming - Solid Divide

```
1  from typing import Union
2
3  Number = Union[int, float]
4
5
6  def divide(x: Number, y: Number) -> float:
7      """Divide two numbers.
8
9      Args:
10         x (Number): the numerator
11         y (Number): the denominator
12
13     Raises:
14         TypeError: if either input is not a number
15         ValueError: if the denominator is zero
16
17     Returns:
18         float: the result of the division
19     """
20     if not isinstance(x, (int, float)) or not isinstance(y, (int, float)):
21         raise TypeError("Both inputs must be numbers (int or float).")
22
23     if y == 0:
24         raise ValueError("Denominator cannot be zero.")
25
26     return x / y
27
```

Defensive Programming - Running

```
28
29 a, b = 10, 2
30 print(divide(a, b))
31
32 c, d = "10", 2
33 print(divide(c, d))
34
```

```
1 $ python3 defense-programming.py
2 5.0
3 Traceback (most recent call last):
4   File "/SOME/PATH/script.py", line 33, in <module>
5     print(divide(c, d))
6     ^^^^^^^^^^^^^^^
7   File "/SOME/PATH/script.py", line 21, in divide
8     raise TypeError("Both inputs must be numbers (int or float).")
9   TypeError: Both inputs must be numbers (int or float).
10
```



Defensive Programming - Static Type Analysis

```
1 $ mypy script.py
2 script.py:33: error: Argument 1 to "divide" has incompatible type "str"; expected "int | float" [arg-type]
3 Found 1 error in 1 file (checked 1 source file)
```



- > write code...
- > git add script.py
- > git commit -m > "implement divide"
- > git push
- > deploy to prod
- > eat lunch



- > write code
- > how could it break?
- > think...
- > write tests...
- > write tests...
- > run tests
- > git add script.py
- > git commit -m > "impl
divide"
- > git push
- > review
- > deploy to prod

```
1  class DivideTestCase(unittest.TestCase):
2      def test_positive_numbers(self):
3          self.assertEqual(divide(10, 2), 5)
4          self.assertEqual(divide(3.14, 1.57), 2.0)
5
6      def test_negative_numbers(self):
7          self.assertEqual(divide(-10, 2), -5)
8          self.assertEqual(divide(10, -2), -5)
9
10     def test_decimal_results(self):
11         self.assertAlmostEqual(divide(5, 2), 2.5)
12         self.assertAlmostEqual(divide(1, 3), 0.333333, places=6)
13
```


Defenseless Programming - Happy Path Only

- Didn't consider what could go wrong
 - No input validation
 - No response validation
 - Limited error handling
 - Lack of Authorization
 - Tests ?
-
- LGTM! 👉 🚀





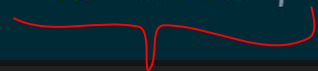
Input Validation

Defenseless Programming - Input Validation - SQLi

```
-- Bad: concatenating user input directly
SELECT * FROM users
WHERE username = 'admin' AND password = 'password';

-- Attacker supplies:
-- username = admin
-- password = ' OR '1'='1'

-- Resulting query (insecure string concatenation):
SELECT * FROM users
WHERE username = 'admin' AND password = '' OR '1'='1';
```



Defenseless Programming - Input Validation - SQLi

```
// Node.js (vulnerable) - naive builder that concatenates user input  
app.get('/users', (req, res) => {  
  // optional query params: ?name=&age=&role=  
  const { name, age, role } = req.query;  
  
  // Start with always-true base so we can append "AND ..." or "OR ..." easily  
  let sql = "SELECT * FROM users WHERE 1=1";  
  
  if (name) sql += " OR name = '" + name + "'";  
  if (age)   sql += " OR age = " + age;  
  if (role) sql += " OR role = '" + role + "'";  
  
  // run raw query (vulnerable)  
  db.query(sql).then(rows => res.json(rows));  
});
```

Defenseless Programming - Timing Attacks

```
app.post("/login", async (req, res) => {
  const { email, password } = req.body;

  const user = await db.users.findOne({ email: email });

  if (!user) {
    return res.status(401).json({
      error: "Invalid email or password"
    });
  }

  const passwordMatch = await bcrypt.compare(password, user.passwordHash);

  if (!passwordMatch) {
    return res.status(401).json({
      error: "Invalid email or password"
    });
  }

  res.json({ message: "Login successful" });
});
```

