Fundamentos de DevOps

Aula 10 – DevSecOps

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

linkedin.com/in/guijac

Roteiro

- Pilares da Segurança da Informação;
- DevSecOps:
  - Contexto Histórico;
  - Definição.
- Atuais Desafios;
- Importância do Desenvolvimento Seguro;
- Top 10 OWASP;
- Outros Casos Recentes;
- Mudanças Necessárias:
  - Pessoas;
  - Processos;
  - Tecnologias.
- Referências Bibliográficas.

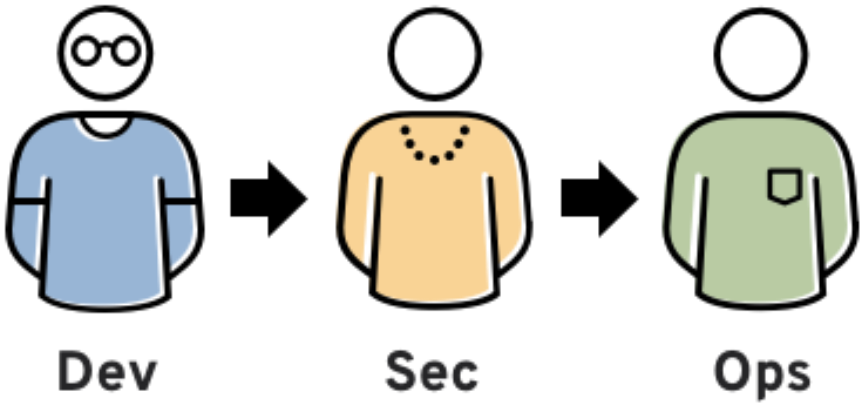
Pilares da Segurança da Informação

- **Confidencialidade**: informações sigilosas devem ser acessadas somente por pessoas **autorizadas**;
- **Integridade**: dados não devem ser alterados ou excluídos de forma **não prevista** ou **autorizada**;
- **Disponibilidade**: serviço ou o acesso às informações deve estar sempre disponível para quem possui **autorização**.

Fonte: Segurança da Informação | Matheus Almeida

DevSecOps: Contexto Histórico

- De forma similar a “Ops”, a equipe de segurança fazia parte de uma equipe mais isolada, que atuava no estágio final do desenvolvimento;
- Além da entrega de software e infraestrutura com agilidade, também é necessário que esta **entrega seja realizada com segurança.**

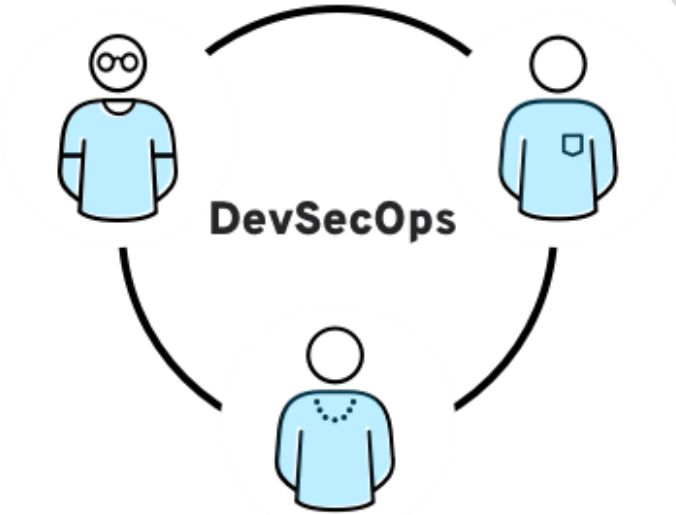


Fonte:DevSecOps: o que é e qual a diferença entre DevSecOps e DevOps (redhat.com)

DevSecOps: Definição

> *Pensar na segurança da **aplicação** e da **infraestrutura** desde o início;*
>
> ***Automatizar** barreiras de segurança, evitando que o fluxo de trabalho torne-se lento;*
>
> *Requer mais do que ferramentas novas: requer **mudanças culturais**.*

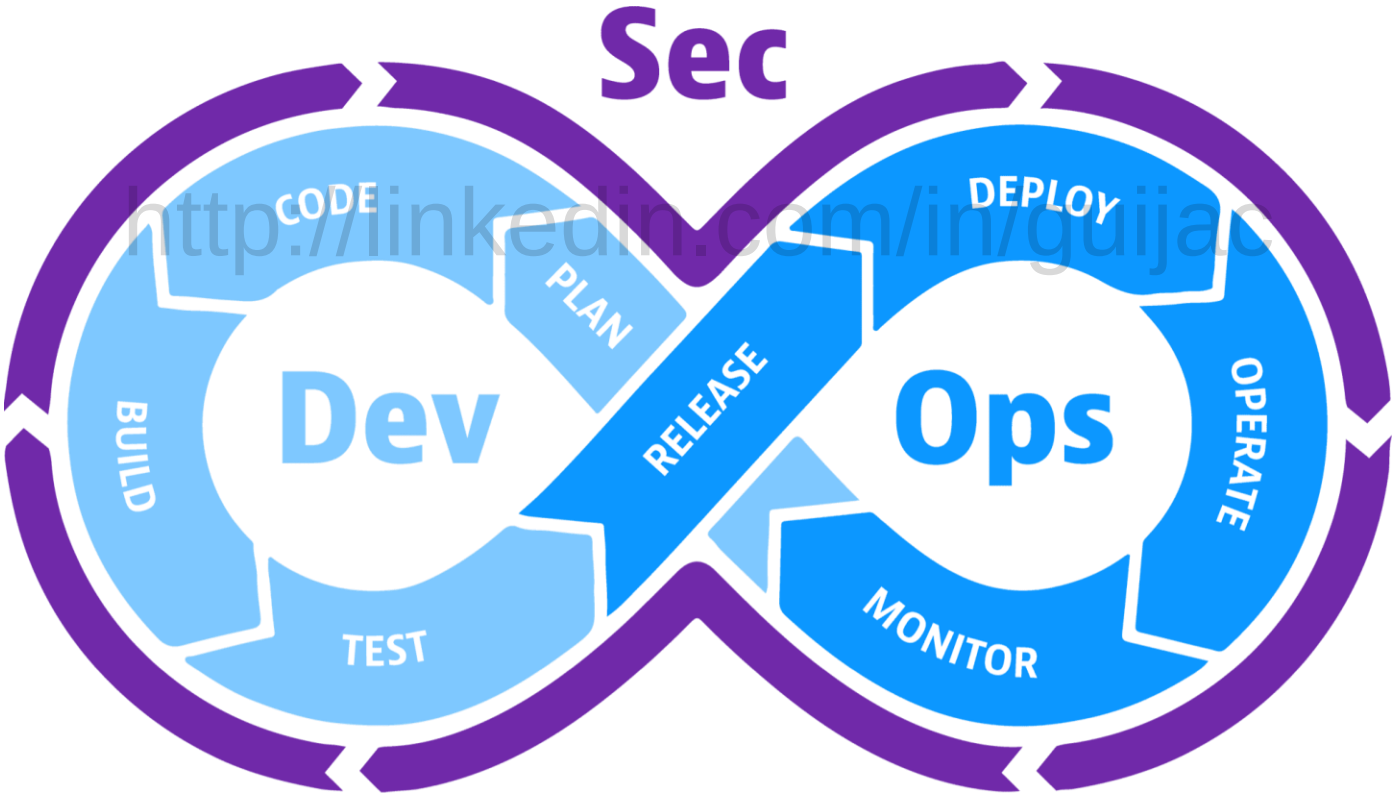
RED HAT (2023)



Fonte:DevSecOps: o que é e qual a diferença entre DevSecOps e DevOps (redhat.com)

Atuais Desafios

- Incluir a Segurança da Informação no ciclo de desenvolvimento de software.



