NOME

wapiti- Um scanner de vulnerabilidade de aplicativo da web em Python

SINOPSE

wapiti-u *BASE_URL* [opções]

DESCRIÇÃO

Wapiti permite que você audite a segurança de seus aplicativos da web.

Ele executa varreduras de "caixa preta", ou seja, não estuda o código-fonte do aplicativo, mas varre as páginas da web do aplicativo implantado, procurando scripts e formulários onde pode injetar dados.

Depois de obter essa lista, o Wapiti atua como um difusor, injetando cargas úteis para ver se um script está vulnerável.

Wapiti é útil apenas para descobrir vulnerabilidades: não é uma ferramenta de exploração. Alguns aplicativos bem conhecidos podem ser usados para a parte de exploração, como o sqlmap recomendado.

RESUMO DAS OPÇÕES

Aqui está um resumo das opções. É essencialmente o que você obterá quando lançar o Wapiti sem qualquer argumento. Mais detalhes sobre cada opção podem ser encontrados nas seções a seguir.

ESPECIFICAÇÃO DO ALVO:

- **-u** *URL*
- **--scope** {página, pasta, domínio, url}

ESPECIFICAÇÃO DE ATAQUE:

- **-m** *MODULES_LIST*
- **--list-modules**
- **-l** *NÍVEL*

OPÇÕES DE PROXY E AUTENTICAÇÃO:

- **-p** *PROXY_URL*
- **-a** *CREDENCIAIS*
- **--auth-type** {básico, resumo, kerberos, ntlm}
- **-c** *COOKIE_FILE*

OPÇÕES DE SESSÃO:

- **--skip-crawl**

- **--resume-crawl**
- **--flush-attacks**
- **--flush-session**

SINTONIA DE VARREDURA E ATAQUES:

- **-s** *URL*
- **-x** *URL*
- **-r** *PARÂMETRO*
- **--skip** *PARÂMETRO*
- **-d** *PROFUNDIDADE*
- **--max-links-per-page** *MAX_LINKS_PER_PAGE*
- **--max-files-per-dir** *MAX_FILES_PER_DIR*
- **--max-scan-time** *MAX_SCAN_TIME*
- **--max-parameters** *MAX*
- **-S**, **--scan-force**{paranóico, sorrateiro, educado, normal, agressivo, insano}

OPÇÕES DE HTTP E REDE:

- **-t** *SEGUNDOS*
- **-H** *CABEÇALHO*
- **-A** *AGENTE*
- **--verify-ssl** {0,1}

OPÇÕES DE SAÍDA:

- **--color**
- **-v** *NÍVEL*

OPÇÕES DE RELATÓRIO:

- **-f** {json, html, txt, openvas, vulneranet, xml}
- **-o** *OUPUT_PATH*

OUTRAS OPÇÕES:

- **--no-bugreport**
- **--version**
- **-h**

ESPECIFICAÇÃO DE ALVO

- **-u**, **--url** *URL*
  O URL que será usado como base para a varredura. Cada URL encontrado durante a varredura será verificado em relação ao URL base e ao escopo de varredura correspondente (consulte --scope para obter detalhes).
  Este é o único argumento necessário. A parte do esquema do URL deve ser http ou https.

- **--scope** *ESCOPO*
  Defina o escopo da varredura e dos ataques. As opções

válidas são:

- url: irá apenas verificar e atacar o URL base exato fornecido com a opção -u.
- página: atacará cada URL que corresponda ao caminho do URL base (cada variação da string de consulta).
- pasta: irá verificar e atacar cada URL, começando com o valor de URL base. Este URL base deve ter uma barra final (sem nome de arquivo).
- domínio: irá verificar e atacar cada URL cujo nome de domínio corresponda ao do URL base.
- punk: irá escanear e atacar todos os URLs encontrados em qualquer domínio. Pense duas vezes antes de usar esse escopo.