Defenseless Programming - Timing Attacks

```
const DUMMY_HASH = "$2b$10$KbQiY2h3p2q2S3N6r5uQ0e8vKZcQ1qKjY8xj2mJq1b3sZ4Kp6dN9a";

app.post("/login", async (req, res) => {
  const { email, password } = req.body;

  const user = await db.users.findOne({ email: email });

  const hash = user ? user.passwordHash : DUMMY_HASH;

  const passwordMatch = await bcrypt.compare(password, hash);

  if (!user || !passwordMatch) {
    return res.status(401).json({
      error: "Invalid email or password"
    });
  }

  res.json({ message: "Login successful" });
});
```

Defenseless Programming - Input Validation - Directory Traversal

Endpoint: foo.com/view?file={filename}

Defenseless Programming - Input Validation - Directory Traversal

Normal Request

```
curl "foo.com/view?file=report.txt"
```

Malicious Request

```
curl "foo.com/view?file=../../../../etc/passwd"
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
...
```

Defenseless Programming - Input Validation - Encoding

Character	URL Encoded	Double Encoded	Common Use in Attacks
/	%2F	%252F	Path traversal
.	%2E	%252E	Combined with /
?	%3F	%253F	Query string manipulation
&	%26	%2526	Fake query params
%	%25	%2525	Enables double encoding
<	%3C	%253C	Script tags
>	%3E	%253E	Script Tags
;	%3B	%253B	Command chaining, shell injection

Scientific Notation

[illegible]

Defenseless Programming - Input Validation - Java BigDecimal

```
public static class InputData {  
    public BigDecimal value;  
  
    @Override  
    public String toString() {  
        return "InputData{value=" + value + '}';  
    }  
}
```

Defenseless Programming - Input Validation - Java BigDecimal

```
ObjectMapper mapper = new ObjectMapper();  
InputData data = mapper.readValue(jsonString, InputData.class);
```

Defenseless Programming - Input Validation - Java BigDecimal

```
...
int digits = 1000000;

StringBuilder sb = new StringBuilder(digits + 16);
sb.append("{ \"value\": ");

for (int i = 0; i < digits; i++)
    sb.append('9');

sb.append("}");

String chosen = sb.toString();
...
```

Defenseless Programming - Input Validation - Java BigDecimal

Duration: 8585 MS

Memory : 226 MB

Defenseless Programming - Input Validation - Java BigDecimal

```
com.fasterxml.jackson.core.exc.StreamConstraintsException -  
Number length (1000000) exceeds the maximum length (1000)
```

Patched: v2.15.0 (April 2023)

Defenseless Programming - Input Validation - Java BigDecimal

Other Mitigations

- Create custom deserialization logic to validate before mapping
- Use JSON schemas to validate data shape before mapping
- Use appropriate data types for business logic
- Limit request payload size

