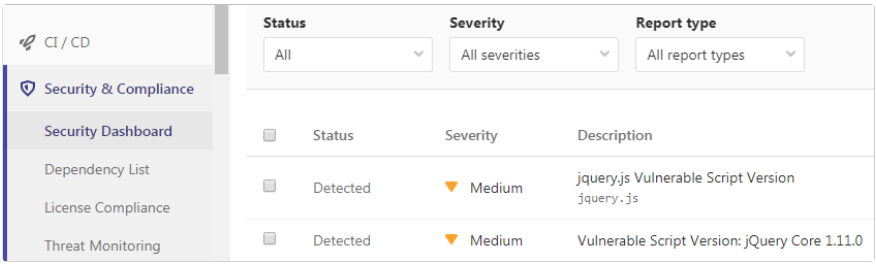
INTEGRANDO O SYHUNT AO GITLAB

As informações contidas neste documento se aplicam a **versão 6.8.6** do Syhunt Hybrid.

INTRODUÇÃO

Iniciar varreduras do Syhunt a partir de um script CI YML no GitLab é simples e fácil, permitindo integrar as ferramentas de teste de segurança Syhunt Dynamic, Syhunt Code e Syhunt Mobile ao seu processo de entrega contínua e painel de segurança, agendar varreduras e mais. Você também pode configurar rastreadores de problemas no Syhunt, permitindo que vulnerabilidades sejam enviadas para a área de problemas de projetos.



O exemplo a seguir de script CI YML do Gitlab verificará o código-fonte do repositório atual, falhando o job se forem identificadas vulnerabilidades médias ou altas. Além disso, ele anexa um relatório de vulnerabilidade em PDF aos artefatos da tarefa de pipeline e adiciona as vulnerabilidades identificadas ao painel de segurança do GitLab.

```
syhunt_test:
  script:
          - Start-CodeScan -pfcond 'fail-if:risk=mediumup' -output 'report.pdf' -outputex 'gl-sast-repo
  artifacts:
    reports:
      sast: gl-sast-report.json
    paths:
    - report.pdf
    when: on_failure
  tags:
   - syhunt
```

ATIVANDO O SYHUNT RUNNER

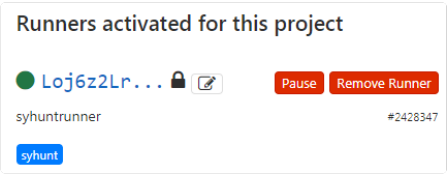
O Syhunt Runner é um serviço de CI que receberá os comandos de varredura, executando as varreduras e comunicando os resultados da análise da segurança com o GitLab.

IMPORTANTE: Por motivos de segurança e desempenho, é recomendável que o Runner seja instalado em uma máquina virtual separada ou em uma máquina física dedicada.

1. Primeiro, entre nas configurações do seu projeto GitLab e acesse as opções de CI (Continuous Integration):
   1. Clique em **Settings** (Configurações)
   2. Clique em **CI / CD**
   3. Expanda **Runners**
   4. Role para baixo até **Set up a specific Runner manually** (Configurar um Runner manualmente)
   5. Salve o token de registro em um local seguro. Você precisará mais tarde abaixo.

2. Instale com as configurações padrão o Git for Windows, que pode ser baixado em https://gitforwindows.org/
3. Instale com as configurações padrão o Syhunt Hybrid (**syhunt-hybrid-6.9.14.1.exe**)
4. Faça o download e execute o Syhunt Runner (**syhunt-runner-14.8.0.exe**). Após iniciar a instalação, você será solicitado a fornecer as informações de registro do Runner.
   1. Cole o token de registro que você copiou no campo de texto do token e clique em Next (Avançar) para continuar e concluir a instalação.

Após concluir a instalação, retorne ao GitLab e à seção **Runners** em que estávamos e pressione F5 para atualizar a tela. O syhuntrunner agora deve estar registrado para este projeto. Você o verá listado em **Runners activated for this project** no final da página, como mostrado na imagem abaixo.



O Syhunt já está pronto para ser chamado de scripts CI YML! Veja exemplos abaixo

ADICIONANDO O SYHUNT AO SEU SCRIPT CI YML

Se você não possui um arquivo CI YML em seu repositório, acesse **Project overview** (Visão Geral do Projeto) e clique **Set up CI/CD** (Configurar CI/CD). Isto criará um arquivo chamado .gitlab-ci.yml no repositório.