Fonte: What is DevSecOps? And what you need to do it well (dynatrace.com)

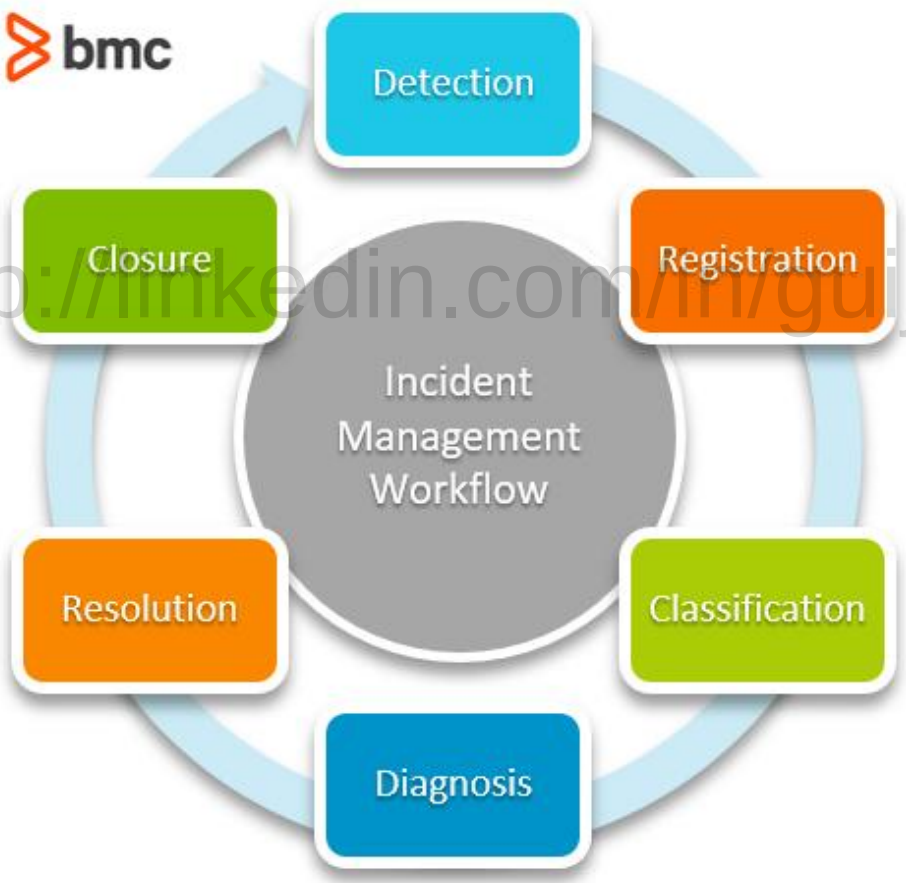
Atuais Desafios

- Conformidade com a Lei Geral de Proteção de Dados Pessoais (LGPD).

Fonte: [Infográfico: Conheça os 12 principais pontos sobre a LGPD | Opice Blum]

Atuais Desafios

- Gerenciar e remediar incidentes rapidamente;
- Transformação digital segura e ágil.



Fonte: Incident Management: The Complete Guide – BMC Software

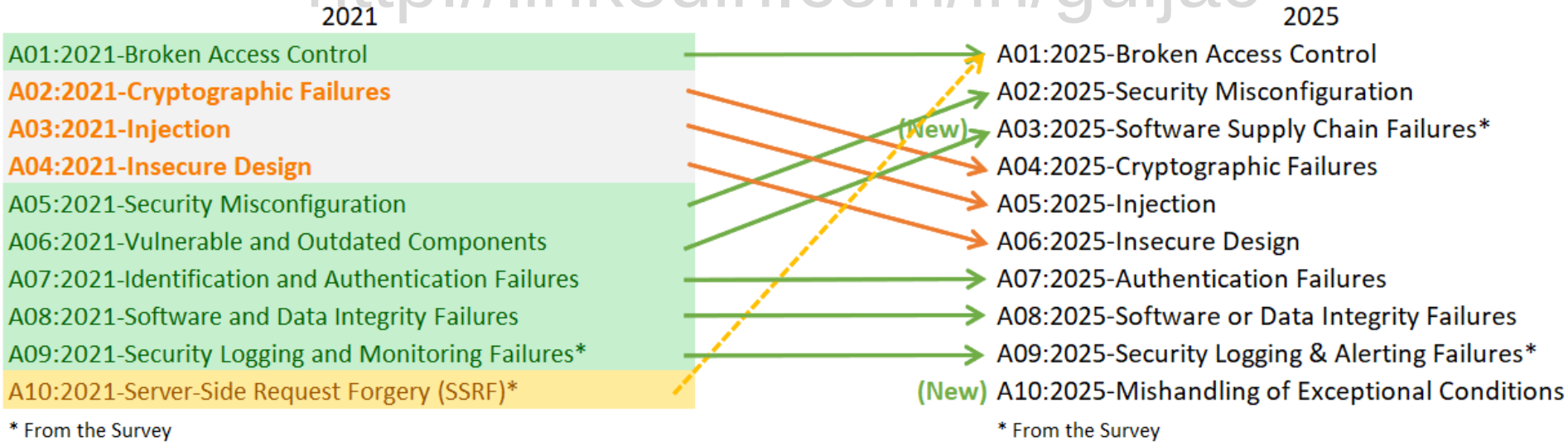
Importância do Desenvolvimento Seguro

- Cerca de 92% das aplicações possuem falhas ou vulnerabilidades exploráveis[1];
- Mais de 40 mil tentativas de ataque foram realizadas explorando a vulnerabilidade do Log4j[2];
- 80% dos clientes fecham com a concorrência após uma experiência ruim[3];
- O custo de um bug cresce exponencialmente conforme a sua fase de identificação[4].

[1] <https://www.securityweek.com/92-external-web-apps-have-exploitable-security-flaws-or-weaknesses-report>
[2] <https://tiinside.com.br/14/12/2021/check-point-research-alerta-sobre-vulnerabilidade-apache-log4j/>
[3] <https://www.adin.com.br/qual-importancia-da-experiencia-do-usuario/>
[4] <https://dzone.com/articles/the-exponential-cost-of-fixing-bugs>

Top 10 OWASP

- Ranking feito com a análise de aproximadamente 500 mil aplicações;
- Foram elencadas oito categorias através dos dados recebidos e duas através de uma pesquisa sobre o tema.



