Curso de Extensão

Tecnologias Microsoft

UNICAMP

INF-0998

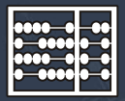
Programação segura (Segurança de software) Aula 1

Prof. Dr. Alexandre Braga, CISSP, CSSLP, PMP

alexbraga@ic.unicamp.br / ambraga@cpqd.com.br / braga.alexandre.m@gmail.com

br.linkedin.com/in/alexmbraga

12 de Novembro de 2022



INSTITUTO DE COMPUTAÇÃO

Ementa, plano de aulas, avaliação, etc.

Ementa proposta

- Introduction to cybersecurity: fundamentals and security services
- Cryptographic algorithms and protocols for confidentiality, integrity, authenticity, and non-repudiation
- Authentication and access control
- Secure programming: common errors and best practices

Plano de aulas

Aula 1 (12/11/2022)

- Parte 1-1 (sala de aula): Introdução e conceitos fundamentais de segurança cibernética. Processo de desenvolvimento de software seguro MS-SDL.
- Parte 1-2 (laboratório): Modelagem de ameaças com a ferramenta Microsoft Threat Modeling Tool.

Aula 2 (19/11/2022)

- Parte 2-1 (sala de aula): Algoritmos criptográficos e protocolos para confidencialidade, integridade, autenticidade e não repúdio.
- Parte 2-2 (laboratório): Prática de criptografia aplicada.

Aula 3 (26/11/2022)

- Parte 3-1 (sala de aula): Regras de ouro do software seguro e boas práticas de segurança de software.
- Parte 3-2 (laboratório): Prática de testes de segurança e análise de vulnerabilidades de software.

Aula 4 (3/12/2022)

- Parte 4-1 (sala de aula): Falhas de projeto de software, autenticação e controle de acesso e segurança de APIs.
- Parte 4-2 (laboratório): Prática de testes de segurança de APIs e análise de vulnerabilidades semânticas.

Atendimento

Datas e horários

- Monitor: Terças-feiras, das 18h às 19h e Quintas-feiras, das 18h às 19h.
- Professor: Segundas-feiras, das 18h às 19h.

Avaliação

- Trabalho 1 (T1): 50% da nota
  - Em grupo de até 3 alunos
  - Entrega: **05/12/2022**
  - Modelagem de ameaças com ferramenta Threat Modeling Tool
    - Escolher uma aplicação de seu interesse (hipotética)
    - Entregar: Descrição do sistema (1 página)
    - Entregar: Modelo de ameaças na ferramenta
      - Arquivo no formato da ferramenta
      - Relatório exportado em HTML e salvo em PDF
    - Critérios de entrega:
      - Tamanho do modelo (> 30 objetos, < 50 objetos)
      - Riqueza de detalhes das ameaças
        - Tecnologias específicas (Cloud, IoT, Azure, etc.)

Avaliação

Trabalho 2 (T2): 50% da nota

- Individual
- Entregar: Resolução de todos os playbooks da disciplina
- Entrega: **12/12/2022** (segunda-feira)
- Critérios de entrega:
  - Usar screenshots como evidências de execução da tarefa
  - Usar seu nome como texto claro nos exercícios
  - Extra: Encriptar a entrega com chave pública do professor (+1)
    - Observação: O trabalho só será corrigido se o professor conseguir decriptar a entrega
  - Extra: Assinar a entrega com a chave privada do aluno (+1)
    - Observação: Professor já deve conhecer a chave pública do aluno

Parte 1-1 (sala de aula) - Introdução

Objetivos da segurança de software

Segurança de software é sobre minimização de riscos

- Agindo na mitigação dos riscos inaceitáveis
- Reduzir a chance (probabilidade) de que as pessoas, os sistemas e os dados sejam atacados (mal usados) causando prejuízos financeiros, perigos físicos, danos a reputação e a imagem de pessoas e empresas
- Minimização de risco é sobre entender o que pode dar errado, como e com qual impacto, e assim se preparar para agir de modo apropriado

Não é requisito funcional

Segurança é uma propriedade emergente do software

"*Security is not a feature that can simply be added to a software system, but rather a property emerging from how the system was built and is operated*"

Gary McGraw

Fonte: https://www.garymcgraw.com

Segurança by design

Projeto das funcionalidades com segurança em mente

Não é um adendo a posteriori

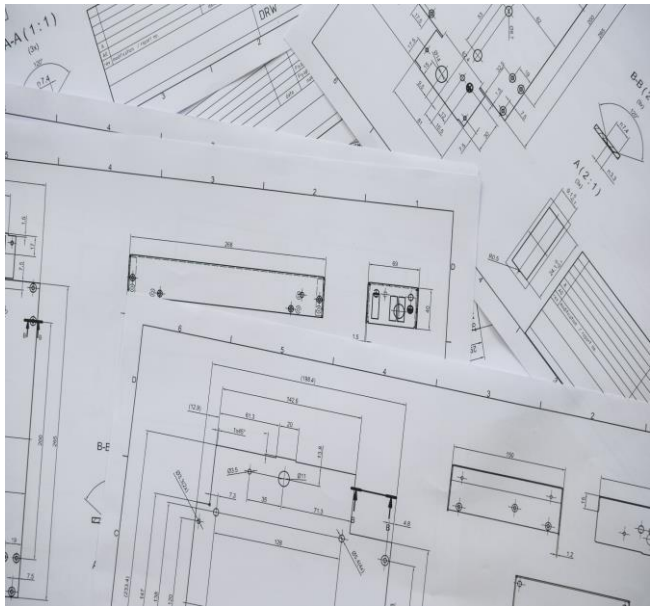
- É mais difícil “plugar” segurança depois
- Projeto incremental da segurança desde o início
- “Good Enough” Security

Objetivos concretos e mensuráveis. Ex:

- Apenas certos usuários podem fazer ação X
- Saída do serviço Y deve ser criptografada
- Função Z deve estar disponível 99.9% do tempo

Contra exemplos:

- Windows 98 – sem login no modo de segurança
- Internet – confiança irrestrita nos IPs e no tráfego
- Form WEB sem validação de dados de entrada
- APIs sem controle de acesso aos serviços



Evitar o “casco de tartaruga”

Projeto de segurança baseado na concha do caramujo

Uma casca externa resistente não deve ser a única linha de defesa

- Sistema inerentemente inseguro protegido por outro sistema mediador do controle de acesso

Exemplos de *turtle shell*:

- Firewalls guardam sistemas vulneráveis
- WAF antes de aplicação web vulnerável

Exemplos na cultura pop (para geeks!)

- Estrela da Morte em Star Wars
- Escudos da Nave Enterprise em Star Trek

Segurança X conveniência

Propriedades conflitantes?

As vezes são inversamente proporcionais

- Mais seguro → menos conveniente
- Muito conveniente → menos seguro

Muito inconveniente → difícil de usar → usuários vão evadir → uso inseguro!

- Ex.: Senhas geradas aleatoriamente e trocadas frequentemente são mais suscetíveis a serem escritas e guardadas em locais inseguros

O desafio é o equilíbrio

- Benefício relativo de segurança com um pequeno aumento da inconveniência

Usabilidade X segurança

Programadores e a usabilidade da segurança

O desenvolvimento seguro é útil se os programadores:

- São conscientes das ações de segurança que realizam
- Sabem como realizar as ações de segurança
- Não cometem erros perigosos
- Estão confortáveis com as ferramentas de segurança

A segurança é usável quando

- As tarefas são fáceis e seguras ao mesmo tempo
- Ação segura não depende de documentação
- Segurança está ativa por default,
- Segurança é projetada para ser fácil de usar

Usuários e programadores ignoram mensagens de alerta

- Tem ações inseguras e precisam de ajuda na operação

O que é software seguro?

O software seguro é aquele que contempla ...

com um grau alto de confiança justificada,

mas não com certeza absoluta, ...

um conjunto substancial de *propriedades* explícitas de segurança e de *serviços* de segurança.

O software seguro resulta de um *ciclo de desenvolvimento* com segurança e de uma *arquitetura* de segurança

Conceitos fundamentais

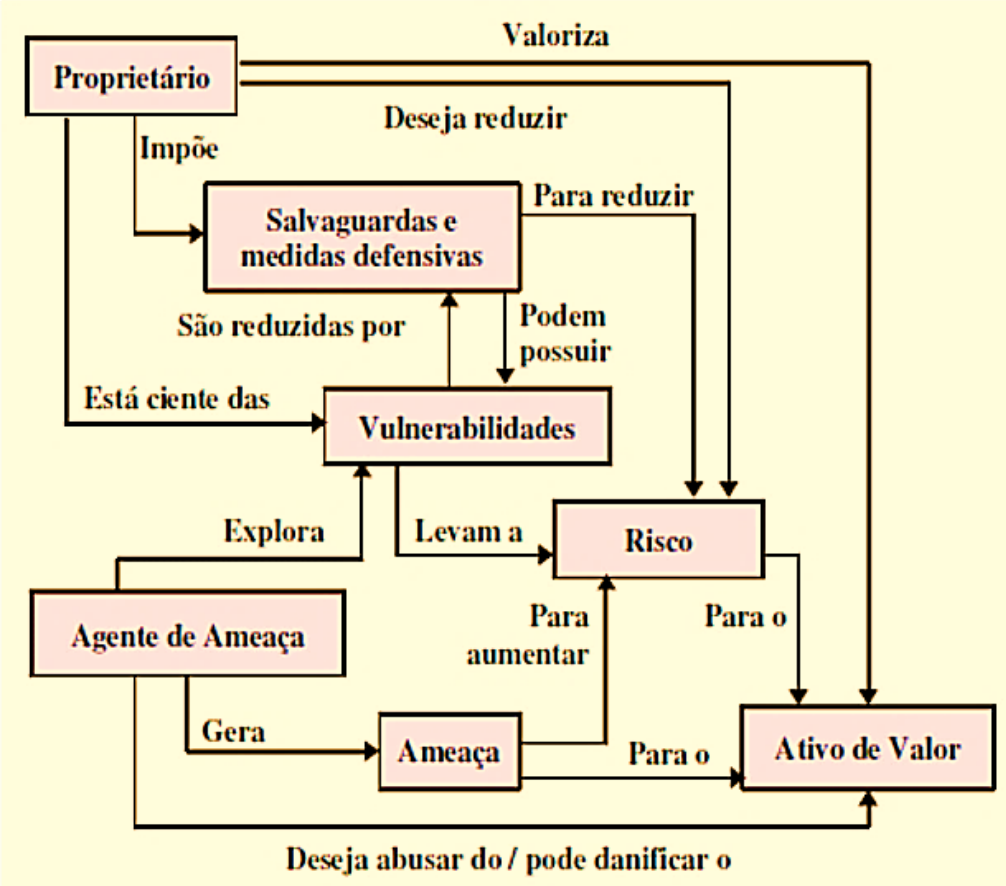
Palavras comuns com significado específico em segurança