Exemplo de SAST - O exemplo a seguir de script CI YML do Gitlab verificará o código-fonte do repositório atual, falhando o job se forem identificadas vulnerabilidades médias ou altas. Além disso, ele anexa um

relatório de vulnerabilidade em PDF aos artefatos da tarefa de pipeline e adiciona as vulnerabilidades identificadas ao painel de segurança do GitLab.

```
syhunt_test:
  stage: test
  script:
          - Start-CodeScan -pfcond 'fail-if:risk=mediumup' -output 'report.pdf' -outputex 'gl-sast-repo
  artifacts:
    reports:
      sast: gl-sast-report.json
    paths:
    - report.pdf
    when: on_failure
  tags:
   - syhunt
```

Exemplo de DAST - O exemplo a seguir de script CI YML do Gitlab verificará a aplicação web no ar depois de entrar em produção, falhando o job se forem identificadas vulnerabilidades médias ou altas. Além disso, ele anexa um relatório de vulnerabilidades em PDF aos artefatos da tarefa de pipeline e adiciona as vulnerabilidades identificadas ao painel de segurança do GitLab.

```
production:
stage: deploy
  script:
          - Start-DynamicScan -target 'www.productionurl.com' -pfcond 'fail-if:risk=mediumup' -output '
  artifacts:
    reports:
      sast: gl-dast-report.json
    paths:
    - report.pdf
    when: on_failure
  tags:
   - syhunt
only:
  - tags
```

Exemplos adicionais:

```
# Exemplo de SAST - Analisar diretório / repositório local
Start-CodeScan -pfcond "fail-if:risk=mediumup"

# Exemplo de SAST - Analisar um repositório GIT remoto
$MyProject = @{
  target = 'https://github.com/syhunt/vulnphp.git';
  branch = 'main';
  pfcond = 'fail-if:risk=mediumup';
  output = 'report.pdf'
}
Start-CodeScan @MyProject
```

```
# Exemplo de DAST - Analisar um URL
$MyWebsite= @{
  target = 'https://www.somewebsite.com';
  pfcond = 'fail-if:risk=mediumup';
  output = 'report.pdf'
}
Start-DynamicScan @MyWebsite
```

INTEGRANDO O SYHUNT AO PAINEL DE SEGURANÇA DO GITLAB

Syhunt gerará um arquivo de relatório de vulnerabilidade compatível com GitLab se o parâmetro outputex estiver simplesmente definido como:

- SAST: **gl-sast-report.json** ou **qualquernome.gls.json**
- DAST: **gl-dast-report.json** ou **qualquernome.gld.json**

Lembre-se de adicionar o nome do arquivo ao artifacts.reports.sast ou artifacts.reports.dast do seu arquivo CI YML file como mostrado no primeiro exemplo acima.

No momento, o GitLab adicionará apenas pipelines bem-sucedidos ao painel de segurança e às áreas de relatório de vulnerabilidade de um projeto. Isto significa que se você quiser que todas as vulnerabilidades identificadas sejam exibidas no painel, você deve omitir o parâmetro pfcond da linha de script que chama o Syhunt. Como alternativa, se você fornecer uma condição de aprovação/reprovação alta e nenhuma vulnerabilidade alta for encontrada, todas as vulnerabilidades identificadas de medium a info serão exibidas no painel.

Isto não se aplica à aba Segurança de Pipelines, a aba Segurança sempre exibirá todas as vulnerabilidades seja o status sucesso ou falha - no entanto, se o pipeline falhou devido a uma condição de aprovação/reprovação correspondente, você poderá ver o GitLab mostrar inconsistentemente 0 (zero)

vulnerabilidades enquanto lista logo abaixo as vulnerabilidades identificadas. Trata-se de um bug menor não resolvido pela equipe do GitLab.

Dica importante: Se esta é a primeira varredura em uma base de código grande é recomendável analisar sua aplicação sem a integração do painel ativada para garantir que você não tenha um grande número de vulnerabilidades. Se um grande número de vulnerabilidades for encontrado, aprimore o estado de segurança da aplicação corrigindo o lote inicial de vulnerabilidades relatadas pelo Syhunt e só então ative a integração com o painel.