Defenseless Programming - Input Validation - Regex

Regex Pattern: `^(a+)+$`

`^` – Start of the string

`(a+)` – One or more a characters, grouped

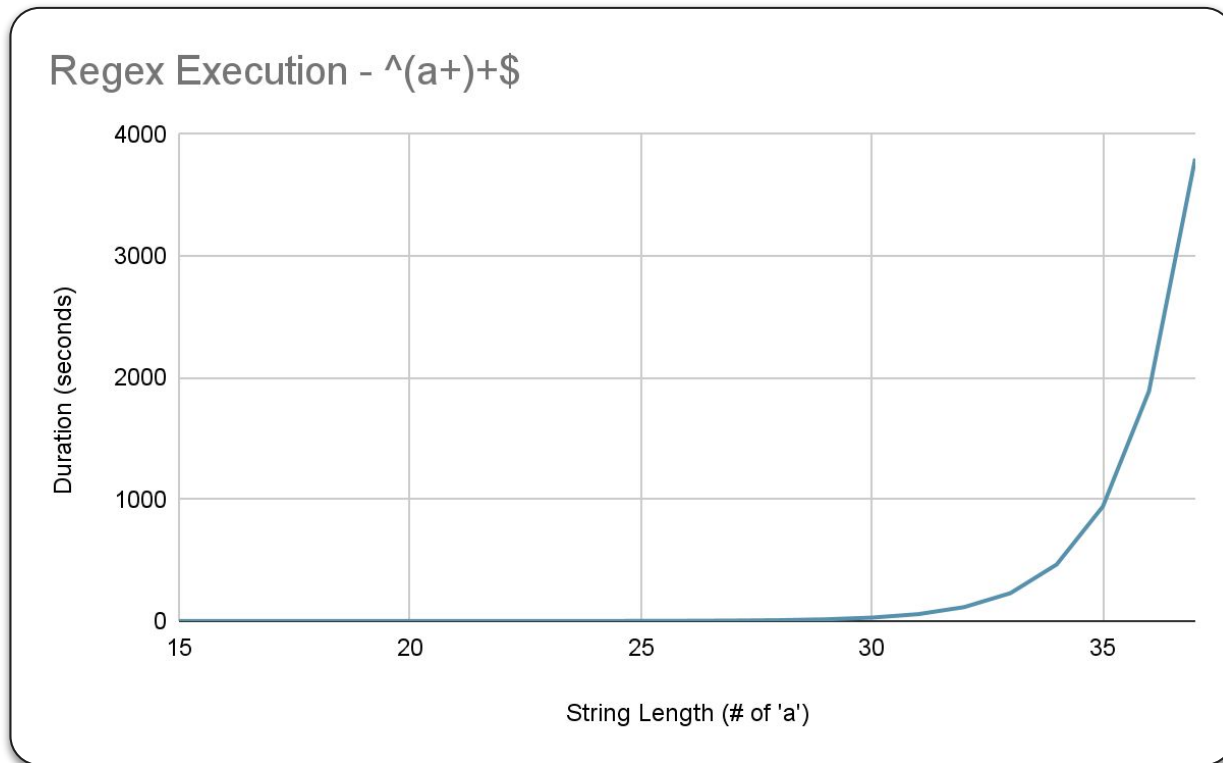
`(a+)+` – The group itself is repeated one or more times

`$` – End of the string

"a"	✓
"aa"	✓
"aaa"	✓
"ab"	✗
"ba"	✗
""	✗

Malicious input: `aaa...aaaX`

Defenseless Programming - Input Validation - Regex



Defenseless Programming - Input Validation - Regex

Fixed Regex Pattern: `^a+$`

- Removes the nesting
- Support a subset of regex for user supplied patterns
- Put a time bound around regex evaluations



Response Validation

Response Body

Server Error in '/' Application.

Maximum request length exceeded.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Web.HttpException: Maximum request length exceeded.

Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:

```
[HttpException (0x80004005): Maximum request length exceeded.]
  System.Web.HttpRequest.GetEntireRawContent() +9899668
  System.Web.HttpRequest.GetMultipartContent() +63
  System.Web.HttpRequest.FillInFormCollection() +160
  System.Web.HttpRequest.EnsureForm() +69
  System.Web.HttpRequest.get_Form() +13
  System.Web.HttpRequestWrapper.get_Form() +14
  System.Web.Mvc.HttpRequestExtensions.GetHttpMethodOverride(HttpRequestBase request) +121
  System.Web.Mvc.AcceptVerbsAttribute.IsValidForRequest(ControllerContext controllerContext, MethodInfo methodInfo) +37
  System.Web.Mvc.HttpPostAttribute.IsValidForRequest(ControllerContext controllerContext, MethodInfo methodInfo) +39
```

Response Headers

- Modify
 - Server
 - X-Powered-By
 - ...
- Include
 - Content-Security-Policy
 - CORS
 - ...

▼ General	
Request URL	https://www.bom.gov.au/qld/
Request Method	GET
Status Code	200 OK
Remote Address	23.40.73.19:443
Referrer Policy	strict-origin-when-cross-origin
▼ Response Headers ☐ Raw	
Accept-Ranges	bytes
Connection	keep-alive
Content-Encoding	gzip
Content-Length	9738
Content-Type	text/html; charset=UTF-8
Date	Mon, 06 Oct 2025 10:00:46 GMT
Server	Apache
Server-Timing	cdn-cache; desc=HIT
Server-Timing	edge; dur=1
Server-Timing	ak_p; desc="1759744846744_3879"
Set-Cookie	bm_sv=E17058F5DF27AC3ED33E885uSrFc/DtoNIMcglV+tfYPILGfff2pXAqunvc5InfjQeH8j6DnlY6fTYdqsI0H11:44:05 GMT; Max-Age=6199; Sec
Vary	Accept-Encoding
X-Akamal-Transformed	9 - 0 pmb=mRUM,1