- **Fraqueza** é um defeito ou falha de software que pode ser explorada maliciosamente
- **Vulnerabilidade** é a fraqueza que foi explorada maliciosamente (exploração documentada)
- **Ameaça** é o potencial/possibilidade de exploração da vulnerabilidade, intencional ou não
- **Agente da ameaça** é algo (natureza) ou alguém (pessoa) que pode concretizar a ameaça
- **Ataque** é a concretização ou realização de uma ameaça, com potencial de danos ou prejuízos
- **Risco** é a chance do agente de ameaça concretizar o ataque e a perda potencial devido a ele
- **Controle** de segurança é a contramedida (*durante*) ou salvaguarda (*antes*) que reduz o risco
    - Controle pode ser administrativo (p.ex. política), técnico (p. ex. criptografia) ou físico (p.ex. câmera)
    - Controle protege por dissuasão (despersuasão), prevenção, detecção, correção, ou recuperação
- **Controle compensatório** é alternativo e substitui o controle ideal, por razões de negócio

Conceitos estão todos relacionados

Os relacionamentos entre os conceitos explicam a proteção

- Os agentes de ameaça exploram vulnerabilidades presentes nos ATIVOS de valor da empresa (que mantém dados sensíveis), concretizando as ameaças, com impactos negativos ao negócio
- O proprietário valoriza o ativo (de valor). Ele está ciente das vulnerabilidades e fraquezas e deseja reduzir o risco sobre o ativo. Por isso, ele impõe salvaguardas e contramedidas para reduzir o risco
- O risco é reduzido pela diminuição da chance de ataque ou pela redução do impacto do ataque



Objetivos da Segurança

Seis visões complementares de segurança: CIDA+2

Confidencialidade	Garantir que apenas as pessoas autorizadas tem acesso às informações (sigilo)
Integridade	Garantir a salvaguarda e a precisão das informações contra adulteração e/ou falsificação
Disponibilidade	Garantir que as informações estão prontas para serem usadas por quem está autorizado
Autenticidade	Garantir a veracidade das informações e que a origem e o destino são verdadeiros
Irrefutabilidade	Garantir que a autoria da informação não pode ser negada (não-repúdio)
Privacidade	Garantir que dados pessoais não são coletados e nem divulgados sem a anuência do proprietário

Tipos de ameaças STRIDE

STRIDE é o modo de identificar ameaças criado na Microsoft

Ameaça	Objetivo violado	Descrição
Spoofing	Autenticidade	Personificação (fingir ser) algo (um programa) ou alguém
Tampering	Integridade	Adulteração, modificação não autorizada de dados ou códigos
Repudiation	Irrefutabilidade	Negação ou alegação falsa de não ter realizado uma ação
Information disclosure	Confidencialidade	Revelação de informação para alguém não autorizado
Denial of Service (DoS)	Disponibilidade	Negar ou deteriorar o serviço prestado aos usuários
Elevation of privilege	Autorização	Obter capacidades (permissões) sem a devida autorização

- Elevação de privilégios viola a autorização presumida para o usuário (também viola a autenticidade)
- A engenharia social não faz parte do STRIDE, mas pode ser considerada um ataque cibernético de trapaça por meios computacionais. O exemplo mais comum de engenharia social, atualmente, é o Phishing

Exemplos de ataques STRIDE

STRIDE é o modo de identificar ameaças criado na Microsoft

Tipo de ataque	Exemplos de ataques
Spoofing	Usuário X conhece a senha do usuário Y e loga no sistema como usuário Y; arquivo xpto.dll malicioso é renomeado e substitui DLL verdadeira xyz.dll apagada
Tampering	Modificar código fonte sem permissão; injetar script na aplicação web; modificar requisições HTTP; modificar arquivos binários; editar dados incorretos de usuário
Repudiation	“Não mandei o e-mail”; “não é a minha voz na gravação”; “não visitei o site”; “não acessei aquele arquivo”; “o dinheiro naquela conta não é meu...”
Information disclosure	Data leaks (vazamentos de dados); acessos sem autorização às listas de usuários e arquivos privados; acessos não autorizados aos dados pessoais
Denial of Service (DoS)	Causar erro fatal em servidor pela injeção de payload corrompido; interromper a comunicação com um site por causa de DNS falsificado; demora na resposta por lentidão do servidor causada pelo uso excessivo de CPU/memória/disco
Elevation of privilege	Usuário da internet (guest) executa comandos no sistema operacional local como root; usuário comum executa comandos como administrador

Métodos de segurança de software

Métodos de desenvolvimento de software seguro

Métodos, metodologias ou processos de desenvolvimento de software seguro são maneiras de organizar e conduzir as atividades de segurança durante o desenvolvimento de software, desde a concepção até a implantação da aplicação

- Tarefas específicas
- Papeis específicos
- Fluxos específicos

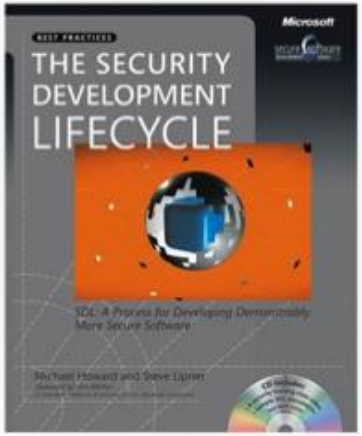
Existem vários!

Métodos de segurança de software

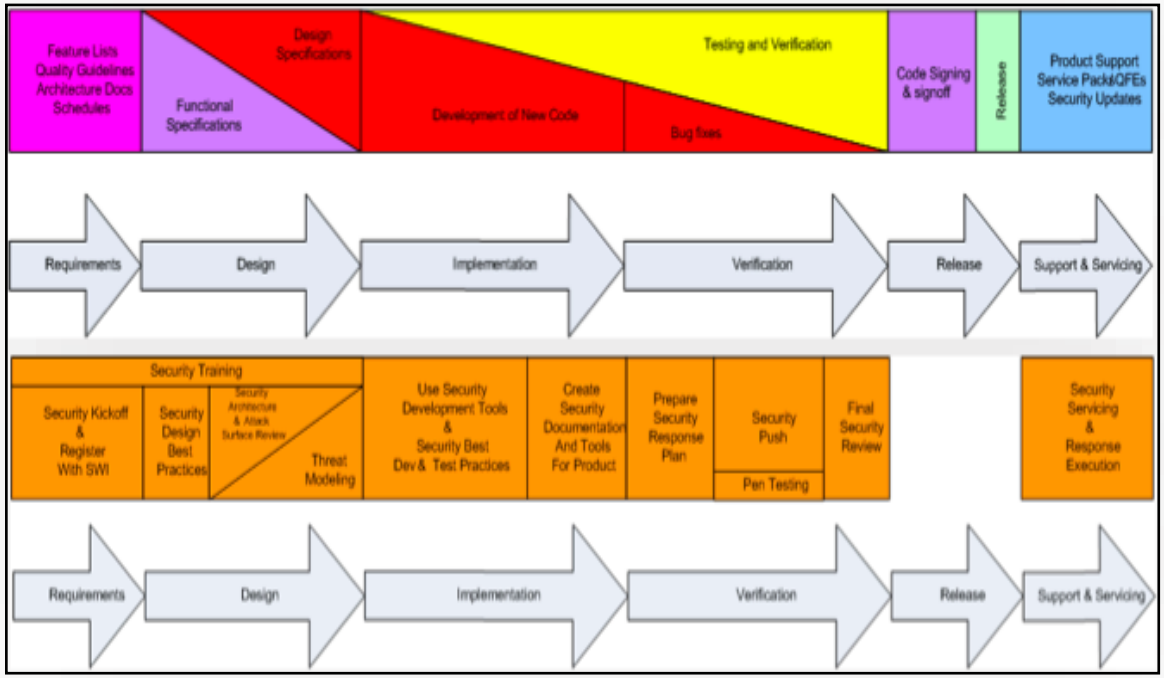
Métodos de segurança de software

Microsoft SDL

- Desenvolvido pela Microsoft apenas para uso interno
- Foi estendido e publicado para o público em geral
- https://www.microsoft.com/en-us/securityengineering/sdl



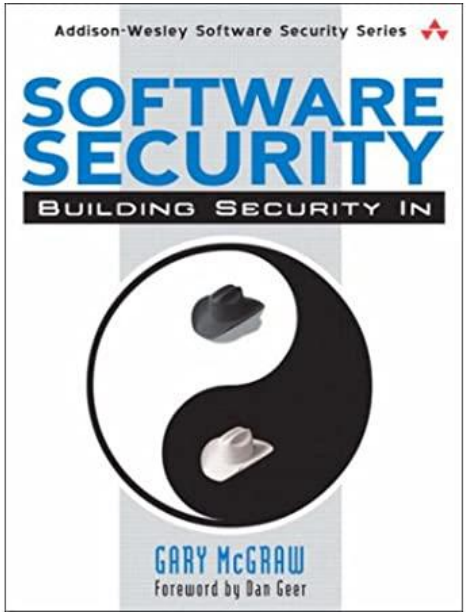
Michael Howard and Steve Lipner. The security development lifecycle. Redmond. Microsoft Press, 2006

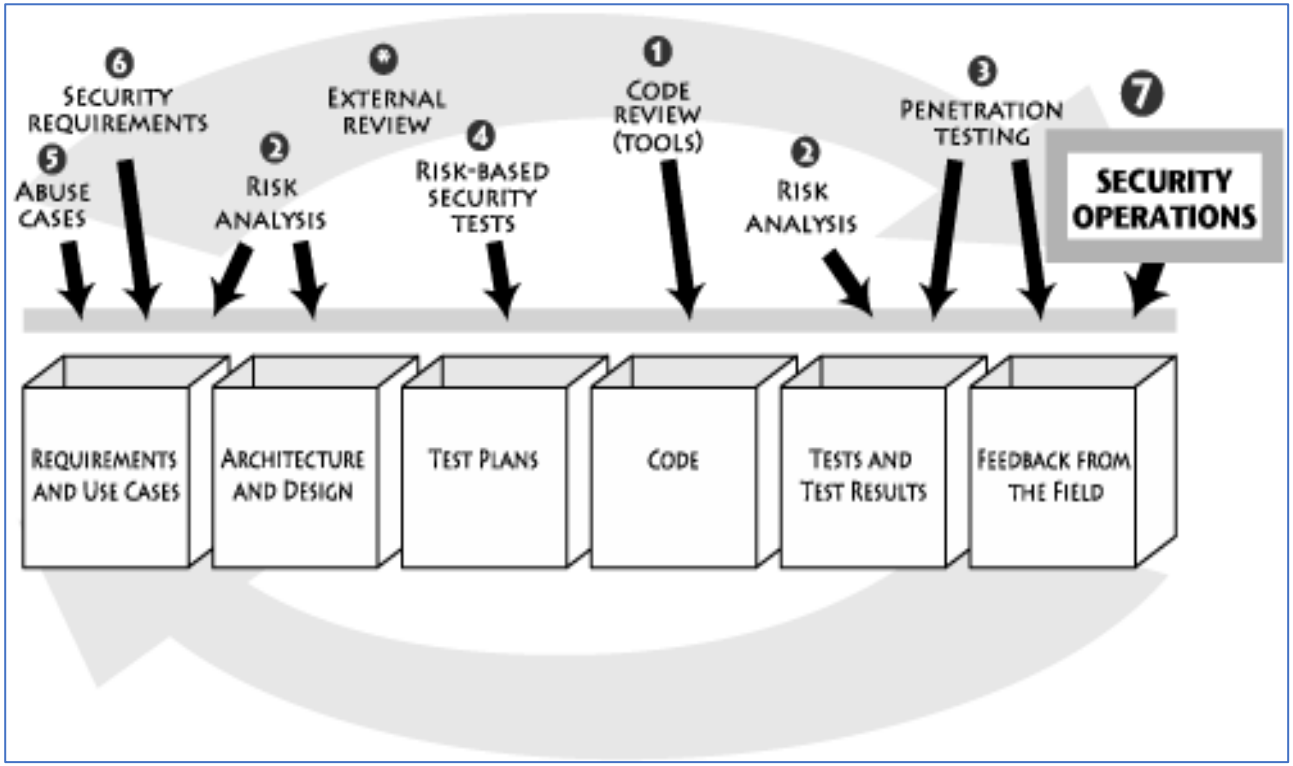


Métodos de segurança de software

Touchpoints (Push left while advancing right)

