Proteção de servidores web

Até mesmo servidores web configurados e atualizados com correções de segurança, podem ser comprometidos por conta em vulnerabilidades na aplicações web. O mod_security é uma extensão do Apache que atua como um firewall de aplicação para proteger o servidor web de ataques.

por Sebastian Wolfgarten e Kemel Zaidan

Problemas de segurança na web não são mais típicos de resultados de configurações mal feitas ou em decorrência do servidor desatualizado. Tomcat, Apache, e até mesmo o IIS, tornaram-se extremamente maduros através dos últimos anos – tanto que mesmo que eles não tenham nenhuma vulnerabilidade notável, exceções podem sempre vir a tona e contrariar a regra. Além disso, os crackers voltaram suas atenções para aplicações web e scripts que são executados nos servidores. Requisições de usuários cada vez mais rebuscadas estão tornando as aplicações web ainda mais complexas: Ajax, interação com bases de dados externas, interfaces back-end e serviços de diretório são parte do conjunto de uma aplicação moderna. Vetores de ataques crescem para acompanhar este desenvolvimento (**quadro 1**).

Firewalls para a Web

Em contraste aos filtros de pacotes legados, os *WAF* (*Web Applications Firewall*, ou Firewall de Aplicações Web) não inspecionam os dados na camada de rede ou transporte, mas sim na camada nível 7 do protocolo HTTP (camada 7 do modelo OSI) [1]. Para isto acontecer, os firewalls analisam os pacotes de entrada e saída, requisições de clientes e respostas de servidores para distinguirem entre requisições legítimas e maliciosas com base em regras.

Se necessário, podem até mesmo lançar medidas de "contra-ataque". Ao serem configurados para isso, podem até mesmo inspecionar conexões HTTPS criptografadas.

Acessórios em Massa

Em firewalls da maior parte das redes – estamos exagerando um pouco aqui – ou você permitirá alguma ou nenhuma conexão HTTP. WAF tem como alvo conexões HTTP individuais baseadas em seu conteúdo. O ModSecurity é um WAF de alto desempenho para o Apache e um módulo complexo para o servidor de Web Apache [2]. Originalmente desenvolvido por Ivan Ristic, o software conta hoje com o apoio da empresa Breach Security, que patrocina sua distribuição e desenvolvimento .

Duas variantes do softwares estão disponíveis: a variante open-source licenciada sob a GPLv2 e a versão comercial, com suporte profissional, softwares embarcados (*appliances*) pré configurados, e console de gerenciamento. O ModSecurity pode ser executado no Linux, Solaris, FreeBSD, OpenBSD, NetBSD, AIX e Windows, com as próximas versões somente disponíveis para o Apache 2.x. Este artigo discute a versão 2.5.10; o sucessor, 2.6.11 é meramente uma correção de falhas.

O escopo funcional do software é enorme, mas compreensivelmente documentado [3]. Ele registra requisições HTTP e dá ao administrador acesso irrestrito a elementos individuais de uma requisição, como o conteúdo de uma requisição POST. Ele também identifica ataques em tempo real baseado em modelos de segurança positivos ou negativos e detecta anomalias baseado em padrões fornecidos de vulnerabilidades conhecidas.

Quadro 1: Ataques em servidores web

Comparado com aplicações locais, aplicações web são mais vulneráveis porque elas envolvem diversos componentes diferentes – do navegador de Internet, ao servidor web e a estrutura de retaguarda. As vulnerabilidades podem ocorrer em qualquer lugar, mas o servidor está sempre no centro deste ambiente.

Caso a aplicação web não valide suficientemente as entradas do usuário e ao invés disso forneça as informações a um banco de dados sendo executado em segundo plano, os invasores poderão usar ataques de *SQL injection* para injetar seus próprios comandos na cadeia de comandos. Assim o invasor será capaz de ler, modificar, deletar dados e assim exercer uma maior influência na aplicação.

Caso a aplicação também permita aos invasores armazenar arquivos no servidor e executá-los através da web, o invasor poderá configurar um Shell web. Isso acontece porque o servidor executará os arquivos do invasor, e este poderá executar comandos do sistema operacional no servidor web e finalmente escalonar seus privilégios para ter acesso a um shell interativo. Assim, a arquitetura de um Apache cuidadoso e configurado, não dará acesso *root* ao invasor, o que é frequentemente desnecessário para acessar dados sensíveis. E quanto mais serviços estiverem instalados no servidor, mais provável que o invasor encontre um que seja vulnerável.