Defenseless Programming - Lack of Authorization (IDOR)

GET: /orders/{ORDER_ID}

GET: /orders/4589d301-7f73-45e2-bbbe-e111a99b4fae



```
app.get('/api/orders/:orderId', async (req, res) => {  
  const { orderId } = req.params;  
  
  try {  
    const order = await db.getOrder(orderID);  
  
    if (!order) {  
      return res.status(404).json({ message: 'Order not found' });  
    }  
  
    res.json(order);  
  } catch (err) {  
    console.error(err);  
    res.status(500).json({ message: 'Internal server error' });  
  }  
});
```

```
app.get('/api/orders/:orderId', async (req, res) => {  
  const { orderId } = req.params;  
  
  try {  
    const order = await db.getUserOrder(req.user.id, orderId);  
  
    if (!order) {  
      return res.status(404).json({ message: 'Order not found' });  
    }  
  
    res.json(order);  
  } catch (err) {  
    console.error(err);  
    res.status(500).json({ message: 'Internal server error' });  
  }  
});
```



Secret Management

GitHub expands security tools after 39 million secrets leaked in 2024

By **Bill Toulas**



April 2, 2025



02:24 PM



0

GitHub announced updates to its Advanced Security platform after it detected over 39 million leaked secrets in repositories during 2024, including API keys and credentials, exposing users and organizations to serious security risks.

In a new report by GitHub, the development company says the 39 million secrets were found through its [secret scanning service](#), a security feature that detects API keys, passwords, tokens, and other secrets in repositories.

"Secret leaks remain one of the most common—and preventable—causes of security incidents," [reads GitHub's announcement](#).

"As we develop code faster than ever previously imaginable, we're leaking secrets faster than ever, too."

Secret Management - DO NOT HARDCODE!

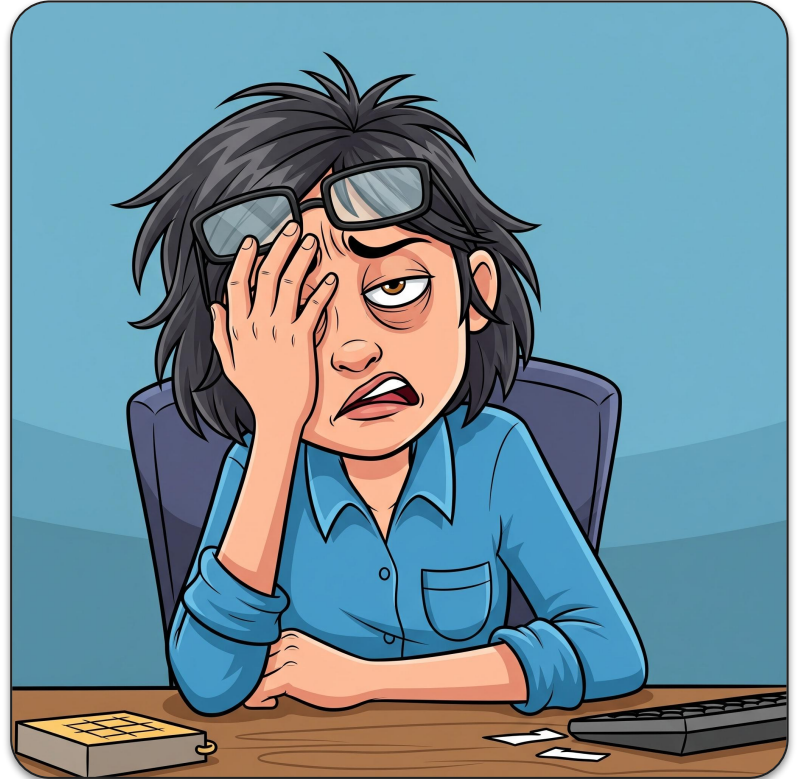
```
1  API_KEY = "c76258305d354beeb1c8ba676c12a4c3"
```

Secret Management - .env files

```
1  # .env.local
2  API_KEY=c76258305d354beeb1c8ba676c12a4c3
3  # .env.dev
4  API_KEY=c76258305d354beeb1c8ba676c12a4c3
5  # .env.staging
6  API_KEY=ce0fd20f75254c93a4e8856f68879338
7  # .env.production
8  API_KEY=72c96c2d25814391854cff64fb2c49ef
```

Secret Management - .env files

- They are still in your VCS...
- Add to .gitignore
- Still on filesystem...
- Still in history...