Desenvolvido pelo Dr. Gary McGraw





Métodos de segurança de software

OWASP CLASP

- Comprehensive, Lightweight Application Security Process (CLASP)
- Foco em 7 AppSec Best Practices
- Cobre todo o ciclo de software
- Adaptável a qualquer processo de desenvolvimento
- Define papeis de segurança pelo SDLC
- 24 atividades baseadas em papeis

Já foi mais popular na época do Waterfall

Ainda é útil para definir responsabilidades de AppSec nos squads

Métodos de segurança de software

Comparação SDL x CLASP x Touchpoints

Microsoft SDL
- Bom para empresas grandes que fazem software em larga escala, como a Microsoft

Touchpoints
- Muito alto nível para não especialistas, faltam detalhes para executar nos squads/times

CLASP
- Muitas atividades, mas sem priorização

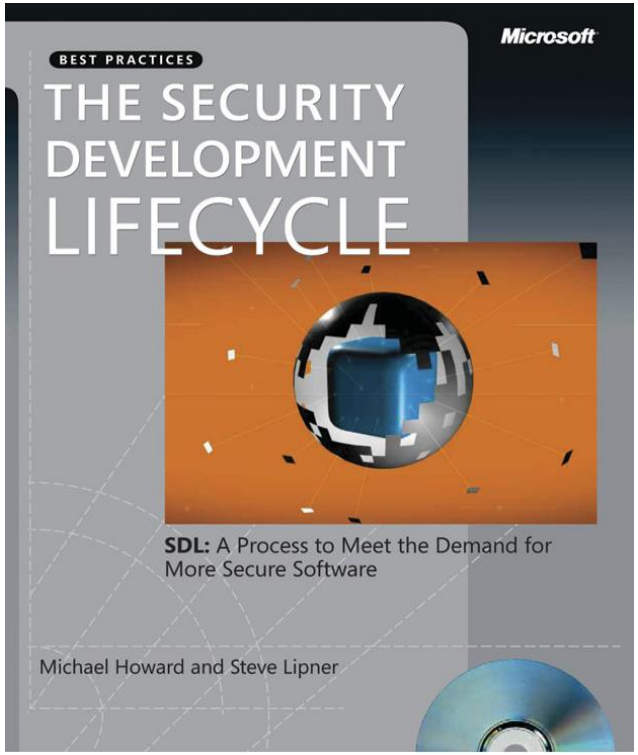
Todos eles
- Bons para orientar o trabalho dos especialistas
- Difíceis para não especialistas em segurança

Métodos de segurança de software

Nesse curso, vamos estudar o método de desenvolvimento de software seguro da Microsoft (MS-SDL)

Motivações para o MS-SDL

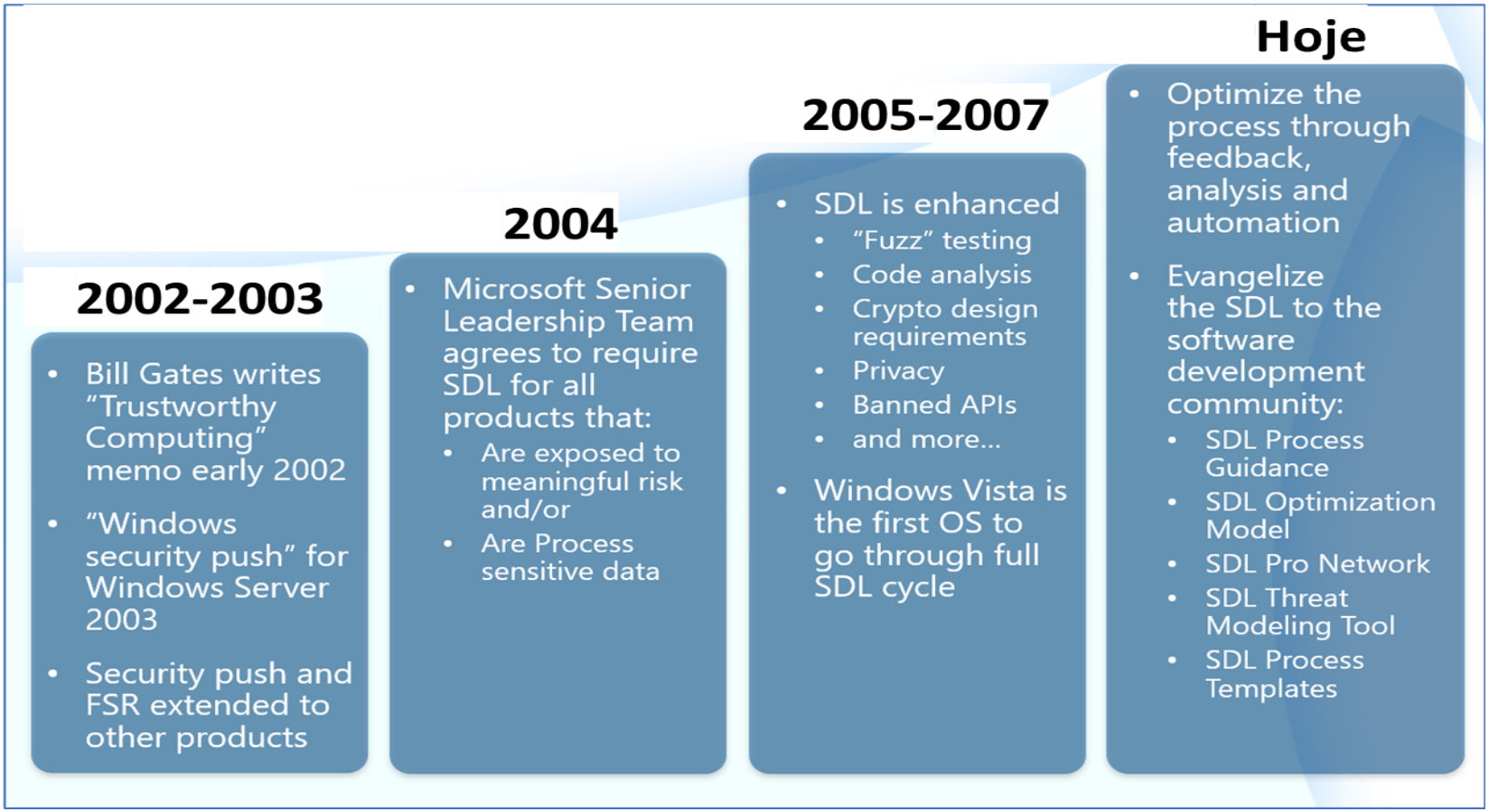
- Apenas Bug Hunting não deixa um software seguro
- Reduz chance de vulnerabilidades no software
- Requer compromisso dos executivos
- Requer melhoria contínua
- Requer treinamento e capacitação
- Requer ferramentas automatizadas
- Requer incentivos e consequências (penalidades?)



Microsoft SDL

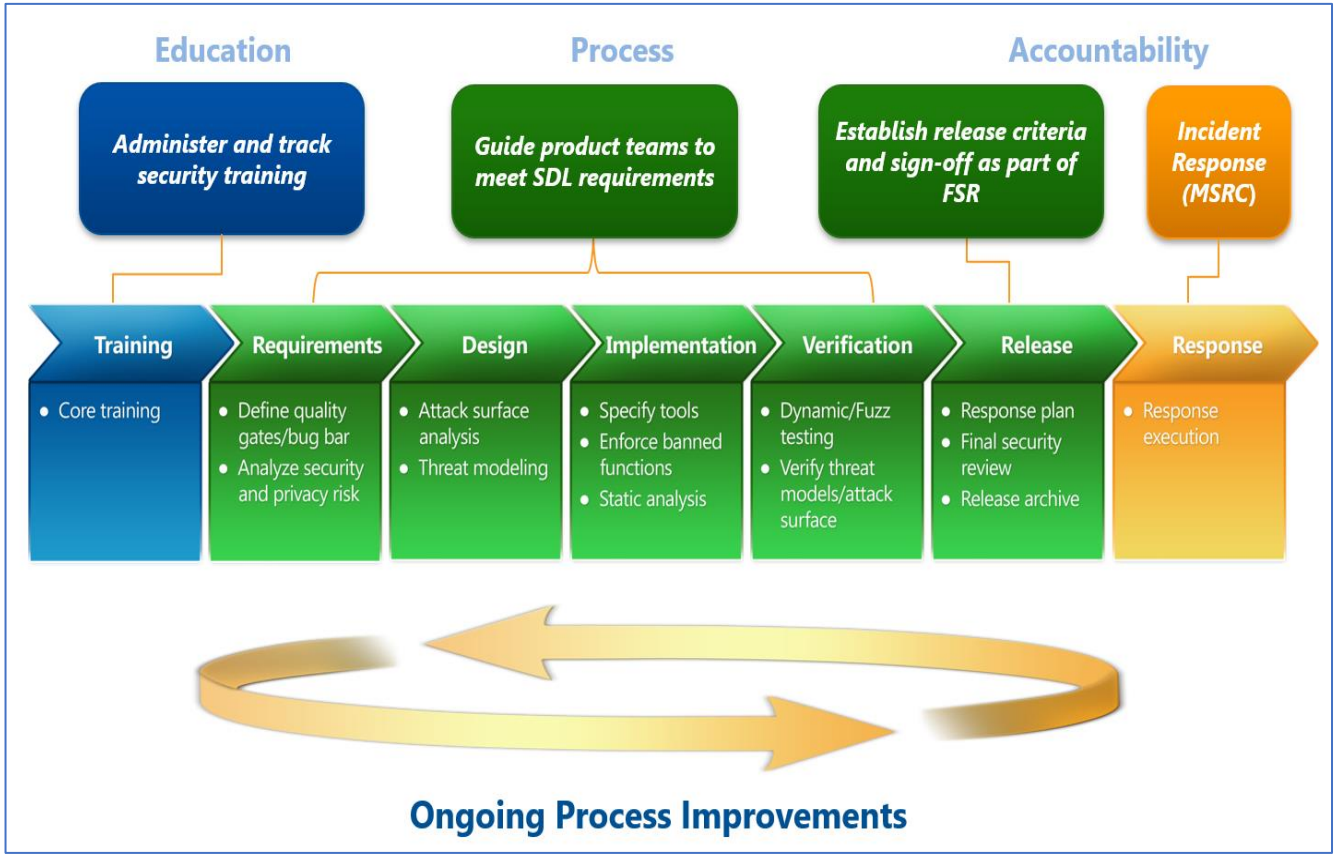
Security Timeline @ Microsoft

- MS-SDL tem sido refinado e adaptado ao longo do tempo
- Segue a evolução de maturidade em segurança de software da Microsoft