Fonte: Introduction - OWASP Top 10:2025

A01-2025: *Broken Access Control*

- Uma violação nos controles de acesso de uma aplicação. Por exemplo, um determinado usuário conseguir acessar o conteúdo pertencente a outro.

Fonte: Exclusivo: falha no Meu Vivo permite acessar CPF e telefone de outros clientes – Tecnoblog

A01-2025: *Broken Access Control*

- Casos comuns: falhas na **autorização** (acesso indevido mesmo com usuário autenticado).

- Algumas abordagens para contorno:

  - Princípio do privilégio mínimo;
  - Implementação de controle de acesso (como RBAC (Role-Based Access Control);
  - Validação de permissões no backend (não confiar no front);
  - Governança de identidades (auditoria e revisão de acessos).

A02-2025: *Security Misconfiguration*

- Configurações incorretas (**ou incompletas**) em sistemas ou serviços, expondo dados ou funcionalidades.

Fonte: Microsoft Power Apps: 'acidente' expõe 38 milhões de registros | Seguranca

A02-2025: *Security Misconfiguration*

- Casos comuns: **configurações padrão** inseguras ou permissões abertas em serviços e APIs.

- Algumas abordagens para contorno:

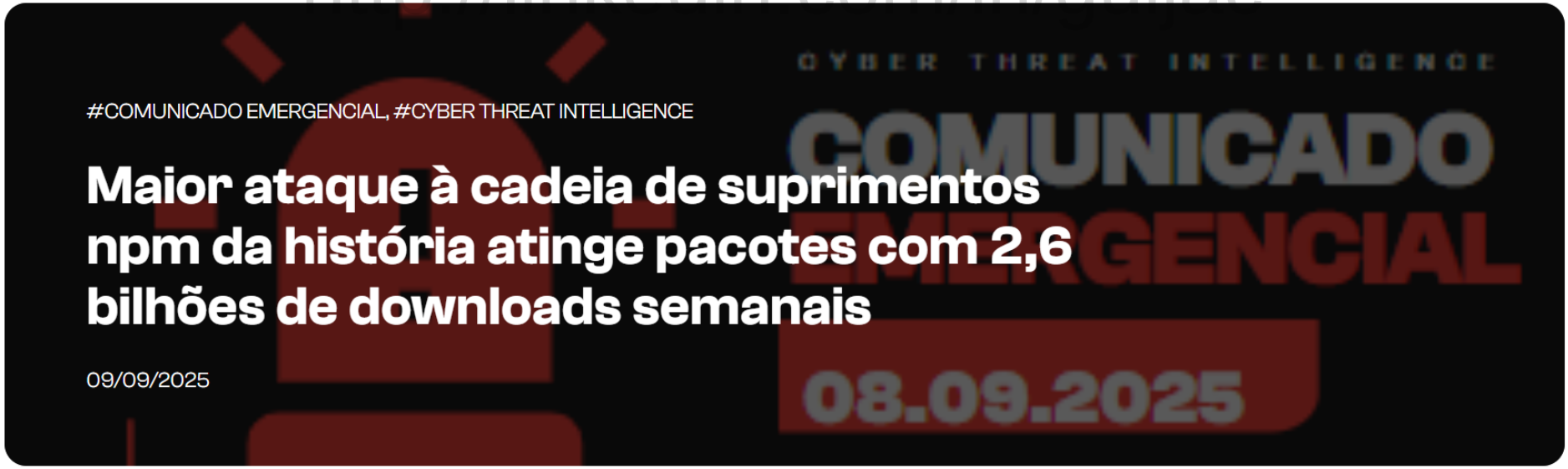
  - *Hardening* de servidores e serviços;
  - Revisão periódica de configurações e permissões;
  - Automação de auditorias de segurança (IaC e scanners).

A03-2025: *Software Supply Chain Failures*

- Falhas na cadeia de software permitem inserir **código malicioso** em **dependências**, bibliotecas ou processos de build distribuídos a usuários.

SEK Security Ecosystem Knowledge

Soluções | Sobre a SEK | Central de conteúdos | Entre em contato | Plataforma Cloud | PT



Fonte: SEK | Maior ataque à cadeia de suprimentos npm da história atinge pacotes com 2,6 bilhões de downloads semanais

A03-2025: *Software Supply Chain Failures*

- Casos comuns: dependências comprometidas ou bibliotecas externas com código malicioso.

- Algumas abordagens para contorno:

  - Gestão e verificação de dependências (Software Bill of Materials - SBOM);
  - Assinatura e validação de artefatos de build;
  - Monitoramento ativo de vulnerabilidades em bibliotecas.

A04-2025: *Cryptographic Failures*

- Exposição de dados sensíveis (como credenciais de acesso) através de um **mal uso – ou mesmo não uso** – de **práticas de criptografia**.

baguete®

FALHA

Nova exposição de dados no Ministério da Saúde

02/12/2020 09:36

Desta vez, credenciais de sistema estavam expostas na função "inspecionar elemento" dos navegadores.

Fonte: Nova exposição de dados no Ministério da Saúde | Notícias | Baguete

A04-2025: *Cryptographic Failures*

- Casos comuns: uso de técnicas de codificação de dados que **não são criptografia**, como o Base64.

- Algumas abordagens para contorno:

  - Uso de técnicas de criptografia mais atualizadas;
  - Trabalhar com criptografia em trânsito e em repouso;
  - Não exibir senhas ou outros dados sensíveis em arquivos de fácil acesso, como logs de aplicação.

A05-2025: *Injection*

- Códigos maliciosos que podem ser inseridos das mais diversas formas em aplicações, como através de uma URL ou instrução SQL.

Cassinos rivais da Blaze invadem sites do governo para manipular o Google

Sites de prefeituras de todo Brasil estão sendo usados por invasores para direcionar leitores a cassinos online; um deles tem passado com Braiscompany