ESPECIFICAÇÃO DE ATAQUE

- **-m**, **--module** *MODULE_LIST*
  Define a lista de módulos de ataque (nomes de módulos separados por vírgulas) para lançar contra o alvo.
  O comportamento padrão (quando a opção não está definida) é usar os módulos mais comuns.
  Módulos comuns também podem ser especificados usando a palavra-chave "comum".
  Se você deseja usar módulos comuns junto com o módulo XXE, pode passar -m common, xxe.
  A ativação de todos os módulos pode ser feita com a palavra-chave "all" (embora não seja recomendado).
  Para iniciar uma varredura sem lançar nenhum ataque, basta fornecer um valor vazio (-m "").
  Você também pode filtrar por métodos http (apenas obter ou postar). Por exemplo -m "xss: get, exec: post".

- **--list-modules**
  Imprima a lista de módulos Wapiti disponíveis e saia.

- **-l**, **--level** *LEVEL*
  Nas versões anteriores, o Wapiti costumava injetar cargas úteis de ataque em strings de consulta, mesmo se nenhum parâmetro estivesse presente no URL original.
  Embora pudesse ser bem-sucedido em encontrar vulnerabilidades dessa maneira, estava causando muitas solicitações sem sucesso suficiente.
  Este comportamento agora está oculto por trás desta opção e pode ser reativado definindo -l como 2.
  Pode ser útil em CGIs quando os desenvolvedores precisam analisar a string de consulta por conta própria.
  O valor padrão para esta opção é 1.

OPÇÕES DE PROXY E AUTENTICAÇÃO

- **-p**, **--proxy** *PROXY_URL*
  O URL fornecido será usado como um proxy para solicitações

HTTP e HTTPS. Este URL pode ter um dos seguintes esquemas: http, https, socks.

- **--tor**
  Faça Wapiti usar um ouvinte Tor (o mesmo que --proxy socks: //127.0.0.1: 9050 /)

- **-a**, **--auth-cred** *CREDENCIAIS*
  Defina as credenciais a serem usadas para autenticação HTTP no destino.
  O valor fornecido deve estar no formato login% senha (% é usado como separador)

- **--auth-type** *TYPE*
  Defina o mecanismo de autenticação a ser usado. As opções válidas são basic, digest, kerberos e ntlm.
  A autenticação Kerberos e NTLM pode exigir que você instale módulos Python adicionais.

- **-c**, **--cookie** *COOKIE_FILE*
  Carrega cookies de um arquivo de cookie Wapiti JSON. Veja wapiti-getcookie (1) para mais informações.

OPÇÕES DE SESSÃO

Desde Wapiti 3.0.0, URLs verificados, vulnerabilidades descobertas e status de ataques são armazenados em bancos de dados sqlite3 usados como arquivos de sessão Wapiti.
O comportamento padrão quando existe uma sessão de varredura anterior para o URL e escopo base fornecidos é retomar a varredura e o status de ataque.
As opções a seguir permitem que você ignore este comportamento /

- **--skip-crawl**
  Se uma varredura anterior foi realizada, mas não foi concluída, não retome a varredura. O ataque será feito em URLs atualmente conhecidos sem fazer a varredura mais.

- **--resume-crawl**
  Se o rastreamento foi interrompido anteriormente e os ataques iniciados, o comportamento padrão é pular o rastreamento se a sessão for restaurada.
  Use esta opção para continuar o processo de varredura enquanto mantém vulnerabilidades e ataques na sessão.

- **--flush-attacks**
  Esqueça tudo sobre vulnerabilidades descobertas e qual URL foi atacado por qual módulo.
  Apenas as informações de varredura (rastreamento) serão mantidas.

- **--flush-session**
  Esqueça tudo sobre o alvo para o escopo fornecido.

- **--store-session** Especifique um caminho alternativo para armazenar arquivos de sessão (.db e .pkl)

SINTONIA DE VARREDURA E ATAQUES

- **-s**, **--start** *URL*
  Se por algum motivo, o Wapiti não encontrar nenhum (ou o suficiente) URLs no URL base, você ainda pode adicionar URLs para iniciar a varredura.
  Essas URLs terão profundidade 0, assim como a URL base.
  Esta opção pode ser chamada várias vezes.
  Você também pode dar a ele um nome de arquivo e o Wapiti lerá os URLs do arquivo fornecido (deve ser codificado em UTF-8), um URL por linha.