Microsoft SDL

- Educação dos desenvolvedores é premissa!
- Muito cuidado com requisitos, design e verificação
- Modelagem de ameaças e redução da superfície de ataque
- Preocupação com resposta a incidentes



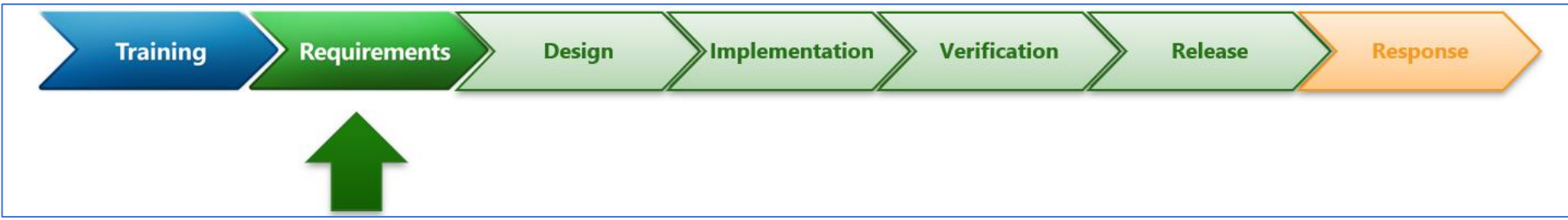
Microsoft SDL

Pre-SDL requirements: Security Training

- Avaliação de conhecimento sobre segurança e privacidade
- Qual o **conteúdo** do treinamento
  - Secure design, desenvolvimento, testes e privacidade
- Qual a **frequência** do treinamento?
  - Quantidade de treinamentos por ano
- Qual a **meta corporativa** do treinamento?
  - Por exemplo, 80% dos times do produto X ou da tribo Y

Microsoft SDL

Phase One: requirements



- Oportunidade de considerar segurança já no início do projeto
- Time/Squad **identifica os requisitos** de segurança e privacidade
- Time/Squad identifica os **líderes/Champions** para segurança e privacidade
- Security Advisor/Champion é designado
- Security Advisor **revisa os planos de projeto**, faz recomendações e pode colocar requisitos adicionais
- Uso obrigatório de **ferramenta bug tracking** com atribuição de tarefas para as vulnerabilidades
- Definição e documentação da "bug bar" de segurança e privacidade

Microsoft SDL

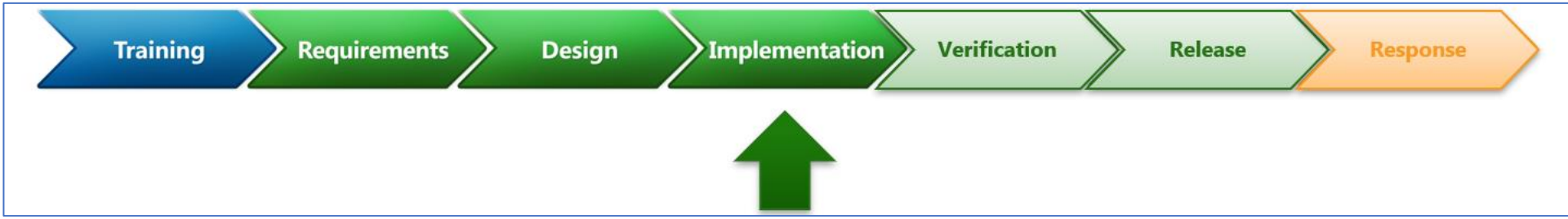
Phase Two: Design

- Define e documenta arquitetura de segurança, identifica os componentes críticos (de maior risco)

- Identifica as técnicas de **design seguro** (camadas, privilégio mínimo, attack surface reduction, etc.)
- Documenta **superfície de ataque** e limita por meio de configurações default
- Define outros critérios de segurança específicos (testado contra XSS, não usa criptografia fraca, cloud, online, etc.)
- **Modelagem de ameaças** como revisão da arquitetura do ponto de vista de segurança, identificando ameaças e mitigações

Microsoft SDL

Phase Three: Implementation



- Revisão completa da implementação usada na determinação de métodos, técnicas e ferramentas para assegurar segurança de implantação e de operação

- Especificação das **ferramentas de build** e opções seguras
- Ferramentas de **análise estática** e controle de falsos positivos
- APIs proibidas
- Proteções do sistema operacional
- Proteções de sistemas web (p.ex., contra OWASP top 10)
- Outras proteções (cloud, VMs, etc.)

Microsoft SDL

Phase Four: Verification

- Iniciada o mais cedo possível, realizada quando houver a código executável

- Planejamento de **resposta a incidentes**, incluindo descoberta de vulnerabilidades
- Reavaliação da **superfície de ataque**
- **Fuzz testing**: arquivos, comandos e controles, comunicação
- **Security Push**: última chance de detectar problemas sérios (code review, pentests, modelagem de novas ameaças)

Phase Five: Release – Response Plan

- Criação da política de suporte técnico e resposta a incidentes

- Plano de **resposta a incidentes** de segurança de software
- Pessoas envolvidas e recursos necessários para responder aos eventos de segurança
- Quem fica de plantão 24x7?
- Envolvidos da resposta: desenvolvimento, infraestrutura, marketing e gerência (executivos?)
- Precisa garantir acesso a todo o pacote de implantação, incluindo componentes de terceiros

Microsoft SDL

Phase Five: Release – Final Security Review

- Revisão de Segurança Final
- Os requisitos de desenvolvimento seguro foram atingidos?
- Não sobraram vulnerabilidades conhecidas sem tratamento?

- A revisão de segurança final
- - Não é apenas um pentest
- - Não é a primeira vez que segurança é revisada
- - Não é o processo de aceitação de risco

- A revisão de segurança final determina se o software release vai ou não ser lançado

Phase Five: Release – Archive

- Plano de resposta a incidentes de segurança
- Completo (detalhado, viável, testado!)
- Arquivado e acessível por quem precisar

- Artefatos de software também são arquivados
- Documentação atualizada de cliente
- Código fonte, modelos de ameaça, objetos gráficos (ícones, etc.) etc.
- Security Signoffs - reconhecimento do estado da segurança do pacote em conformidade com as políticas internas

Microsoft SDL

Response

- Já existe um plano de resposta a incidentes de segurança
- Em caso de incidentes de segurança, executar o plano!

Microsoft SDL

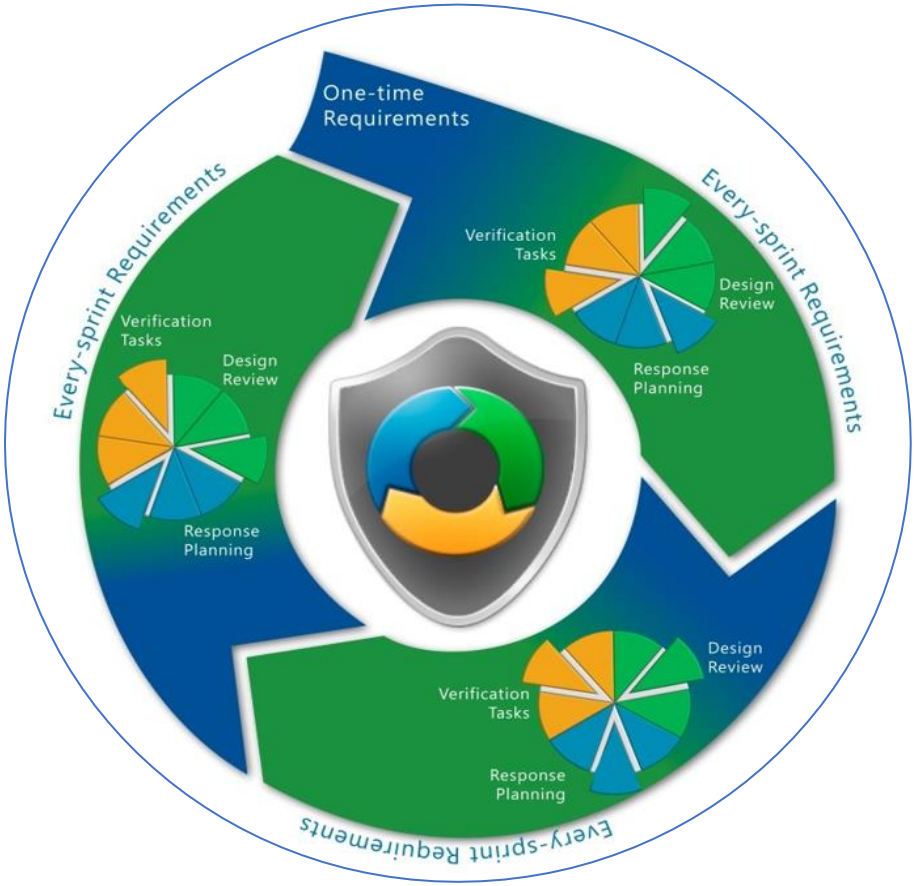
SDL Guidance for Agile Methodology

Requisitos definidos com frequência, não por fase

- Em cada Sprint
- Uma vez, sem repetição

SMART Requirements

- Specific – a target area for improvement
- Measurable – How progress is measured
- Assignable – Who will do it
- Realistic – What can be achieved for real
- Time-bounded – When should it be achieved