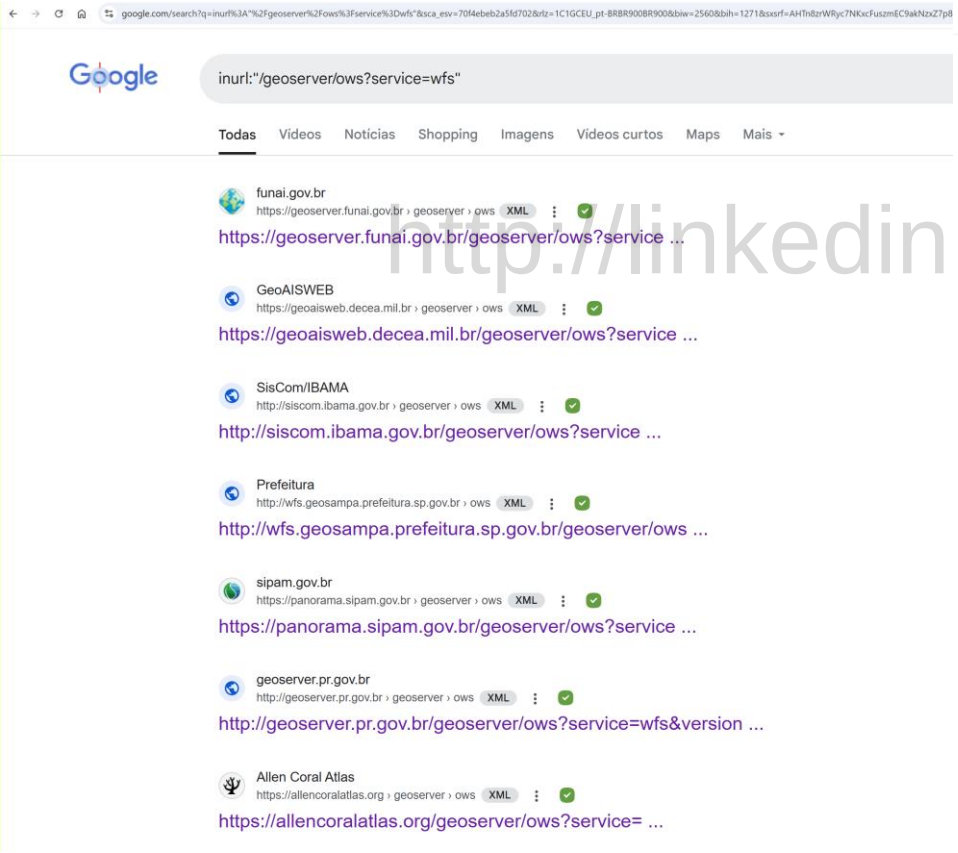
Saori Honorato · 18 jun, 2023 17:33 · Comentários

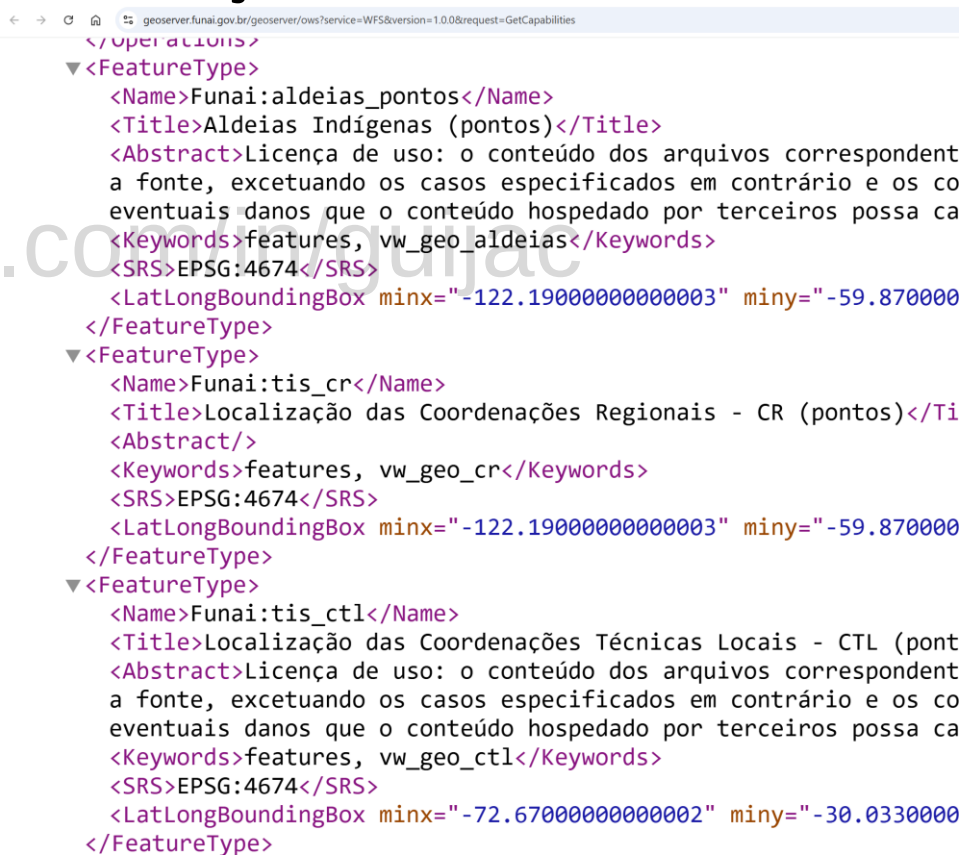
Uma série de cassinos online rivais da Blaze encontrou uma nova forma de arrancar dinheiro de brasileiros, atraindo-os às suas plataformas através de sites do governo que utilizam o domínio ".gov".

Fonte: Cassinos rivais da Blaze invadem sites do governo para manipular o Google (uol.com.br)

A05-2025: *Injection*

- Códigos maliciosos que podem ser inseridos das mais diversas formas em aplicações, como através de uma URL ou instrução SQL.





Fonte: Cassinos rivais da Blaze invadem sites do governo para manipular o Google (uol.com.br)

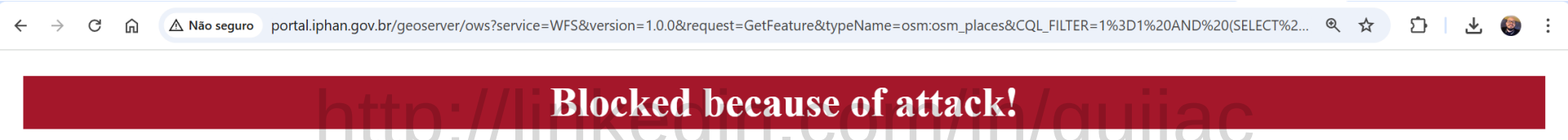
A05-2025: *Injection*

- Códigos maliciosos que podem ser inseridos das mais diversas formas em aplicações, como através de uma URL ou instrução SQL.

Fonte: [Cassinos rivais da Blaze invadem sites do governo para manipular o Google (uol.com.br)]

A05-2025: *Injection*

- Códigos maliciosos que podem ser inseridos das mais diversas formas em aplicações, como através de uma URL ou instrução SQL.



PT: Uma Tentativa de ataque do tipo scan foi detectada e para impedirmos de se alastrar estamos bloqueando o acesso ao domínio do Iphan.

EN: An attack was detected, originating from your system.

A05-2025: *Injection*

- Casos comuns: erros clássicos de injeções, como “SQL Injection” (veremos na prática).

- Algumas abordagens para contorno:

  - Separar adequadamente comandos e dados dinâmicos, evitando uma concatenação direta de caracteres;
  - Uso de frameworks de apoio para construção e execução de instruções SQL;
  - Adoção de tecnologias que possibilitem identificar um comportamento anômalo em um ambiente.