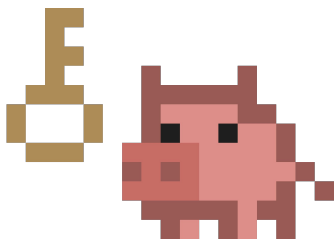


Secrets Management - .env

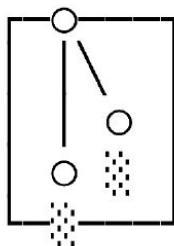
- Add logic to resolve the secrets in your application
- orchestrate startup
- Password managers
- Secret managers
 - Hashicorp Vault
 - AWS SSM Parameter store
 - GCP Secret Manager
 - etc

Secrets Management - VCS continuous evaluation

- Run secrets scanning as a pre-commit hook
 - Prevents secrets being committed and can be resolved quickly and easily
- Run secrets scanning on push (CI)
 - Secret has now been leaked but at least now it has been reported



Trufflehog



Gitleaks



Semgrep

Secrets Management - Git Cleanup

- Secrets have been pushed and now are leaked
 - Can't just update the file, it's in git history...
- Clean the git history
 - Get output from secret scanning tools
 - Use BFG to forcefully modify the git history

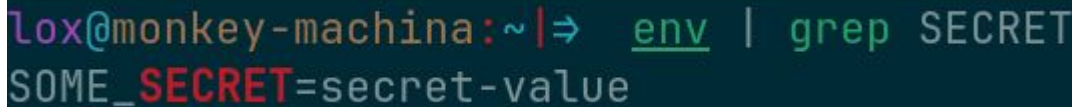


Secret Management - Rotate your secrets!

- Don't wait for a leak
 - Don't wait for an employee to leave
 - Bake rotation into your process from day 1
 - Periodically rotate secrets
-
- Treat rotation as a insurance policy

Secrets Management - Shell profiles

- Retrieving secrets can be annoying
 - I'll export them in '.bashrc' so I never need to think about it!
 - Available for all my projects!
- Lots of stealers out there....
 - npm post install
 - pip
 - Dependencies
 - Malware

A terminal window with a dark background. The prompt is 'loox@monkey-machina:~'. The command 'env | grep SECRET' is entered. The output shows 'SOME_SECRET=secret-value', where 'SECRET' is highlighted in red.

```
loox@monkey-machina:~|⇒ env | grep SECRET
SOME_SECRET=secret-value
```

Secrets Management - Shell History

```
lox@monkey-machina:~|⇒ API_KEY=abc123 python3 app.py
```

```
lox@monkey-machina:~|⇒ grep "API_KEY" ~/.zsh_history  
: 1748338636:0;API_KEY=abc123 python3 app.py
```

```
lox@monkey-machina:~|⇒ API_KEY=abc123 python3 app.py
```

Software Supply Chain



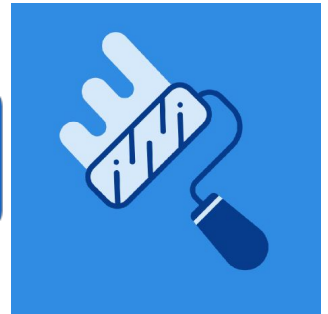
Software Supply Chain - Versioning

- Pinning to certain versions (v1.2.3)
 - If versions in the package manager are immutable
- Git commit (16ecab1875791e2b6ed0c9a6dae5a12a79d92120e1c3afbd3a9c8535ce44666d)
 - If package manager uses git tags
- Vendoring
 - Committing your dependencies
- Lock files
- Internal registries / mirrors