START-DYNAMICSCAN - INICIANDO UMA VARREDURA DINÂMICA

O Syhunt Dynamic deve ser iniciado através da função Start-DynamicScan(). Os seguintes parâmetros devem ser fornecidos ao chamar a função Start-DynamicScan():

- **target** (obrigatório) - o URL alvo a ser analisado (ex. http://www.somesite.com)
- **huntmethod** (opcional) - o Método de Varredura a ser usado durante a análise. Se omitido, o método padrão será usado.
- **pfcond** (opcional) - permite que o script falhe com o código de saída adequado se uma determinada condição for atendida. Veja abaixo uma lista das condições de aprovação / reprovação disponíveis.
- **tracker** (opcional) - o nome do rastreador criado anteriormente ou um rastreador gerado dinamicamente para o qual será enviado um resumo das vulnerabilidades identificadas ao final da varredura. Exemplos
- **output** (opcional) - permite definir um nome de arquivo de relatório (por exemplo, report.pdf ou report.html).
- **outputex** (opcional) - permite definir um segundo nome de arquivo de saída (por exemplo, export.json).
- **verbmode** (opcional) - $ false por padrão. Se alterado para true, ativa o modo detalhado, permitindo que informações de erro e informações básicas sejam adicionadas ao console.
- **genrep** (opcional) - $ true por padrão. Se alterado para false, o Syhunt não gerará um arquivo de relatório.
- **redirIO** (opcional) - $ true por padrão. Se alterado para false, informações de status em tempo-real do scanner Syhunt não serão redirecionadas para o console.
- **timelimit** (opcional) - define o tempo máximo da varredura (padrão: sem limite). Caso o tempo seja atingido, a varredura é cancelada. Exemplos: 1d, 3h, 2h30m, 50m

Ao usar os parâmetros output ou outputex, todos os formatos de saída suportados pelo Syhunt estão disponíveis. O relatório ou exportação será salvo no diretório de trabalho atual, a menos que seja fornecido um nome de caminho completo.

Exemplos:

```
# Exemplo 1 - Analisar um URL com uma única linha
Start-DynamicScan -target 'https://www.somewebsite.com' -pfcond 'fail-if:risk=mediumup'

# Exemplo 2 - Analisar um URL
$MyWebsite= @{
  target = 'https://www.somewebsite.com';
  pfcond = 'failifriskmedium';
  output = 'report.pdf'
}
Start-DynamicScan @MyWebsite
```

START-CODESCAN - INICIANDO UMA VARREDURA DE CÓDIGO-FONTE

O Syhunt Code deve ser iniciado através da função Start-CodeScan(). Os seguintes parâmetros podem ser fornecidos ao chamar a função Start-CodeScan(), sendo todas eles opcionais:

- **target** - o URL de um repositório GIT ou um diretório ou arquivo de código-fonte local a ser verificado. Se o parâmetro target for omitido, o diretório de trabalho atual será verificado.
- **branch** - a ramificação do repositório a ser analisada. Se o parâmetro branch for omitido, o branch padrão será analisado.
- **huntmethod** - o Método de Varredura a ser usado durante a análise. Se omitido, o método padrão será usado.
- **pfcond** - permite que o script falhe com o código de saída adequado se uma determinada condição for atendida. Veja abaixo uma lista das condições de aprovação / reprovação disponíveis.
- **output** - permite definir um nome de arquivo de relatório (por exemplo, report.pdf ou report.html).
- **outputex** - permite definir um segundo nome de arquivo de saída (por exemplo, export.json).
- **verbmode** - $ false por padrão. Se alterado para true, ativa o modo detalhado, permitindo que informações de erro e informações básicas sejam adicionadas ao console.
- **genrep** - $ true por padrão. Se alterado para false, o Syhunt não gerará um arquivo de relatório.
- **redirIO** - $ true por padrão. Se alterado para false, informações de status em tempo-real do scanner Syhunt não serão redirecionadas para o console.
- **timelimit** (opcional) - define o tempo máximo da varredura (padrão: sem limite). Caso o tempo seja atingido, a varredura é cancelada. Exemplos: 1d, 3h, 2h30m, 50m

Ao usar os parâmetros output ou outputex, todos os formatos de saída suportados pelo Syhunt estão disponíveis. O relatório ou exportação será salvo no diretório de trabalho atual, a menos que seja fornecido um nome de caminho completo.