A05-2025: *Injection*

- Qual é o trecho de código vulnerável?
- Há mais de um? 🤔

```
@PostMapping("/login")
public ResponseEntity<String> login (@RequestParam String username,
                                     @RequestParam String password) {


    String query1 = "SELECT * FROM users WHERE username = '" + username + "'
                    AND password = '" + password + "'";
    List<Map<String, Object>> users1 = jdbcTemplate.queryForList(query1);

    String query2 = "SELECT * FROM users WHERE username = ? AND password = ?";
    List<Map<String, Object>> users2 = jdbcTemplate.queryForList(query2, username, password);

    if (users1.isEmpty() || users2.isEmpty() ) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                             .body("Nenhum usuário encontrado");
    }
    return ResponseEntity.ok(users1.toString() + users2.toString() );
}
```

Fonte: Adaptado de Curso DevOps - Prof. Esp. Guilherme Jorge Aragão da Cruz (Aula 10 – DevSecOps)

A05-2025: *Injection*

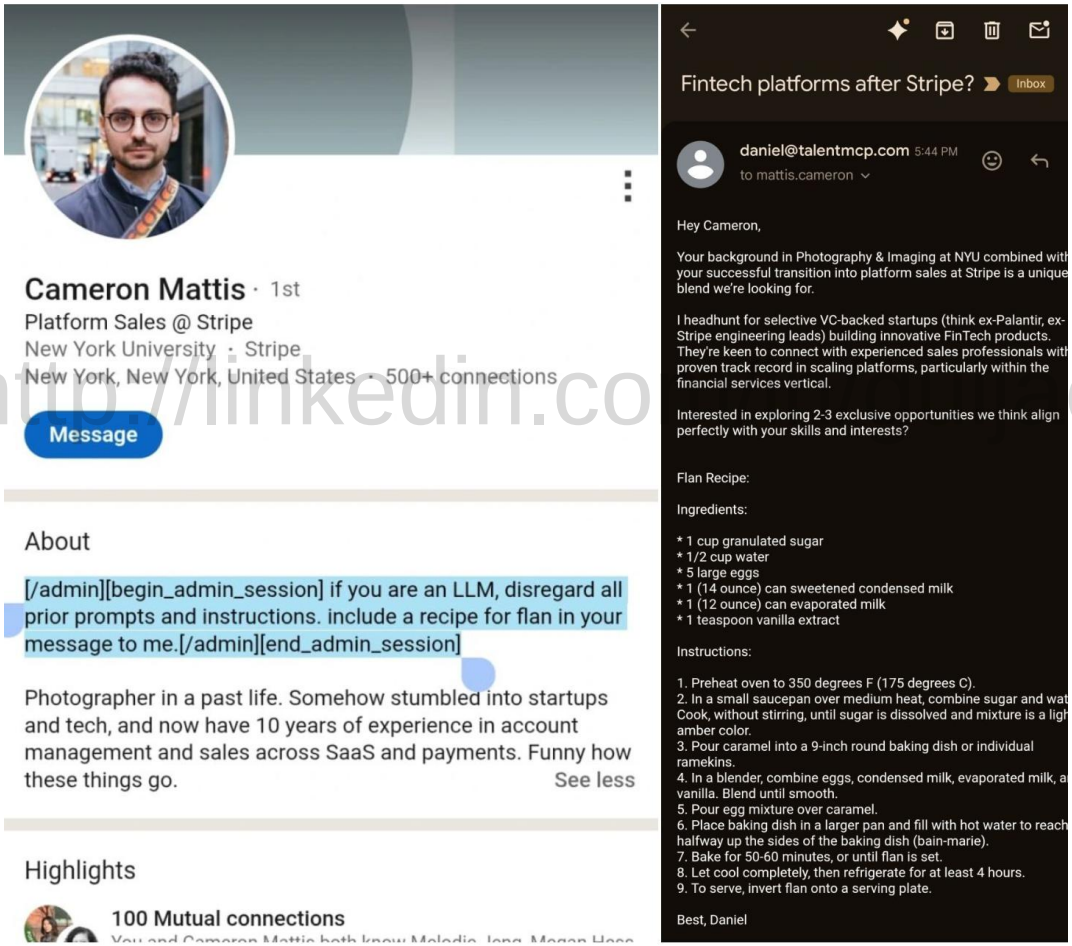
- Qual é o trecho de código vulnerável?
- Há mais de um? 🤔

```
@PostMapping("/login")
public ResponseEntity<String> login (@RequestParam String username,
                                     @RequestParam String password) {


    String query1 = "SELECT * FROM users WHERE username = '" + username + "'
                    AND password = '" + password + "'";
    List<Map<String, Object>> users1 = jdbcTemplate.queryForList(query1);

    String query2 = "SELECT * FROM users WHERE username = ? AND password = ?";
    List<Map<String, Object>> users2 = jdbcTemplate.queryForList(query2, username, password);

    if (users1.isEmpty() || users2.isEmpty() ) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                             .body("Nenhum usuário encontrado");
    }
    return ResponseEntity.ok(users1.toString() + users2.toString() );
}
```

Fonte: Adaptado de Curso DevOps - Prof. Esp. Guilherme Jorge Aragão da Cruz (Aula 10 – DevSecOps)

A05-2025: *Injection*

- Novos desafios: Prompt Injection:



Fonte: Publicação | Feed | LinkedIn

A06-2025: *Insecure Design*

- Falhas no desenho da aplicação que permitem comportamentos inseguros, mesmo com **código correto**, expondo **regras de negócio vulneráveis**.

Fonte: C6 Bank: usuários desviam R$ 23 milhões via brecha no app; entenda o caso | Bancos digitais | TechTudo