Poderosas regras descobrem se os números de cartões de crédito estão entre os dados enviados ou utiliza GeoIP para bloquear o acesso de determinadas regiões. O ModSecurity verifica não somente requisições de entrada mas também requisições de saída. O software pode implementar ambientes *chroot*. Como um proxy reverso, ele protege aplicações em outros servidores web, como o Tomcat ou o IIS.

A Breach Security também fornece uma coleção de regras que garantem a segurança básica do servidor web. Uma documentação compreensível, muitos exemplos e listas de discussão fornecem suporte ao usuário. Isso faz do ModSecurity uma boa escolha para proteger servidores web e suas aplicações contra vulnerabilidades. Mas antes mesmo de pensar em alterar as configurações complexas, você primeiro precisará instalar módulos de terceiros .

Pacotes para qualquer distribuição

Se optar por não compilar o pacote para o Apache 2, poderá utilizar um pacote pré-compilado para Debian, RHEL, CentOS, Fedora, FreeBSD, Gentoo e Windows. A instalação manual requer o módulo `mod_unique_id`, que não é automaticamente fornecido por algumas distribuições. Você poderá usar a diretiva:

```
LoadModule security2_module
modules/mod_security2.so
```

para integrar o módulo no arquivo de configuração do Apache `httpd.conf` ou `apache.conf`, caso sua distribuição não faça isto. Após ser reiniciado, o servidor web lista o módulo em seu `error_log`.

Regras de filtragem

O ModSecurity tem uma série de opções configuráveis, mas para entender como elas funcionam, tudo que você precisa é da configuração básica. A opção `SecRuleEngine` ativa o mecanismo de filtragem de módulos e permite processar regras de filtros. As configurações que podem ser usadas aqui são `On`, `Off` e `DetectionOnly`, que diz ao módulo para monitorar, mas para não tomar alguma ação em relação a conexão cliente-servidor, mesmo que regras individuais estejam configuradas para permitir isto. Esta configuração é útil para testar módulos e suas próprias regras. Para auxílio no início ou para depuração, ative a opção `SecDebugLog` para definir um log de diagnóstico (exemplo: `SecDebugLog /var/log/httpd/modsec_debug.log`). Adicionalmente poderá definido o parâmetro `SecDebugLogLevel` para especificar o detalhamento em uma escala de 0 a 9, que envia comentários sobre sua própria atividade e a maneira com que ele processa regras definidas pelo

Tabela 1: Fases de processamento do ModSecurity

Número	Fase	Designador	Atividades
1	Fase de pré-visualização	`REQUEST_HEADERS`	Filtragem o mais cedo possível de requisições antes do controle de acesso, autenticação, autorização e detecção MIME que são realizados pelo Apache.
2	Requisição do cliente	`REQUEST_BODY`	Acesso completo ao conteúdo de requisições de cliente (caso normal).
3	Requisição POST	`RESPONSE_HEADERS`	Opção inicial para filtragem de respostas do servidor.
4	Resposta do servidor	`RESPONSE_BODY`	Acesso completo ao conteúdo de reposta do servidor a uma requisição de entrada do cliente.
5	Logging	`LOGGING`	Acesso a todas as informações relevantes antes de que ela seja gravada nos arquivos de log do Apache.

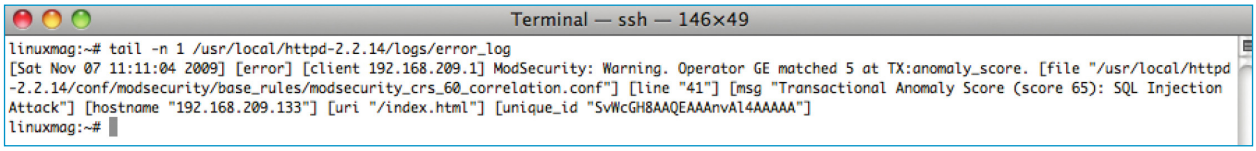

Figura 1: Assim que o ModSecurity for ativado, ele registrará atividades suspeitas no nível de detalhe especificado no SecauditLogParts – neste caso, um ataque SQL Injection.

usuário. Os níveis 4 ou 5 são úteis para fazer um ajuste fino ou diagnóstico. Em ambientes de produção, defina este parâmetro como 0 (zero).