Software Supply Chain - Staying up to date

- Dependabot
- Renovate
- Follow CI/CD and automate patching



Software Supply Chain - Container base images

- It's a container not a VM!
 - Do you need the bloat?
 - Do you need an OS?
 - Do you need a shell?
 - Multi stage build?
- scratch
- Alpine
- Google distroless
- Redhat Universal Base Image
- `{language}:slim`
- Chainguard images

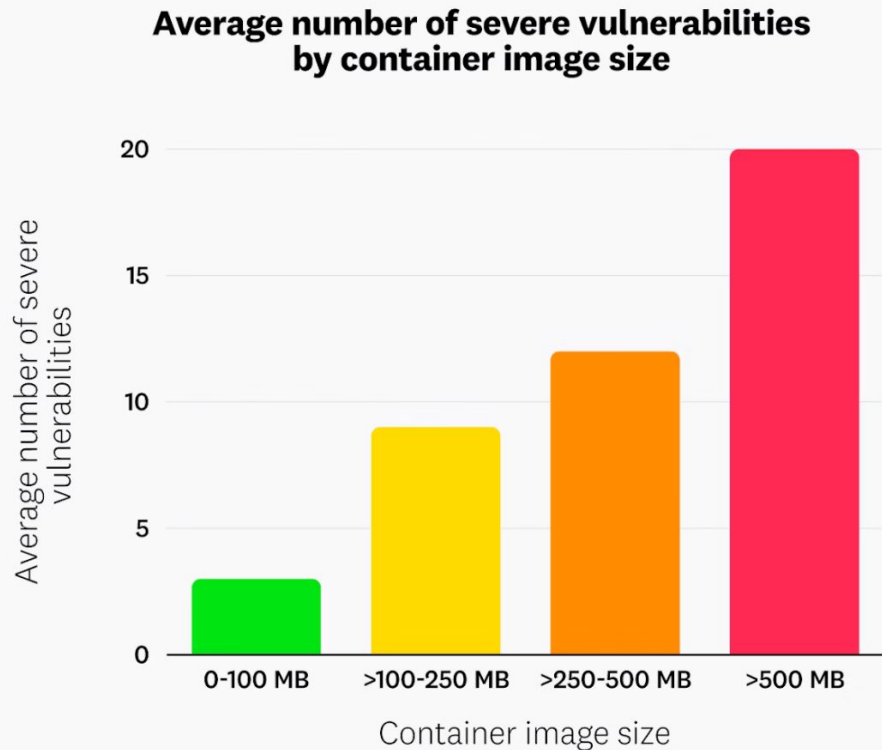


Software Supply Chain - Multi Stage Builds

Golang "Hello World!"	
Base Image	Resulting Image Size
golang:1.24.3-alpine3.20	308 MB
gcr.io/distroless/base-debian12	24.2 MB
scratch	2.21 MB
Raw binary	2.1 MB