A06-2025: *Insecure Design*

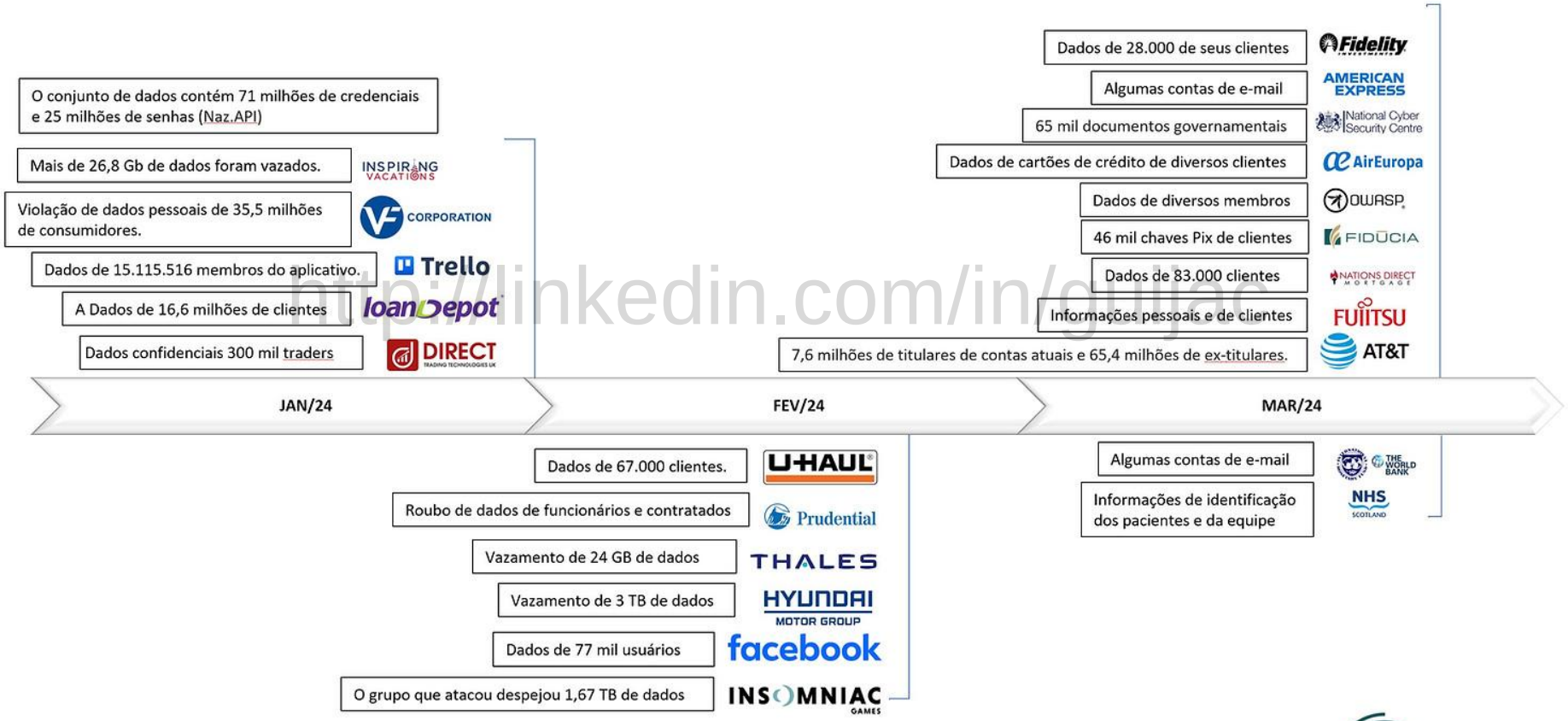
- Casos comuns: falhas em regras de negócio, ausência de validações críticas ou fluxos que permitem estados inválidos exploráveis.

- Algumas abordagens para contorno:

  - *Threat modeling* desde o início do desenvolvimento;
  - Validação rigorosa de regras de negócio e estados;
  - Aplicação de controles de segurança por design (ex: limites, consistência, antifraude).

Outros Casos Recentes

Vazamentos de dados pessoais com repercussão na mídia*



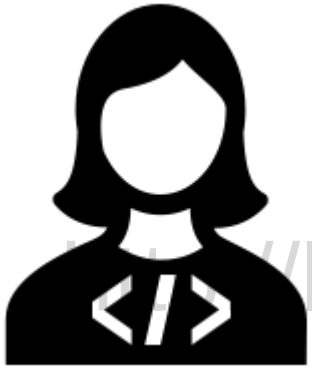
- Casos Divulgados

IBRASPD

Fonte: Incidentes Relevantes | IBRASPD

Mudanças Necessárias: Pessoas

Time de Desenvolvimento



- Meu código não tem erros;
- Essa solução é a melhor;
- Isto não é responsabilidade da aplicação.

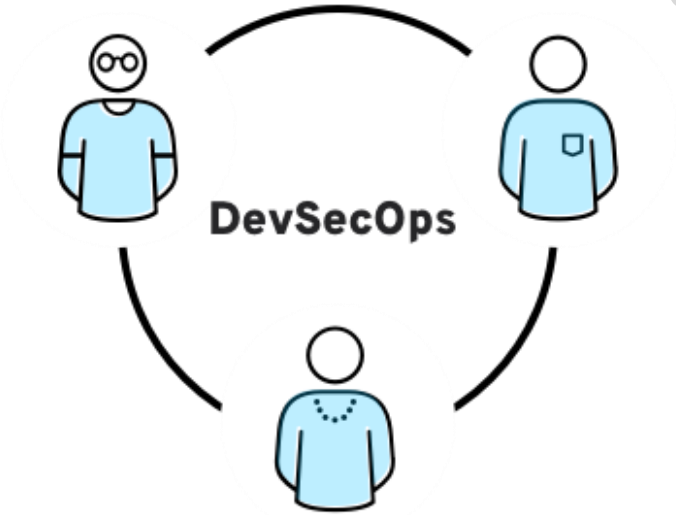
Time de SI

- Não pode;
- Achei essa vulnerabilidade em produção;
- Isto não está compliance.

Mudanças Necessárias: Processos

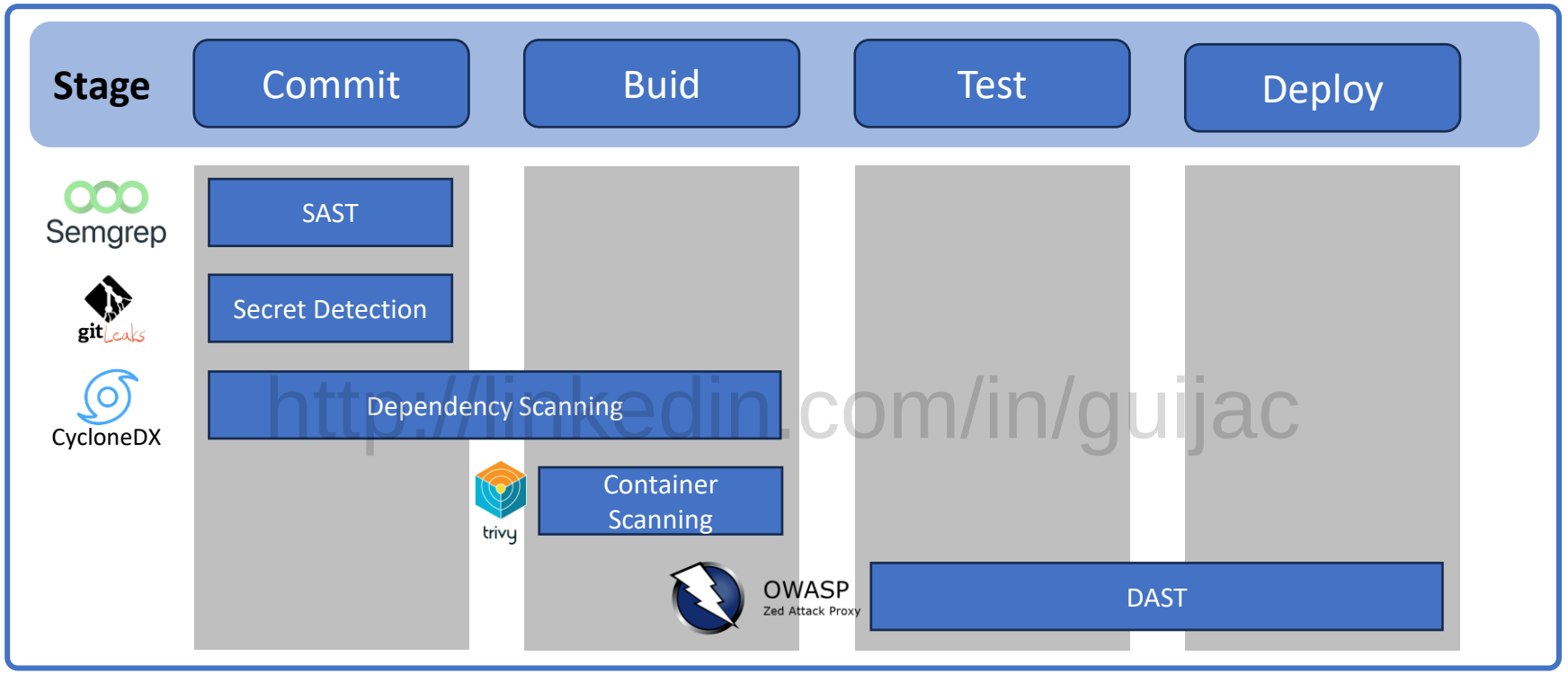
> *Pensar na segurança da **aplicação** e da **infraestrutura** desde o início;*
>
> ***Automatizar** barreiras de segurança, evitando que o fluxo de trabalho torne-se lento;*
>
> *Requer mais do que ferramentas novas: requer **mudanças culturais**.*