Segurança no ciclo de software com MS-SDL

- SDL Optimization Model
- Built by MS to make SDL adoption easier

SAMM Agile guidance

Simplificando a segurança de software ágil

Strategy & Metrics
- Metrics

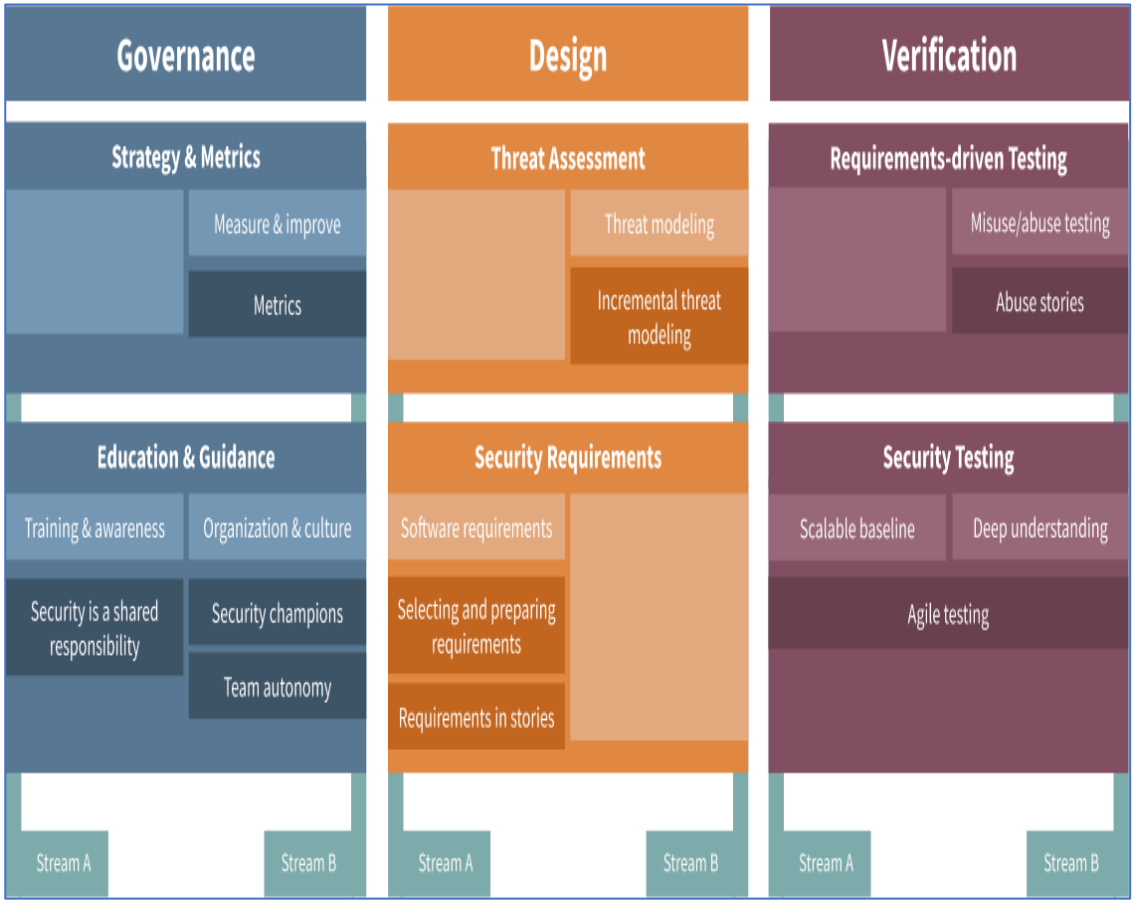
Education & Guidance
- Team autonomy
- Security is a shared responsibility

Threat Assessment
- Incremental threat modeling

Security Requirements
- Selecting and preparing requirements
- Requirements in stories

Security Testing
- Agile testing

Requirements-driven Testing
- Abuse stories



Guildas, Champions e Squads

Security Guilds, Security Champions, Secure squads

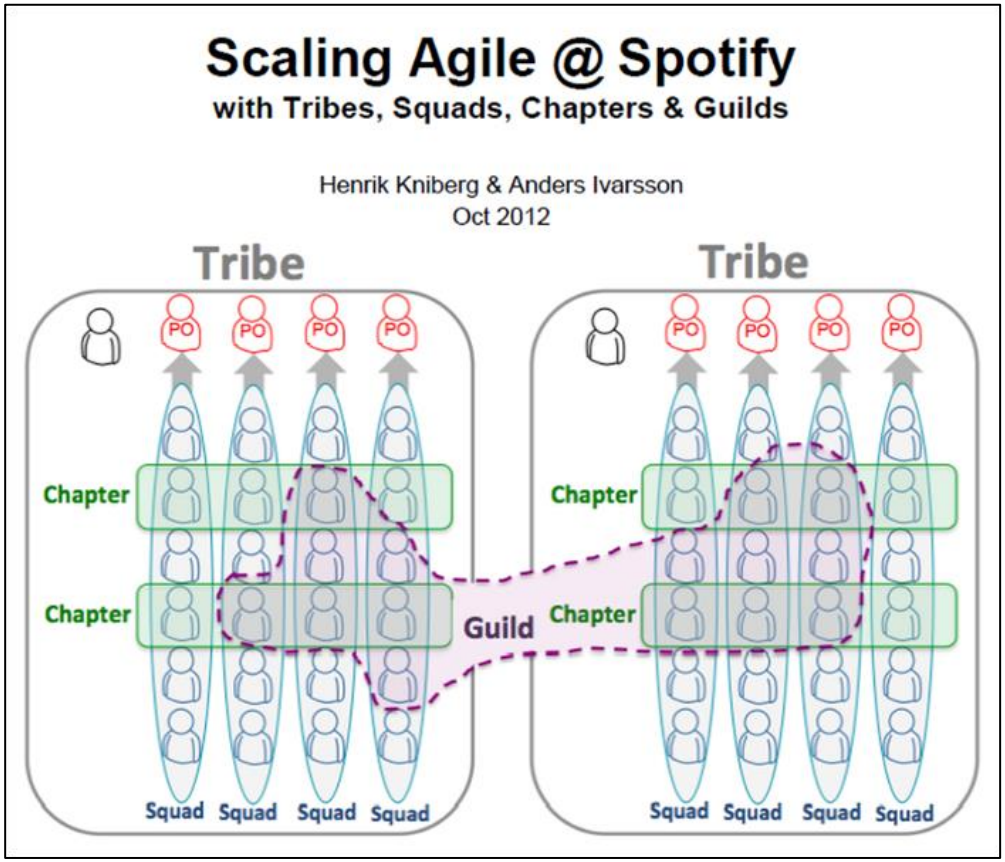
- **Squad**: small cross-functional, self-organized team usually with less than eight members that have end-to-end responsibilities and work together towards a long-term mission
- **Tribe**: Lightweight matrix, a primary dimension focused on product delivery and quality
- **Chapter**: Group formed based on competency areas such as quality assistance, Agile coaching, web development, etc.
- ***Guild: A lightweight community of interest where people across the whole company gather and share knowledge of a specific area. Anyone can join or leave it at anytime***
- Fonte: https://medium.com/pm101/spotify-squad-framework-part-i-8f74bcfcd761

A Guilda de segurança de aplicações

Funções de segurança nos squads de desenvolvimento

- *A guild is a community of members with shared interests. These are a group of people across the organization who want to share knowledge, tools, code, and practices."* **Spotify**

- Mesmo sendo uma comunidade *"livre e solta"*, o *Security Guild* ainda precisa de motivação e engajamento dos membros, além de alguma coordenação de ações



Fonte: https://medium.com/@achardypm/agile-team-organisation-squads-chapters-tribes-and-guilds-80932ace0fdc

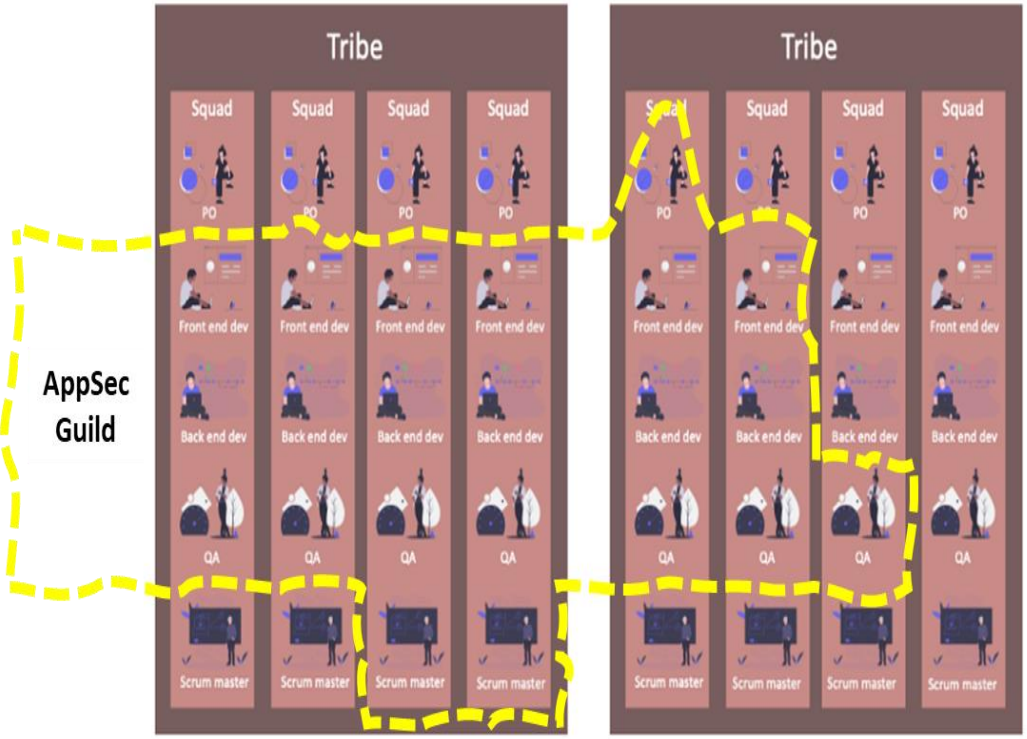
A Guilda de segurança de aplicações

- Estabelecer uma função de Security Champion nos squads empodera a segurança enquanto o Champion mantém e aprimora suas habilidades e competências
- Por que precisamos de uma Guilda de Security Champions?
- A Guilda de Security Champions serve de plataforma para os Champions desenvolverem habilidades e compartilharem conhecimento

Adaptado: https://worth.systems/insight/security-champions

A Guilda de segurança de aplicações

- O Guild de Security Champions ajuda a segurança de software ágil quando:
- Dissemina o pensamento sobre ameaças cibernéticas e a segurança desde o início do desenvolvimento
- Promove a segurança contínua e proativa versus segurança ad-hoc e reativa
- Estabelece melhores práticas e ferramentas comuns para todos os squads
- Garante que a segurança seja uma responsabilidade compartilhada e tratada em cada sprint



Adaptado: https://medium.com/@achardypm/agile-team-organisation-squads-chapters-tribes-and-guilds-80932ace0fdc

O Grupo de Segurança de aplicações

- Application Security Chapter
- Software Security Group (SSG)
- Especialistas em AppSec
- Blue, Red e Purple Teams

- Qual o tamanho do time de AppSec?
- De 0,5% a 3% do número de desenvolvedores

- Por exemplo
  - Para 600 colaboradores
  - SSG tem entre 3 e 18 especialistas

THE SOFTWARE SECURITY GROUP	
SSG SIZE FOR 130 BSIMM11 FIRMS	Average is 13.9, largest is 160, smallest is 1, median is 7
SSG MEMBER-TO-DEVELOPER RATIO FOR 130 BSIMM11 FIRMS	Average of 2.01%, median of 0.63%
SSG MEMBER-TO-DEVELOPER RATIO FOR BSIMM11 FIRMS WITH FEWER THAN 500 DEVELOPERS	Largest is 51.4%, smallest is 0.4%, median is 1.5%
SSG MEMBER-TO-DEVELOPER RATIO FOR BSIMM11 FIRMS WITH MORE THAN 500 DEVELOPERS	Largest is 3.0%, smallest is 0.1%, median is 0.45%

Table 2. The Software Security Group. Statistical data on SSG size helps everyone put their efforts in perspective.