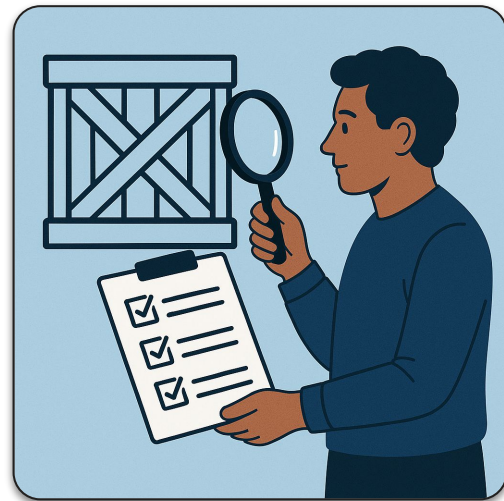
USING SMALLER CONTAINER IMAGES LEADS TO FEWER VULNERABILITIES

Source: Datadog

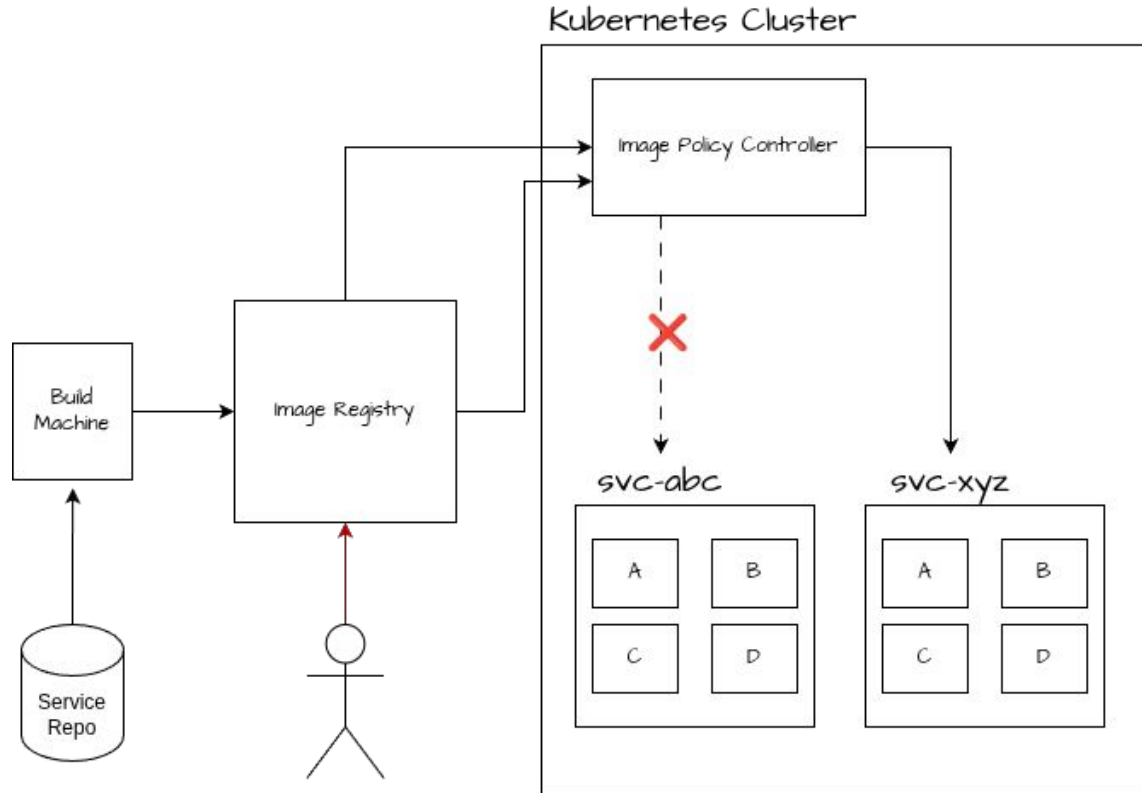


Software Supply Chain - Image Provenance and Attestation

- Sign images at build time
- Include metadata / labels
 - Compliance scans result
 - SAST
 - SBOM
 - Test coverage
- Reject the deployment if requirements are not met



Software Supply Chain - Image Provenance and Attestation





PII - Don't log it!

```
1 2025-05-25 16:00:40, 896 - __main__ - INFO - User did action:
2 User(id='266befc3-9d7d-49a4-acce-987df8514064',
3     first_name='Olivia',
4     middle_names=None,
5     last_name='Kendrick',
6     email='olivia.kendrick@example.com')
```



PII - Don't log it!

```
1 def mask(value: Any) -> str:
2     if not value:
3         return "None"
4
5     raw = str(value)
6     return raw[0] + "*" * 3
```

```
1 @dataclass
2 class User:
3     id: str
4     first_name: str
5     middle_names: Optional[str]
6     last_name: str
7     email: str
8
9     def __str__(self):
10         return (
11             f"User[id: {self.id}, "
12             f"first_name: {mask(self.first_name)}, "
13             f"middle_names: {mask(self.middle_names)}, "
14             f"last_name: {mask(self.last_name)}, "
15             f"email: {mask(self.email)}]"
16         )
```