O parâmetro `SecDefaultAction` define o comportamento padrão do ModSecurity para requisições que sejam condizentes com uma regra de filtro sem uma ação correspondente. A opção também espera que você especifique uma fase de processamento para a ação padrão, como listado na **tabela 1**. O ModSecurity aplica regras de filtragem em cinco diferentes fases de processamento e resposta a requisições de clientes [4]. Na vida real, somente a fase 2, no qual o cliente requisita o conteúdo, é filtrado (ou seja, dados de entrada), e a fase 4, que manipula a resposta do servidor (dados de saída) são importantes. Na prática, você poderá usar a opção de definir uma ou múltiplas ações que o software fará quando a conferência ocorrer [5]. Para registrar requisições de entrada do cliente na auditoria da fase 2 e responder a tentativa de requisição com a mensagem de erro HTTP 403 (Forbidden) você deverá fazer:

```
SecDefaultAction phase:2,log,audit
log,deny,status:403
```

Uma função padrão de negativação massiva como o padrão `deny` é altamente restritivo, mas ele somente oferecerá uma proteção máxima se você definir filtros e ações (**quadro 2**). O software oferece algumas alternativas pré-definidas, incluindo parâmetros de conversão de requisições, execução de scripts externos (por exemplo, executar um antivírus) ou redirecionar requisições maliciosas. O último é útil quando você estiver investigando ataques ou desejar encaminhá-los a um *honeypot* (armadilha destinada a atrair um invasor). A configuração básica (**listagem 1**) também registra o conteúdo de requisições de entrada e respostas retornadas. Ele envia informações para o log de auditoria como mencionado anteriormente. A primeira diretiva, `SecAuditEngine`, ativa este comportamento. A opção para a segunda diretiva define se o software armazena as entradas do log de auditoria em um arquivo único (`Serial`) ou grava um arquivo para cada transação (`Concurrent`). O Concurrent será necessário caso você tenha a intenção de implementar o produto ModSecurity Console add-on (console adicional do ModSecurity). A Breach Security oferece um software para gerenciar múltiplas instâncias, desde que este não precise monitorar mais do que três servidores.

A terceira instrução, define a localização do log de auditoria relativa ao caminho de instalação do Apache. Finalmente a instrução `SecAuditLogParts` define a informação que o ModSecurity registra no log de auditoria (**tabela 2**). Neste caso, será o cabeçalho e conteúdo da requisição, junto com a reação do ModSecurity. Os resultados são mostrados na **figura 1**.

Após esta preparação, você poderá adicionar a diretiva mais importante em sua configuração; `SecRule`. Esta regra define a filtragem e opcionalmente a ação que o módulo fará se descobrir que a regra é pertinente. Se não definir uma ação, a ferramenta executará o comando padrão definido na diretiva em `SecDefaultAction`. As regras sempre seguem este padrão:

```
SecRule Variable Operator [Action]
```

O número de variáveis é enorme e elas cobrem cada elemento da requisição do cliente (tanto para POST quanto para GET) assim como os detalhes mais importantes do ambiente servidor [6]. Além disso, você pode utilizar ex-

Listagem 1: Configuração básica do ModSecurity

```
01 # saída do Dig no arquivo ➥
   dnssec_root.key:
01 SecRuleEngine On
02 SecAuditEngine On
03 SecAuditLogType Serial
04 SecAuditLog logs/audit.log
05 SecAuditLogParts ABCFHZ
06 SecDebugLog logs/debug.log
07 SecDebugLogLevel 5
08 SecRule REQUEST_URI “/etc/➥
   passwd”
09 SecDefaultAction phase:2,➥
   log,auditlog,deny,status:403
```

Quadro 2: Ataques internos

Uma vulnerabilidade na aplicação web pode expor o servidor Apache não importando o quanto ele esteja bem configurado e atualizado. Isto é particularmente perigoso se o servidor estiver em uma ambiente de hospedagem onde muitos consumidores compartilham os recursos de um único servidor físico. Os mecanismos de proteção, como virtualização, separarão instancias individuais; no entanto, a segurança de todo o sistema depende do elo mais fraco. Com um *sniffer* (rastreador de pacotes), por exemplo, os invasores podem simplesmente capturar conversas que não estiverem criptografadas. E isto dará ao invasor um vetor de entrada.

Linguagens de script e camadas como PHP ou Ruby on Rails ajudam desenvolvedores web alcançarem resultados mais rapidamente, mas eles geralmente concentram-se em perigos que podem ocorrer quando não é dada a devida atenção a segurança. Em ambientes mais complexos, como Java, Tomcat e JBoss, não são necessariamente a resposta, já que ocultam muitos aspectos do desenvolvedor.