Fonte: https://www.bsimm.com

Parte 1-2 (Laboratório)

Modelagem de ameaças

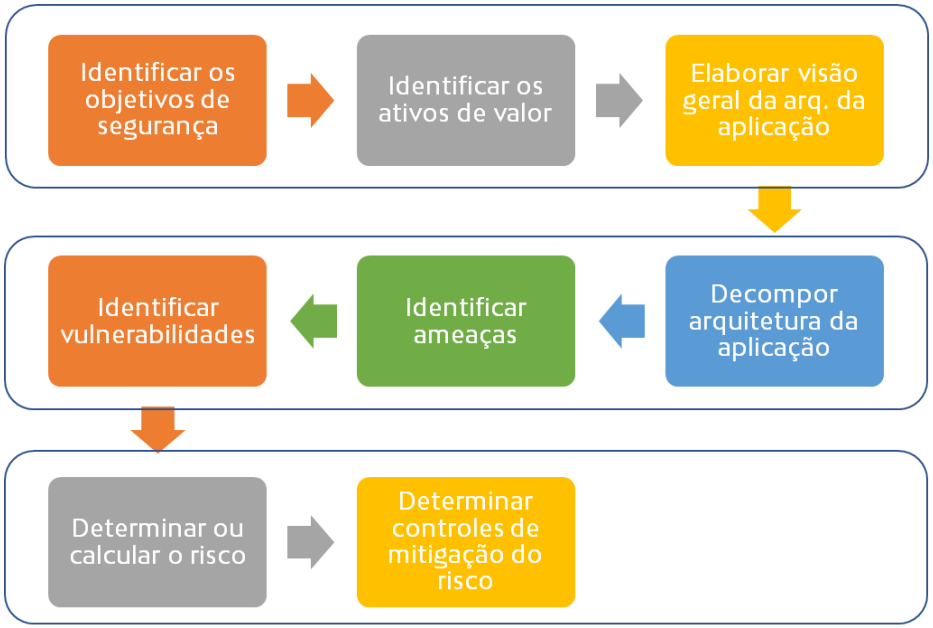
Responde a seguinte pergunta: o que ameaça a minha aplicação?

- A modelagem de ameaças (*Threat Modeling*) é fundamental para o desenvolvimento de software seguro
- O modelo de ameaças permite que os requisitos de segurança sejam refinados e mapeados em salvaguardas e contramedidas
- O modelo de ameaças facilita a quantificação e a priorização dos riscos de segurança
- A modelagem de ameaças de aplicações deve ser conduzida pelo especialista em segurança com apoio do time de desenvolvimento

Fazes da modelagem de ameaças

A Modelagem de ameaças pode seguir 8 passos, embora, muitas vezes, em ambientes simples e grupos pequenos, seja realizada em só um passo

1. Identificar os objetivos de segurança
2. Identificar os ativos de valor
3. Elaborar visão geral da arq. da aplicação
4. Decompor arquitetura da aplicação
5. Identificar ameaças
6. Identificar vulnerabilidades
7. Determinar ou calcular o risco (opcional)
8. Determinar controles



Quando fazer a modelagem de ameaças

Modelagem de ameaças é uma consequência da avaliação da arquitetura do software e orienta a revisão de projeto e os planos de testes de segurança

Nos projetos de software com métodos ágeis, a modelagem de ameaças ocorre primeiro no Sprint0 ou Sprint Planning

Sprint 0/ Sprint Planning	Sprints 1 through N	Release Definition of Done (DoD)
Starting and completing the release threat model (TM) based on the overall design	Update the TM only if new user stories or source code changes invalidated the original TM.	Make sure that the TM reflects the system state, is up-to-date and controls are implemented.

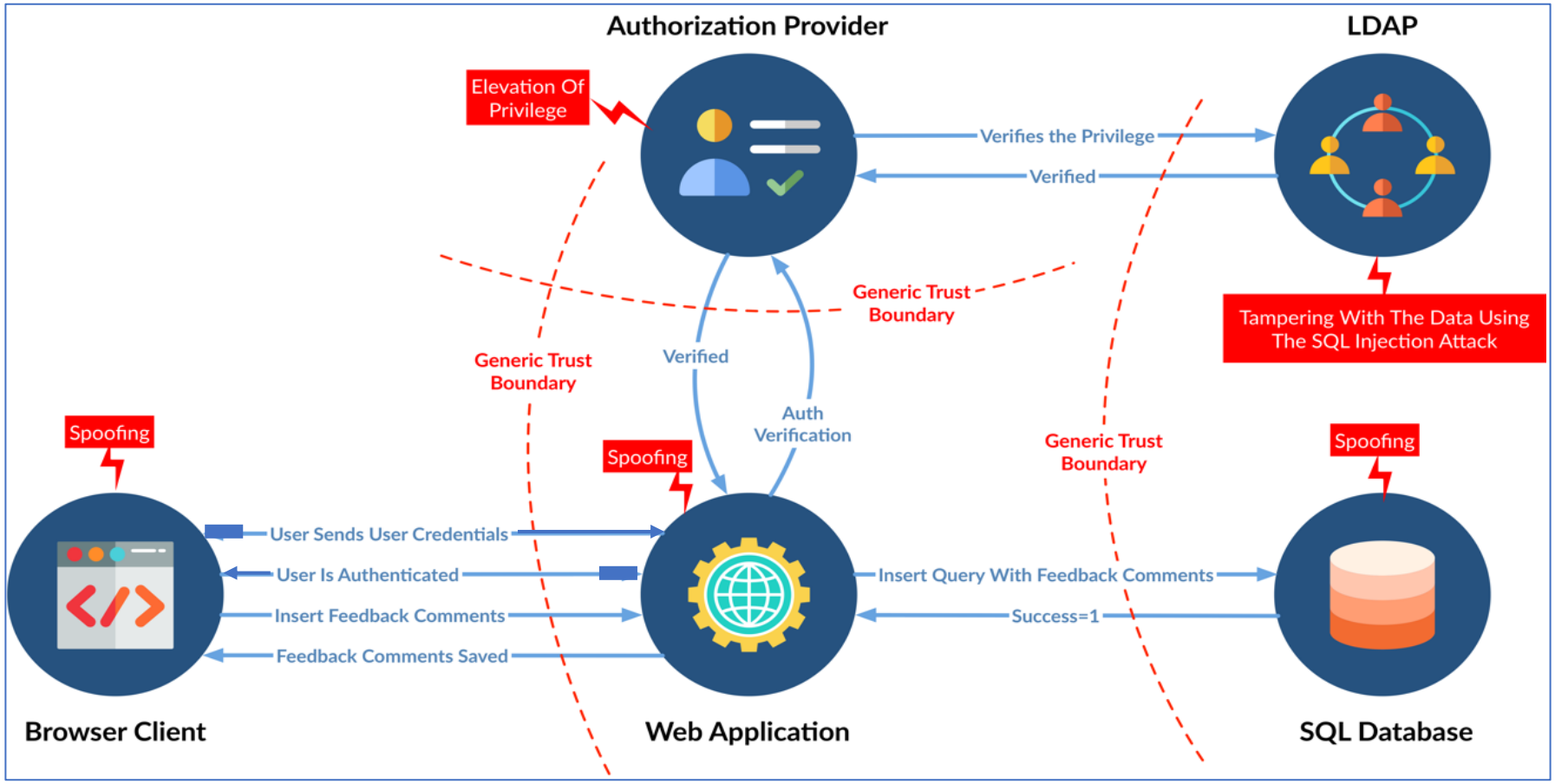
Fonte: SafeCode. Tactical Threat Modeling.
https://safecode.org/wp-content/uploads/2017/05/SAFECode_TM_Whitepaper.pdf

Modelagem de ameaças com DFD

Construção de modelos de ameaças com diagramas de fluxos de dados

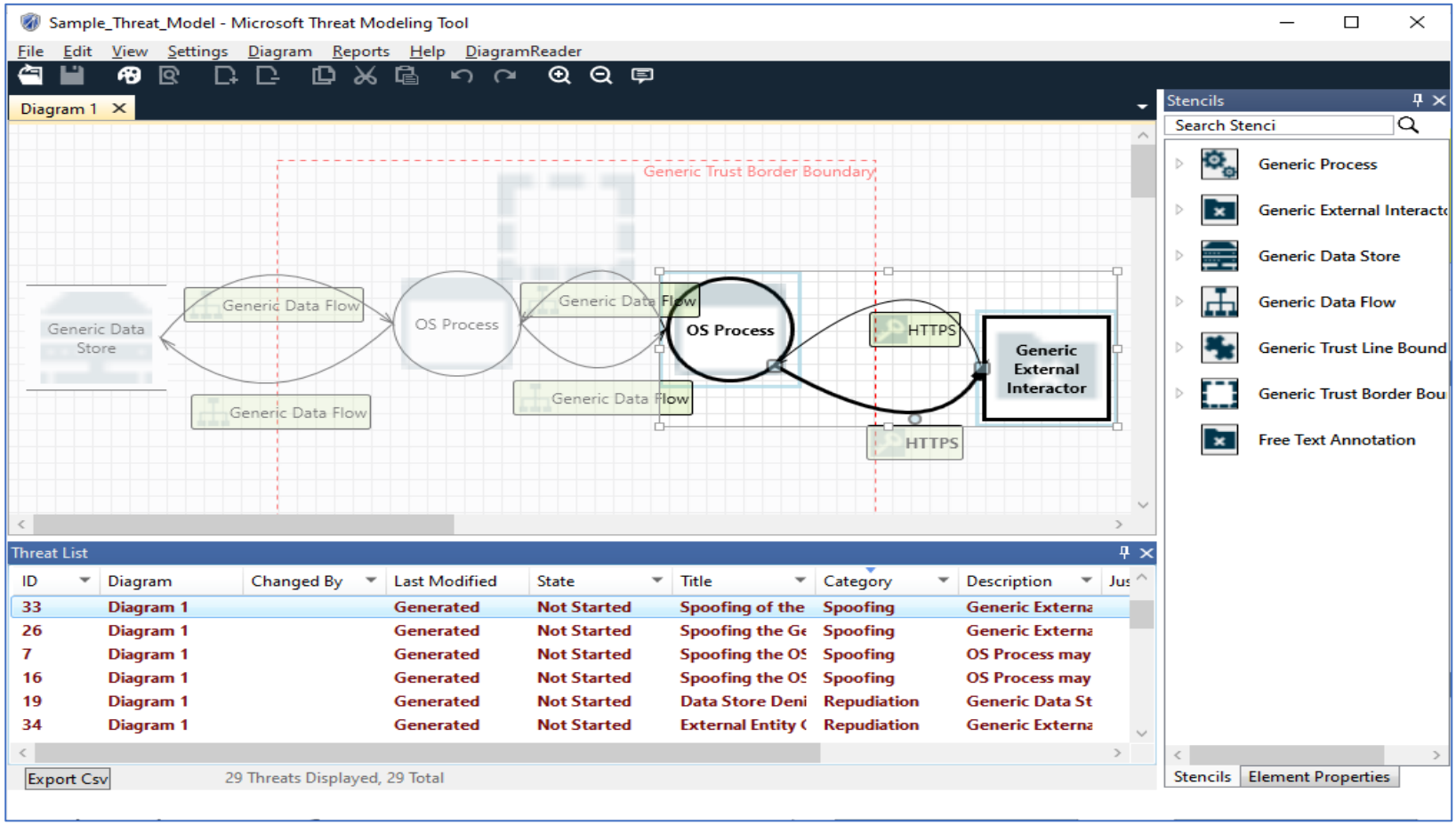
- Identifique e documente as partes do aplicação
- Isso inclui as seguintes partes:
- **Entidades externas**: usuários do sistema, processos externos
- **Processos**: tarefa ou atividade (p.ex., validar dados de entrada, processar cartão de crédito, componente de autenticação, componente de renderização no navegador, etc.)
- **Armazenamento (persistente)**: arquivos de configuração, documentos, bancos de dados
- **Fluxos de dados**: como os dados transitam (strings de pesquisa, cabeçalhos IP, confirmação do usuário, requisição HTTP)
- **Limites de confiança/privilégio**: onde os dados passam entre áreas de diferentes níveis de confiança (entre a entrada do usuário e o aplicativo web, entre o processo do aplicativo web e o banco de dados)