- **-x**, **--exclude** *URL*
  Evita que o URL fornecido seja verificado. O uso comum é excluir o URL de logout para evitar a destruição dos cookies de sessão (se você especificou um arquivo de cookie com --cookie).
  Esta opção pode ser aplicada várias vezes. O URL excluído fornecido como parâmetro pode conter curingas para correspondência de padrão básico.

- **-r**, **--remove** *PARÂMETRO*
  Se o parâmetro fornecido for encontrado no URL verificado, ele será removido automaticamente (os URLs são editados).
  Esta opção pode ser usada várias vezes.

- **--skip** *PARÂMETRO*
  O parâmetro fornecido será mantido em URLs e formulários, mas não será atacado.
  Útil se você já conhece os parâmetros não vulneráveis.

- **-d**, **--depth** *PROFUNDIDADE*
  Quando o Wapiti rastreia um site, ele dá a cada URL encontrado um valor de profundidade.
  O URL base e os URLs iniciais adicionais (-s) recebem uma profundidade de 0.
  Cada link encontrado nesses URLs tem uma profundidade de 1 e assim por diante.
  A profundidade máxima padrão é 40 e é muito grande.
  Este limite garante que a varredura pare em algum momento.
  Para uma varredura rápida, uma profundidade inferior a 5 é recomendada.

- **--max-links-per-page** *MAX*
  Esta é outra opção para poder reduzir o número de URLs descobertos pelo rastreador.
  Apenas os primeiros MAX links de cada página da web serão extraídos.
  Esta opção não é realmente eficaz, pois o mesmo link pode aparecer em diferentes páginas da web.

Deve ser útil em raras condições, por exemplo, quando há muitas páginas da web sem string de consulta.

- **--max-files-per-dir** *MAX*
  Limita o número de URLs a rastrear em cada pasta encontrada no servidor da web.
  Observe que um URL com uma barra no final do caminho não é necessariamente uma pasta com Wapiti e o tratará como está.
  Como a opção anterior, deve ser útil apenas em determinadas situações.

- **--max-scan-time** *MINUTES*
  Interrompe a verificação após MINUTES minutos se ela ainda estiver em execução.
  Deve ser útil para automatizar a varredura de outro processo (teste contínuo).

- **--max-parameters** *MAX*
  URLs e formulários com mais de MAX parâmetros de entrada serão descartados antes de lançar módulos de ataque.

- **-S**, **--scan-force** *FORCE*
  Quanto mais parâmetros de entrada um URL ou formulário tiver, mais solicitações Wapiti enviará.
  A soma das solicitações pode crescer rapidamente e atacar um formulário com 40 ou mais campos de entrada pode levar uma grande quantidade de tempo.
  Wapiti usa uma fórmula matemática para reduzir o número de URLs verificados para um determinado padrão (nomes de mesmas variáveis) quando o número de parâmetros aumenta.
  A fórmula é **maximum_allowed_patterns = 220 / (math.exp(number_of_parameters * factor) ** 2)** onde factor é um controlador de valor interno pelo valor *FORCE que* você fornece como opção.
  As opções disponíveis são: paranóico, sorrateiro, educado, normal, agressivo, insano.
  O valor padrão é normal (147 URLs para 1 parâmetro, 30 para 5, 5 para 10, 1 para 14 ou mais).
  O modo insano apenas remove o cálculo desses limites, cada URL será atacado.
  O modo paranóico atacará 30 URLs com 1 parâmetro, 5 para 2 e apenas 1 para 3 e mais).

- **--endpoint** *URL* Alguns módulos de ataque estão usando um endpoint HTTP para verificar vulnerabilidades.
  Por exemplo, o módulo SSRF injeta a URL do endpoint nos argumentos da página da web para verificar se o script de destino tenta buscar essa URL.
  O endpoint padrão é http://wapiti3.ovh/. Lembre-se de que o destino e seu computador devem ser capazes de ingressar nesse ponto de extremidade para que o módulo funcione.
  Em pentests internos, este ponto de extremidade pode não

estar acessível ao destino, portanto, você pode preferir configurar seu próprio ponto de extremidade.
Esta opção definirá o URL do endpoint interno e externo com o mesmo valor.