Tabela 2: Argumentos do SecAuditLogParts

Abreviação	Descrição
A	Cabeçalho da linha (obrigatório).
B	Cabeçalho da requisição.
C	Conteúdo requisitado; somente disponível se o conteúdo existir e o ModSecurity estiver configurado para armazená-lo
D	Reservado
E	Conteúdo de resposta temporária; somente disponível se o Modsecurity estiver configurado para isso.
F	Cabeçalho de reposta final após possível manipulação pelo ModSecurity; O Apache gravará os cabeçalhos Date e Server.
G	Reservado
H	Trailer de auditoria equivalente ao C, exceto que quando a requisição contém alguma forma de dado; neste caso, o software construirá uma requisição adequada que exclui o conteúdo do arquivo para simplificar as conferências.
J	Reservado
K	Lista todas as regras que conferem linha por linha na ordem da aplicação.
Z	Fim da linha (obrigatório).

pressões regulares. Por exemplo, para investigar uma requisição HTTP com o intuito de investigar se o cliente esta requisitanto a *string* (cadeia de caracteres) `/etc/passwd` com um método GET, você usaria esta regra:

```
Secrule REQUEST_URI "/etc/passwd"
```

Se a requisição for condizente com a regras, o ModSecurity executará a ação padrão `deny`. A filtragem por tipo de navegador poderá ser feita desta forma:

```
SecRule REQUEST_HEADERS: User-
Agent "nikto"
```

Este exemplo diz ao servidor web para recusar tentativas de acesso do scanner de segurança Nikto [7]. Se desejar que o Modsecurity execute uma ação específica para uma regra, você poderá substituir a ação padrão:

```
Secrule REQUEST_HEADERS: User-
Agent "nikto"
"phase:2,pass,msg:'Scan do Nikto
detectado'"
```

Esta regra diz ao módulo para gravar uma mensagem "scan do Nikto detectado" no arquivo de log quando detectar que o navegador da requisição do cliente da fase 2 conferir. A regra então substitui a ação padrão `drop`, que é definida por `SecDefaultAction` com a ação `pass`. Isto permite ao cliente requisições de passagem. A **listagem 1** mostra uma visão geral da configuração básica para fins de testes do ModSecurity discutidos até aqui.

Na prática

Após iniciar o Apache, uma requisição como `http://www.exemplo.com/index.html?file=/etc/passwd` conferirá com a regra modelo da linha 8. Então a ação definida na linha 9 bloqueará a requisição. O cliente verá um erro HTTP 403 Forbidden (negado). Ao mesmo tempo, as linhas 3 a 7 dirão ao ModSecurity para registrar a transação no log de auditoria e enviar uma visão detalhada da maneira como o cliente foi processado até o log de depuração. Isto significa que o arquivo `error_log` do Apache, terá uma nota para demonstrar que a requisição do cliente foi efetivamente bloqueada, como mostrado na **listagem 2**. Uma vez que o ModSecurity esteja funcionando corretamente, você poderá iniciar a adição de regras e modificá-las para as aplicações web que deseja proteger.

Listagem 2: Requisição bloqueada

```
01 SecAuditLogType Serial Wed Nov 04 05:39:19
02    [error] client ModSecurity: Access
03    denied with code 403 (phase 2). Pattern match
04    "/etc/passwd" at REQUEST_URI. [file "/usr/local/
05    httpd-2.2.14/conf/httpd.conf"] [line "420"] [hostname
06    "www.example.com"] [uri "/index.html"] [unique_id
07    "SvFZ138AAQEAAAc4AgQAAAAA"]
```

A arte de detectar ataques

Para fornecer proteção efetiva contra diversos tipos de ataques, os administradores de sistema precisam configurar um conjunto de regras robustas para o ModSecurity. Para formular regras que os protejam contra SQL injections, scripts cross-site, ou ataque de inclusão de arquivos locais e remotos, por exemplo, você precisará de conhecimento de como ataques em servidores web funcionam.

É claro que nem todo administrador tem esse conhecimento e tempo para reinventar a roda. Para contornar isso, o projeto *Open Web Application* (OWASP) oferece um conjunto de regras para o ModSecurity [8] que se apoia na detecção de anomalias para proteger servidores web contra alguns ataques padrões, como requisições inválidas de clientes, SQL injections, scripts cross-site, e ataques remotos ou locais de injeção de comandos ou email. Isso fornece uma proteção robusta básica, que poderá ser então modificada para se adaptar ao seu ambiente de aplicações, caso necessário.