Exemplo Threat Modeling



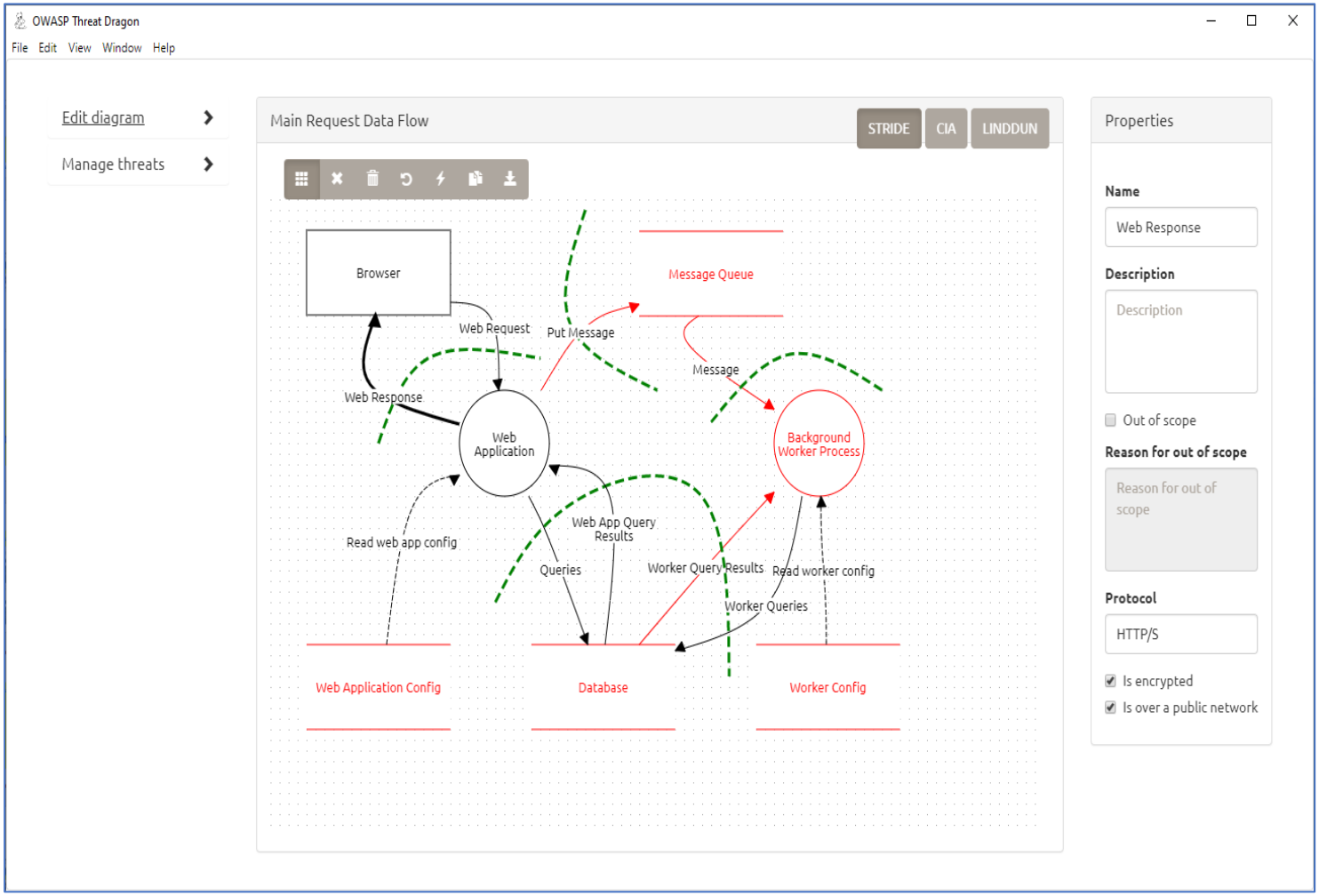
Fonte: SafeCode. https://safecode.org/wp-content/uploads/2017/05/SAFECode_TM_Whitepaper.pdf

Microsoft Threat Modeling Tool



Fonte: https://docs.microsoft.com/pt-br/azure/security/develop/threat-modeling-tool

OWASP Threat Dragon



Fonte: https://owasp.org/www-project-threat-dragon

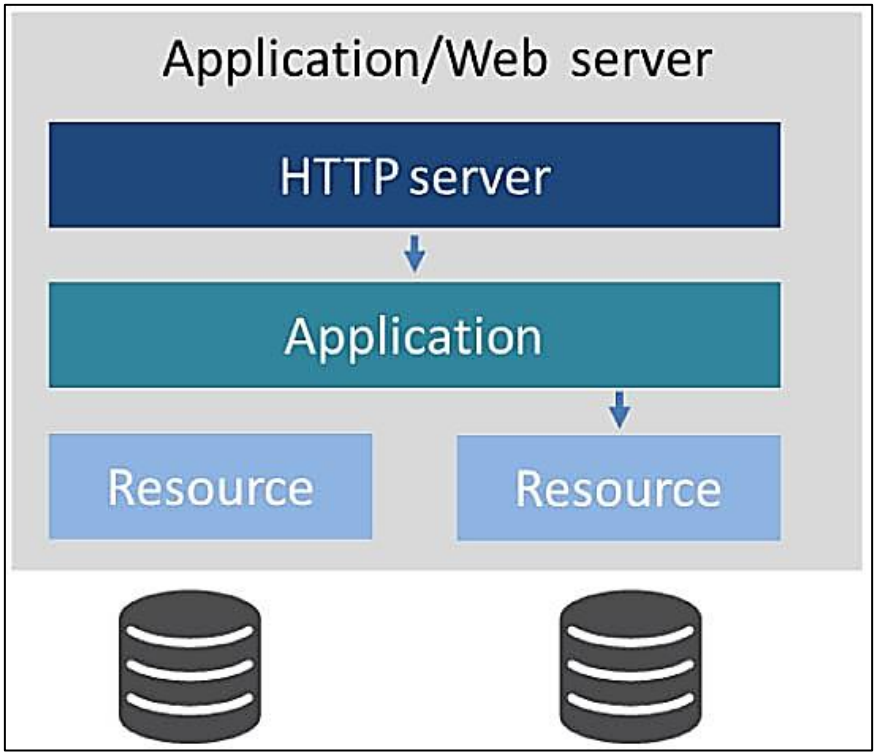
Varredura de vulnerabilidades HTTP

Varredura de vulnerabilidades HTTP

- Quem é responsável pela segurança do servidor web?
  - É uma questão de infraestrutura?
  - É uma questão de desenvolvimento?

- A figura mostra uma visão “invertida” da arquitetura de alto nível de uma aplicação: a camada superior é o servidor web

- A superfície de ataque da aplicação começa por aquilo que está mais exposto: o servidor web (a.k.a. HTTP Server)

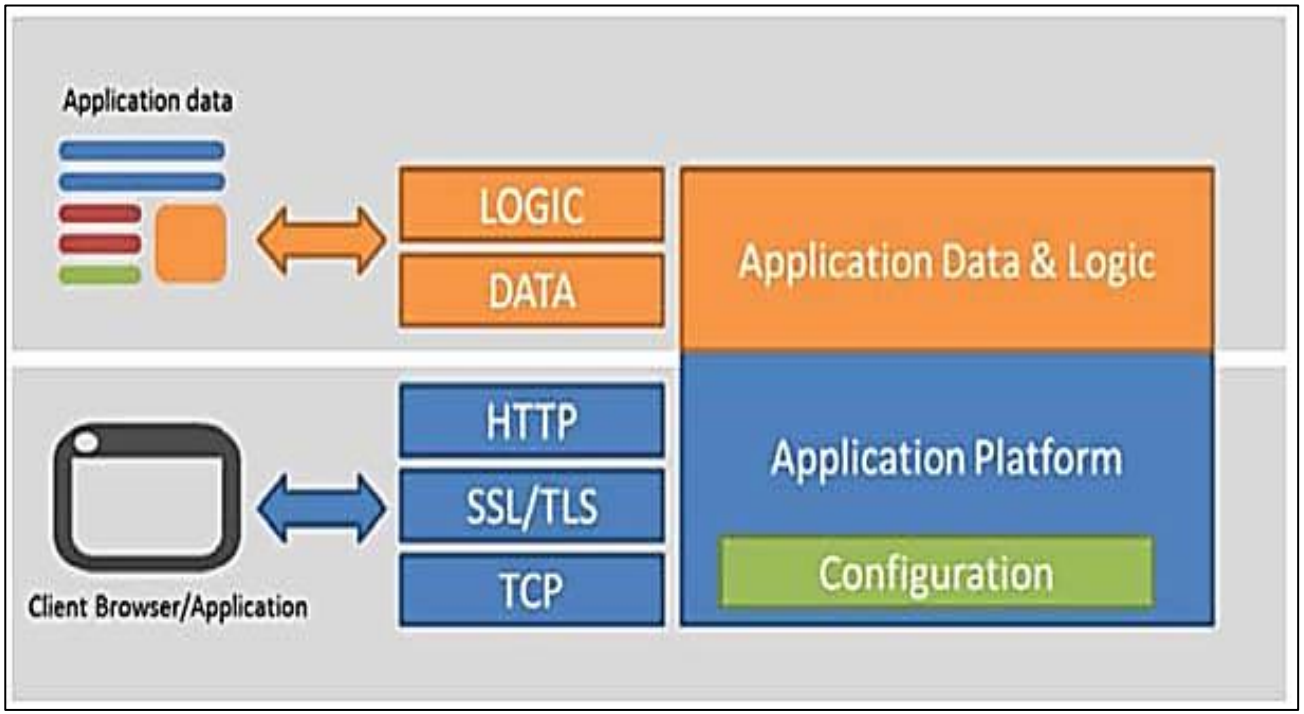


Fonte: Web Application Security is a Stack: How to CYA (Cover Your Apps) Completely by Lori Mac Vittie