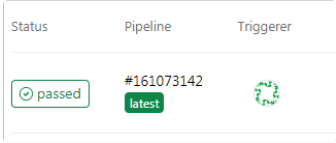
Exemplos:

```
# Exemplo 1 - Analisando o diretório / repositório atual
Start-CodeScan -pfcond "fail-if:risk=mediumup"

# Exemplo 2 - Analisando um projeto GIT remoto
Start-CodeScan -target "https://github.com/someuser/somerepo.git" -huntmethod "normal" -pfcond "fail-if

# Exemplo 3 - Analisando um diretório local específico
Start-CodeScan -target "C:\www\" -huntmethod "normal" -pfcond "fail-if:risk=mediumup"
```

CONDIÇÕES DE APROVAÇÃO / REPROVAÇÃO



A seguir estão as condições de aprovação / reprovação atualmente suportadas pelo Syhunt:

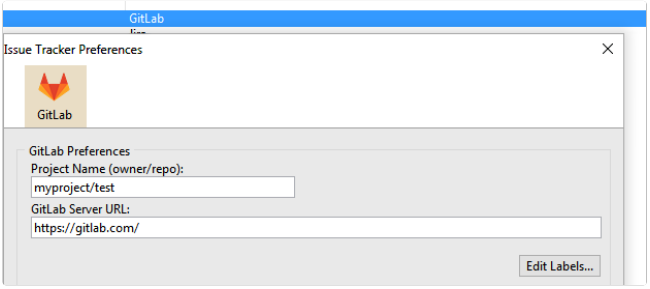
- `fail-if:risk=high` - Falha se for encontrada uma vulnerabilidade ou ameaça de alto risco
- `fail-if:risk=mediumup` - Falha se for encontrada uma vulnerabilidade ou ameaça de risco Médio ou Alto
- `fail-if:risk=lowup` - Falha se for encontrada uma vulnerabilidade ou ameaça de risco Baixo, Médio ou Alto

CONFIGURAÇÕES AVANÇADAS DO RUNNER

Atualize o valor de **concurrent** para Runners em C:\SyhuntRunner\config.toml para permitir vários jobs simultâneos, conforme detalhado em advanced configuration details.

INTEGRANDO COM O ISSUES DO GITLAB

Configurar um rastreador de problemas é uma tarefa fácil e as vulnerabilidades podem ser enviadas para um projeto específico com o clique de um botão.



Em primeiro lugar, se ainda não o fez, você precisa criar um token de acesso pessoal com permissão de API no GitLab:

1. Acesse o GitLab.
2. Na parte superior do canto direito, clique no seu avatar e selecione **Settings** (Configurações).
3. No menu User Settings, selecione **Access Tokens** (Tokens de Acesso).
4. Escolha um nome e uma data opcional de expiração para o token a ser criado.
5. Escolha o escopo API.
6. Clique no botão **Create personal access token** (Criar token pessoal de acesso).
7. Guarde o seu personal access token em algum lugar seguro. Uma vez que você deixe ou atualize a página, você não será capaz de acessar o token novamente.

Finalmente, você deve adicionar um rastreador do tipo GitLab:

1. Clique no ícone Issue Trackers na barra da tela inicial. A tela de rastreadores de problemas irá abrir.
2. Clique no ícone Add Tracker na barra da tela de reastreadores e escolha a opção do menu Add tracker: GitLab (Adicionar rastreador: GitLab).
3. Digite um nome de referência para o novo rastreador (como NomeDoMeuProjeto) e pressione **OK**. Uma janela de preferências irá abrir.
4. Digite o nome do projeto no GitLab. O formato precisa ser **usuario/repositorio**.
5. Digite o URL do servidor GitLab. Por exemplo: `https://gitlab.com/` ou URL do seu próprio servidor.
6. Preencha seu personal access token do GitLab e clique no botão **OK**.

O rastreador está pronto! Clique com o botão direito do mouse no item que você acabou de editar na lista e clique na opção **Submit Test Issue** (Enviar Problema de Teste). Se você configurou tudo corretamente, um problema de teste deve ser criado em `https://[gitlab_server]/[owner]/[repo]/issues`. Caso contrário, você verá uma mensagem de erro indicando o que precisa ser feito.

Para obter mais detalhes sobre como enviar vulnerabilidades ao rastreador GitLab, consulte: Enviando vulnerabilidades para um rastreador.