Listagem 3: Acesso ao GeoIP

```
01 LoadModule geoip_module modules/mod_geoip.so
02 LoadModule security2_module modules/mod_security2.so
03 GeoIPEnable On
04 GeoIPDBFile /usr/tmp/GeoLiteCity.dat
05 SecRuleEngine On
06 SecGeoLookupDb /usr/tmp/GeoLiteCity.dat
07 SecRule REMOTE_ADDR "@geoLookup"
08 "chain,drop,msg:'Connection attempt from .CN!'"
09 SecRule GEO:COUNTRY_CODE "@streq CN: "t:none"
```

Para instalar estas regras básicas, baixe o pacote e descompacte-o no diretório de configuração do Apache. Então mova as regras, incluindo o subdiretório `base_rules` para o diretório do `ModSecurity`. Você poderá ver os arquivos de configuração `modsecurity_crs_10_global_config.conf` e `modsecurity_crs_10_config.conf` e modificá-los para atender suas necessidades. As regras são bem documentadas. Adicionalmente você deverá ativar os logs de auditoria e depuração para ver exatamente o que o módulo está fazendo. Para fazer isso, você precisará incluir as regras padrão no seu arquivo `httpd.conf` como segue:

```
Include conf/modsecurity/*.conf
Include conf/modsecurity/base_
rules/*.conf
```

Após reiniciar, o seu servidor Apache terá uma proteção sólida que responderá com o erro HTTP 403 para quaisquer requisições de clientes classificadas como ataques.

Figura 2: O site do secretário-geral da ONU Ban Ki-moon foi afetado por uma vulnerabilidade de validação de parâmetro de variável. Uma regra ModSecurity virtual patching protegeu o website temporariamente até que o administrador corrigisse o problema.

Os desenvolvedores devem primeiro testar as regras padrão exaustivamente em um ambiente de testes antes de colocá-las em um ambiente de produção. Caso contrário corre-se o risco de bloquear acessos de usuários legítimos. Também é necessário lembrar que o ModSecurity adicionará uma carga extra de aproximadamente 5% no processamento do seu servidor web.

Evitando ataques nas Nações Unidas

Em algumas situações, você não pode corrigir uma vulnerabilidade em aplicações web imediatamente. Imagine uma loja online descobrindo uma falha de segurança uma semana antes do Natal e serão necessários diversos dias para corrigir o problema, significando que a loja deverá ficar offline durante aquele tempo: o dono deverá tomar uma decisão: viver com o risco e manter a loja, incluindo a vulnerabilidade online e assim tendo o benefício da lucratividade das compras de Natal, ou proteger a empresa e seus consumidores desligando o website e corrigindo a vulnerabilidade.

O ModSecurity fornece uma técnica para contornar o problema na forma de um patch virtual, permitindo definir uma ou mais regras que evitem que a vulnerabilidade seja explorada, porém sem removê-la.

A documentação do ModSecurity refere-se ao caso que data do ano de 2007, quando tentaram invadir o site das Nações Unidas [9]. A sub-página de comunicação com o secretário General Ban Ki-moon [10] teve o parâmetro `statID` que expôs uma vulnerabilidade SQL Injection (**figura 2**). Se descobrir uma vulnerabilidade deste tipo, você poderá temporariamente definir uma regra para o Modsecurity que evitará que os hackers explorem a vulnerabilidade mesmo que ela continue a existir. No entanto, esta não é uma boa solução a longo prazo, pois ela não evita um desastre até que os desenvolvedores web removam a vulnerabilidade do código fonte da página. A seguinte solução deverá funcionar para corrigir a falha:

```
<Location /apps/news/infocus/
sgspeeches/statements_full.asp>
SecRule &ARGS "!@eq 1"
SecRule ARGS_NAMES "!^statid$"
SecRule ARGS:statID "!^\d{1,3}$"
</Location>
```

Três linhas adicionadas em um container de localização do Apache asseguram que as requisições válidas de usuários para o arquivo `statements_full.asp` somente serão permitidas se tiverem um argumento (primeira regra) chamado `statid` (segunda regra) com números de 1 a 3 dígitos (terceira regra) como terceiro parâmetro. Quaisquer requisições que não seguirem este padrão, serão limpas pela ação padrão, como definido em `SecDefaultAction`. Isto efetivamente evitará que um invasor explore a vulnerabilidade de SQL injection.