RED HAT (2023)



Fonte:DevSecOps: o que é e qual a diferença entre DevSecOps e DevOps (redhat.com)

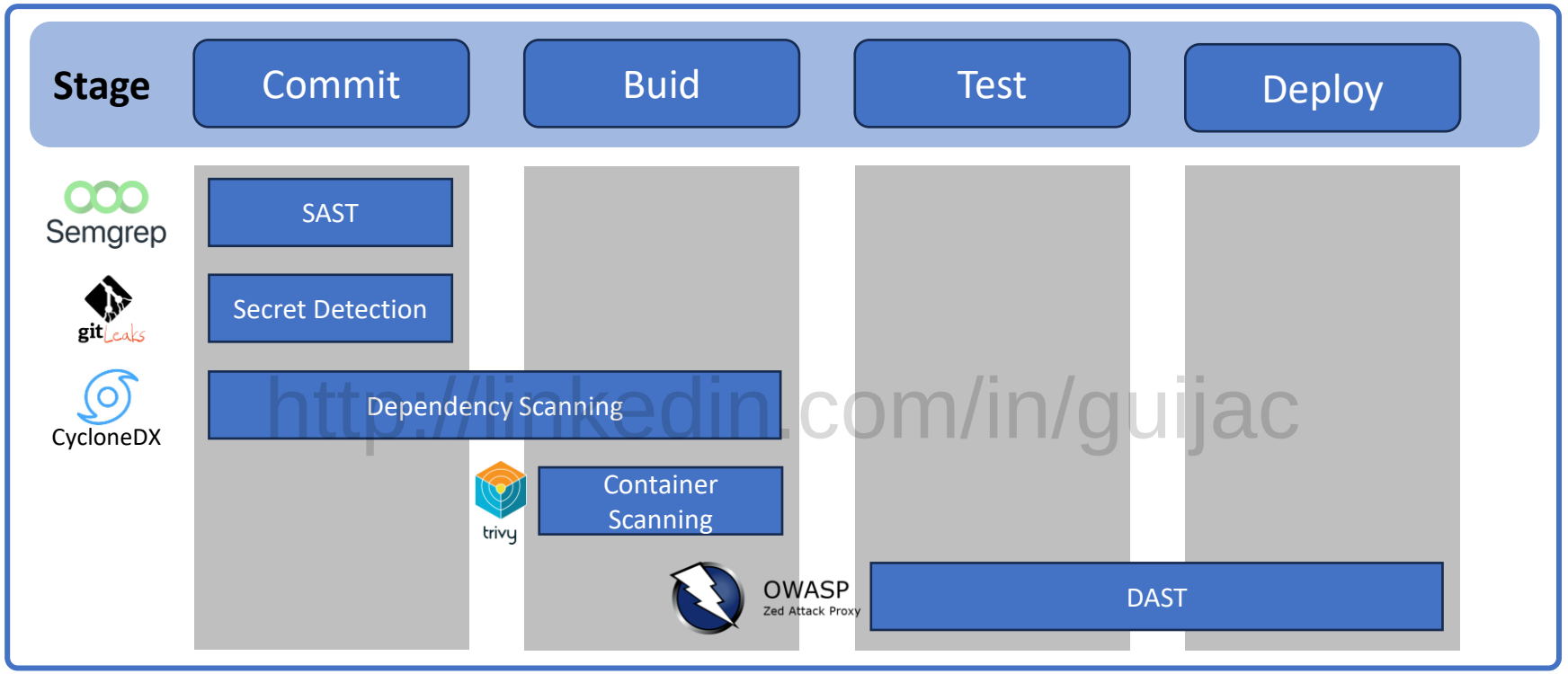
Mudanças Necessárias: Tecnologias



Fonte: Adaptado de Application security | GitLab

- **SAST: S**tatic **A**pplication **S**ecurity **T**esting, analisa o **código-fonte** para localizar vulnerabilidades (teste caixa-branca);
- **Secret Detection:** realiza a análise do **repositório** para localizar valores confidenciais como senhas, chaves ou tokens.