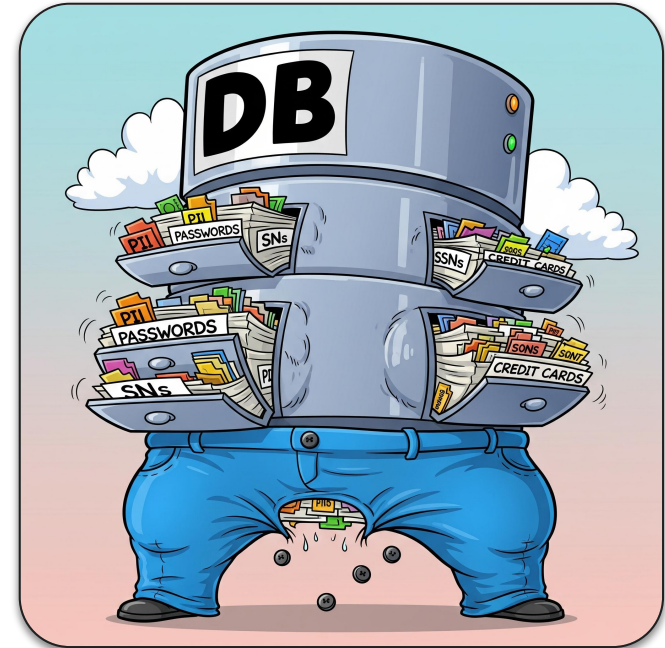
PII - Don't log it!

```
1 2025-05-25 16:02:57, 555 - __main__ - INFO - User did action:
2 User[id: f34a3603-22ac-445d-978b-e7a450ca9672,
3   first_name: 0***,
4   middle_names: None,
5   last_name: K***,
6   email: o***]
```



PII - DB Access

- Do not share credentials
- Service accounts
- temporary user accounts
 - Scoped
 - Time bound



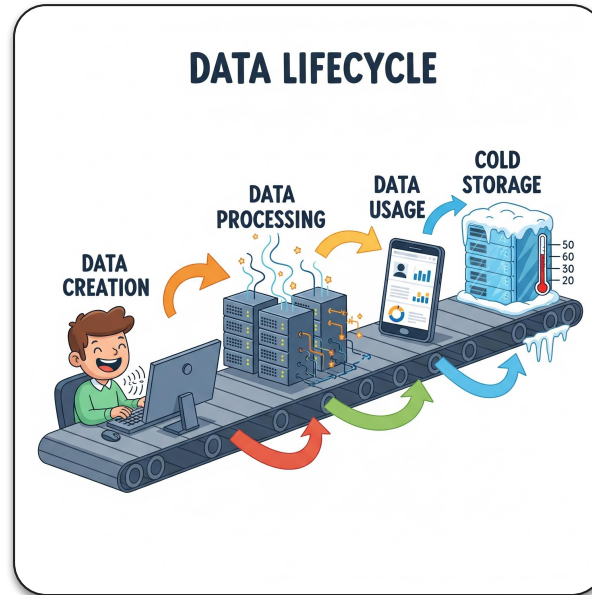
PII - DB Audit Logs

- Enable it!
- Security and Compliance
- Incident Response



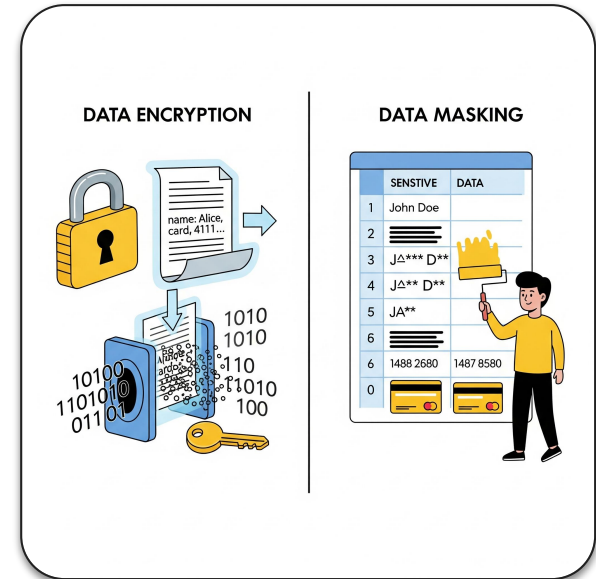
PII - Data Lifecycle

- Do you still need the data?
- Can you put the data somewhere else?



PII - Encryption or Data Masking

- Encrypt what you can
 - Hardware based if possible
 - Use a KMS from a cloud provider
 - Disk Encryption protects you from physical access
- Mask what you can
 - Names, emails, DoB, etc.
 - Just in time masking via proxies (identity aware)



Summary

- Spoke about a lot!
- Developing Software is hard
- Developing secure Software is harder
- Training is crucial
- Developers are incentivized to ship features



LachlanAshcroft.com/talk