Varredura de vulnerabilidades HTTPS

As superfícies de ataque estão relacionadas aos clientes da aplicação

- Dados da aplicação
- Navegador web



Fonte: Web Application Security is a Stack: How to CYA (Cover Your Apps) Completely by Lori Mac Vittie

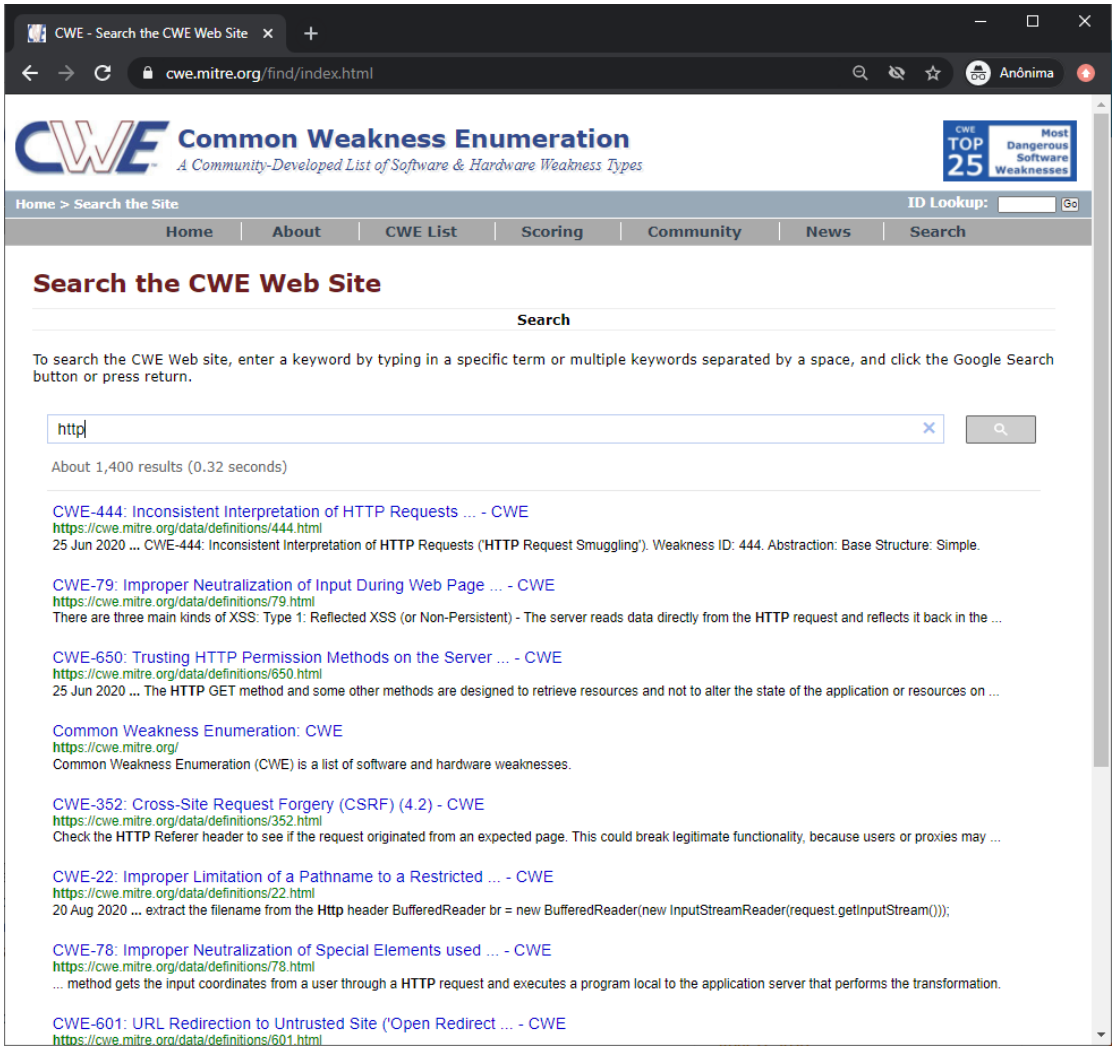
Varredura de vulnerabilidades HTTPS

Pergunta:

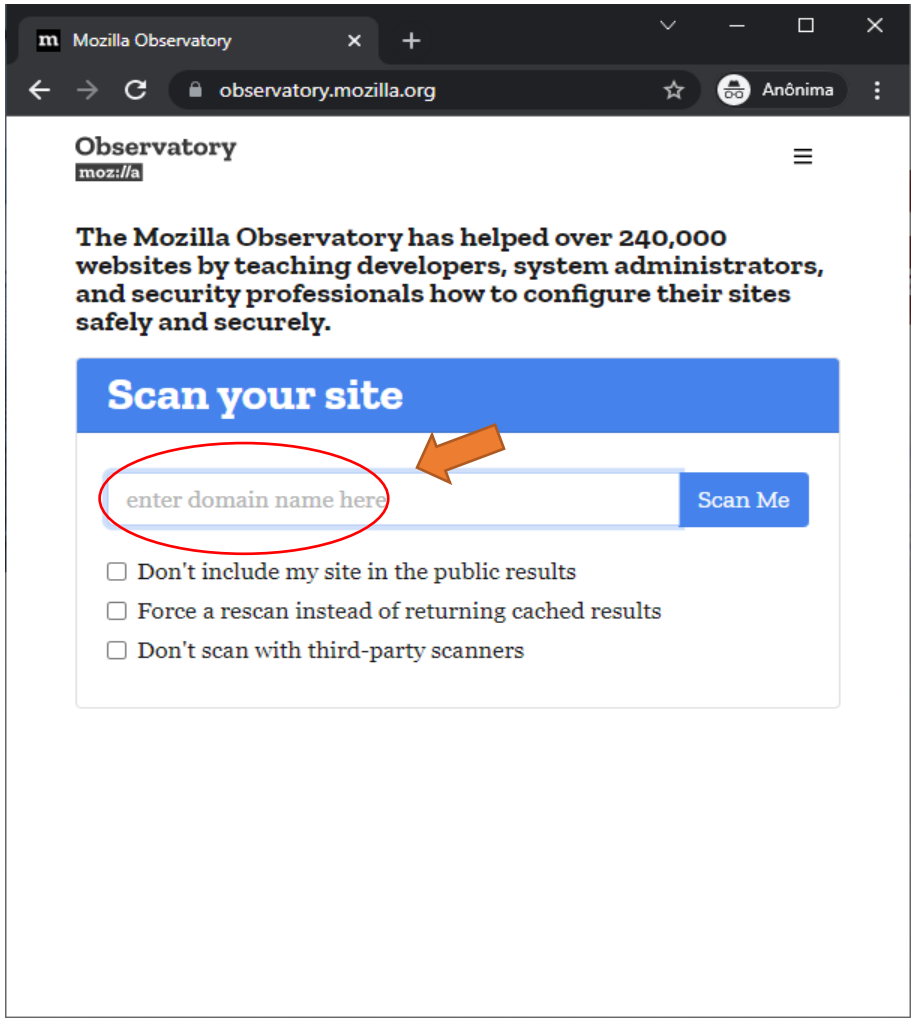
- Existem fraquezas ou vulnerabilidades do HTTP que afetam as aplicações ?

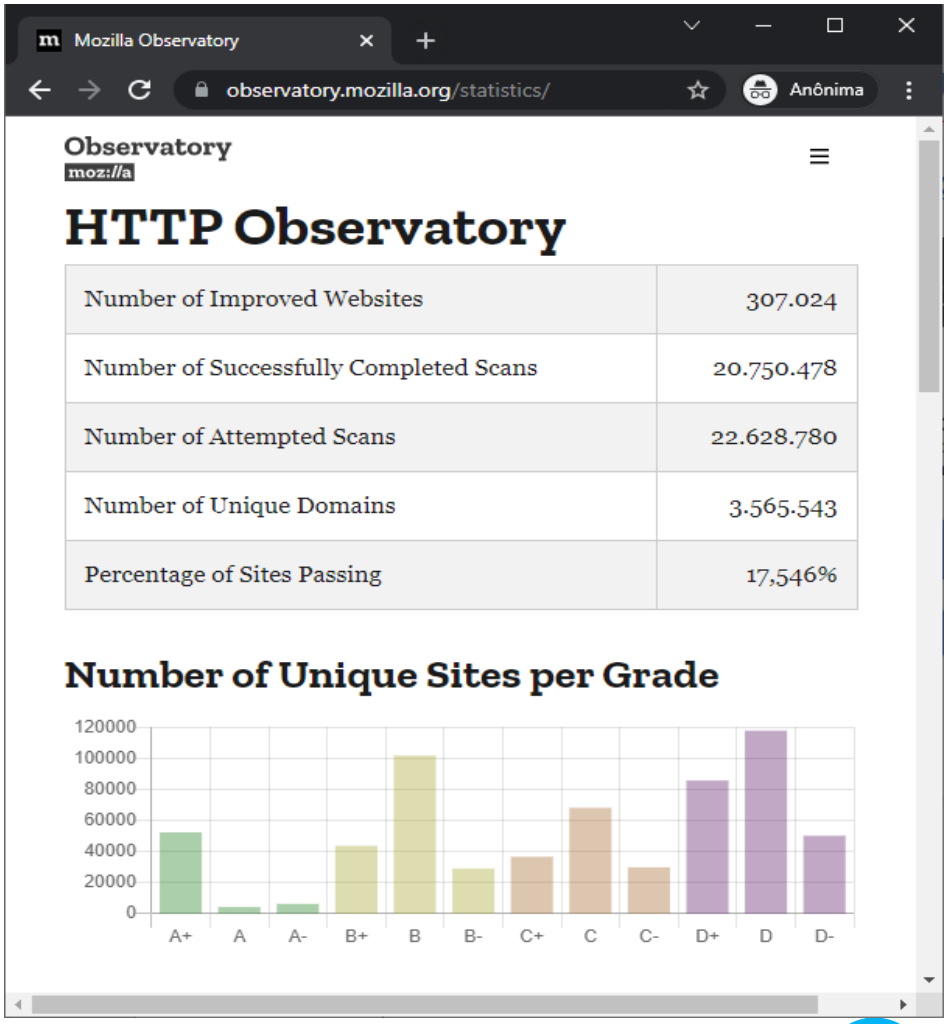
Resposta:

- Sim!
- Busca por “HTTP” no CWE https://cwe.mitre.org/find/index.html



Varredura de vulnerabilidades HTTPS





Obrigado!