- **--internal-endpoint** *URL* Você pode querer especificar um endpoint interno diferente do externo.
  O endpoint interno é usado pelo Wapiti para buscar resultados de ataques.
  Se você estiver atrás de um NAT, pode ser uma URL para um servidor local (por exemplo http://192.168.0.1/)

- **--external-endpoint** *URL* Defina o *URL* do endpoint (aquele que o destino buscará em caso de vulnerabilidade).
  Usar seu próprio endpoint pode reduzir o risco de ser detectado pelo NIDS ou WAF.

OPÇÕES DE HTTP E REDE

- **-t**, **--timemout** *SECONDS*
  Tempo de espera (em segundos) por uma resposta HTTP antes de considerar a falha.

- **-H**, **--header** *HEADER*
  Defina um cabeçalho HTTM personalizado para injetar em cada solicitação enviada pelo Wapiti. Esta opção pode ser usada várias vezes.
  O valor deve ser uma linha de cabeçalho HTTP padrão (parâmetro e valor separados por um sinal:).

- **-A**, **--user-agent** *AGENTE*
  O comportamento padrão do Wapiti é usar o mesmo User-Agent do TorBrowser, tornando-o discreto ao rastrear sites padrão ou .onion.
  Mas você pode ter que alterá-lo para contornar algumas restrições, então esta opção está aqui.

- **--verify-ssl** *VALOR*
  Wapiti não se preocupa com a validação de certificados por padrão. Esse comportamento pode ser alterado passando 1 como um valor para essa opção.

OPÇÕES DE SAÍDA

Wapiti imprime seu status na saída padrão. As duas opções a seguir permitem ajustar a saída.

- **--color**
  Outpout será colorido com base na gravidade das informações (vermelho é crítico, laranja para avisos, verde para informações).

- **-v**, **--verbose** *LEVEL*
  Defina o nível de verbosidade da saída. Os valores possíveis

são silencioso (0), normal (1, comportamento padrão) e detalhado (2).

OPÇÕES DE RELATÓRIO

O Wapiti irá gerar um relatório no final do processo de ataque. Vários formatos de relatórios estão disponíveis.

- **-f**, **--format** *FORMAT*
  Defina o formato do relatório. As opções válidas são json, html, txt, openvas, vulneranet e xml.
  Embora os relatórios HTML tenham sido reescritos para serem mais responsivos, eles ainda são impraticáveis quando há muitas vulnerabilidades encontradas.

- **-o**, **--output** *OUTPUT_PATH*
  Defina o caminho onde o relatório será gerado.

OUTRAS OPÇÕES

- **--version**
  Imprima a versão Wapiti e saia.

- **--no-bugreport**
  Se um módulo de ataque Wapiti travar de uma exceção não detectada, um relatório de bug é gerado e enviado para análise a fim de melhorar a confiabilidade do Wapiti. Observe que apenas o conteúdo do relatório é mantido.
  Você ainda pode impedir que relatórios sejam enviados usando essa opção.

- **-h**, **--help**
  Mostra a descrição detalhada das opções. Mais detalhes estão disponíveis nesta página de manual.

LICENÇA

Wapiti é coberto pela GNU General Public License (GPL), versão 2. Por favor, leia o arquivo COPYING para mais informações.

DIREITO AUTORAL

AUTORES

Nicolas Surribas é o autor principal, mas a lista completa de colaboradores pode ser encontrada no arquivo AUTORES separado.

LOCAL NA REDE INTERNET

http://wapiti.sourceforge.net/

RELATÓRIO DE ERROS

Se você encontrar um bug no Wapiti, informe-o em https://sourceforge.net/p/wapiti/bugs/

VEJA TAMBÉM

O arquivo INSTALL.md que vem com o Wapiti contém todas as informações necessárias para instalar o Wapiti.