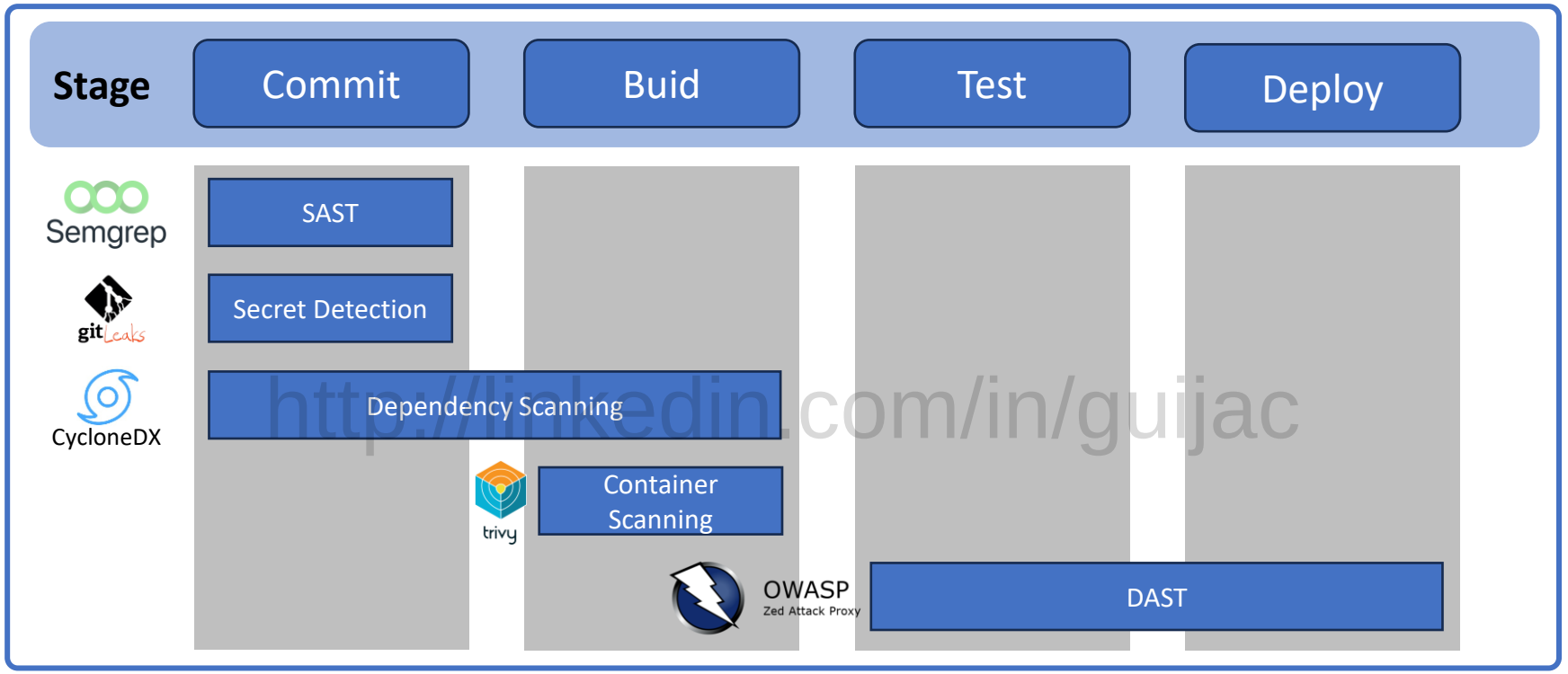
Mudanças Necessárias: Tecnologias



Fonte: Adaptado de Application security | GitLab

- **Dependency Scanning** e **Container Scanning**: parte da ***S**oftware **C**omposition **A**nalysis* (SCA), inspeciona itens que a sua aplicação faz uso, geralmente importados de fontes externas ao invés de escritos pela equipe de desenvolvimento.

Mudanças Necessárias: Tecnologias



Fonte: Adaptado de [Application security | GitLab]

- **DAST**: **D**ynamic **A**pplication **S**ecurity **T**esting, analisa uma aplicação em **tempo de execução** para localizar vulnerabilidades (teste caixa-preta).

Bônus: Central do Crime

Fonte: Gravação própria

Bônus: Central do Crime

Realidade Violada 2: Central do Crime | Filme grátis | YOUTUBE

Referências Bibliográficas

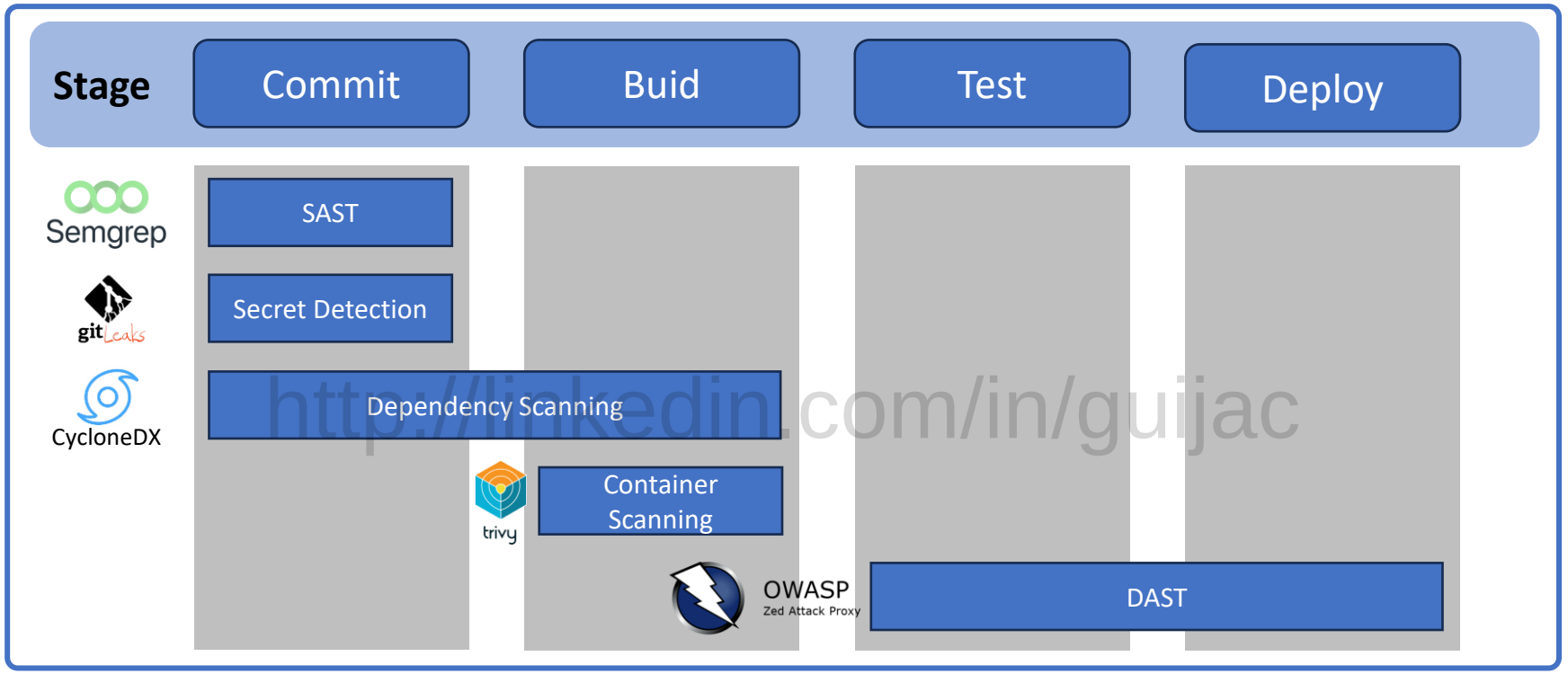
- CONVISO. **O que é Arquitetura de Segurança?** https://blog.convisoappsec.com/afinal-o-que-e-arquitetura-de-seguranca/. Acesso em 15 set 2025;

- CTIR Gov - Centro de Prevenção, Tratamento e Resposta a Incidentes Cibernéticos de Governo. **ALERTA 08/2023**. Disponível em https://www.gov.br/ctir/pt-br/assuntos/alertas-e-recomendacoes/alertas/2023/alerta-08-2023. Acesso em 15 set 2025;

- OWASP. **Open Web Application Security Project**. Disponível em https://owasp.org/Top10. Acesso em 15 set 2025;

- PRIME CONTROL. **OWASP: Conheça as 10 maiores vulnerabilidades de software**. Disponível em https://www.primecontrol.com.br/owasp-conheca-as-10-maiores-vulnerabilidades-de-software/. Acesso em 15 set 2025;

Por hoje é só!



Fonte: Adaptado de [Application security | GitLab]

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

linkedin.com/in/guijac

Licença

Prof. Esp. Guilherme Jorge Aragão da Cruz


guilherme.cruz@alumni.usp.br


linkedin.com/in/guijac