Sem informações internas

O Modsecurity também filtra dados de saída, especificamente requisições de respostas do servidor para informações internas. A linguagem de programação PHP mostra mensagens de erro desta forma:

```
Fatal Error: Connecting to MySQL
server 'dbserv.example.com' failed.
```

No entanto, você poderá desativar mensagens de erro do PHP nas respostas. O Google lista uma série de sites onde mensagens de erros de

PHP revelam muitos detalhes das aplicações. Estas informações são muito úteis a invasores, pois podem ajudá-los a entender o funcionamento interno e a estrutura de uma aplicação web e direcionar melhor o ataque. Para dizer ao ModSecurity para bloquear mensagens de erro do PHP e evitar que estas sejam exibidas aos usuários, você poderá definir uma regra como esta:

```
SecRule RESPONSE_BODY "Fatal error:"
```

O cabeçalho `RESPONSE_BODY` refere-se ao conteúdo de respostas a um cliente, e apesar de não ser particularmente elegante, ele indica o que você potencialmente tem em mãos. Com uma expressão regular cuidadosamente trabalhada, você pode, por exemplo, utilizar a mesma técnica para evitar que números de cartões de crédito sejam revelados por uma aplicação web comprometida após um ataque de SQL injection feito com sucesso.

Localização geográfica

Outro cenário avançado para o ModSecurity envolve cooperar com um provedor GeoIP, o Maxmind [11]. O GeoIP localiza os usuários geograficamente, a partir do seu endereço IP, o que significa que poderá restringir o acesso de um website a uma região específica, por exemplo, a Pensilvânia – se você administrar um site com conteúdo em Holandês da Pensilvânia que ninguém mais entenda – e ou bloquear totalmente um determinado país. Para fazer isto, você deverá instalar o módulo `mod_geoip2` no Apache2, junto com o software GeoIP e o banco de dados `GeoLitecity.dat` [12].

Imagine que o engenheiro mecânico de uma empresa da Alemanha esteja preocupado com a possibilidade de espionagem industrial vinda do oriente; neste caso, ele deverá usar a configuração da **listagem 3** para evitar o acesso a partir da China – isso se os usuários da China não falsificarem a sua origem. As últimas duas linhas formam uma cadeia de regras de filtragem. A linha 6 localiza a região geográfica para o endereço IP de requisição, então a linha 7 efetua a requisição e envia a mensagem para um arquivo de log caso ela venha da China. Isto pode não ser politicamente correto, mas é tecnicamente efetivo.

Conclusão

O ModSecurity possui um grande escopo de características, e pode levar algum tempo para entendê-lo completamente. Mas se você enfrentar o desafio de desvendar as profundezas deste módulo, ele te retornará dividendos através de métodos compreensivos que darão proteção adicional contra ataques em aplicações web, graças ao esquema de regras pré-definidas que torna o inicio fácil. E também ao desenvolvedor por trás do projeto que fornece produtos comerciais e serviços como treinamentos. Os administradores armados com o ModSecurity, podem sentar-se bem no alto de suas suas celas, até mesmo se invasores estiverem tentando derrubá-los do cavalo. ■

Sobre os autores

Sebastian Wolfgarten trabalha como especialista de segurança da informação com o Banco Central da Europa como um consultor, gerente e auditor de projetos internos com o objetivo de aumentar a segurança da infraestrutura de TI. Antes disto, ele passou diversos anos trabalhando para o Ernst & Young AG na Alemanha e como um assessor de Segurança da Informação na Irlanda. Ele também trabalhou como perito em segurança da informação na T-Mobile Alemanha.

Kemel Zaidan é escritor com formação em dramaturgia pela USP. Membro ativo da comunidade Ubuntu e de software livre brasileira, graduando em Análise de Sistemas pela FATEC, Faculdade de Tecnologia de São Paulo, já palestrou em diversos eventos pelo país e é editor da Linux New Media.

Gostou do artigo?

Queremos ouvir sua opinião. Fale conosco em cartas@linuxmagazine.com.br

Este artigo no nosso site: http://lnm.com.br/article/4885

Mais informações

[1] Melhores Práticas da OWASP: Uso de firewalls para aplicações web: http://www.owasp.org/index.php/

[2] ModSecurity: http://modsecurity.org/

[3] Manual de referencia do modSecurity: http://modsecurity.org/documentation/modsecurity-apache/2.5.10/html-multipage

[4] Fases de processamento do ModSecurity: http://modsecurity.org/documentation/modsecurity-apache/2.5.10/html-multipage/processing-phases.html

[5] Ações do ModSecurity: http://modsecurity.org/documentation/modsecurity-apache/1.9.3/html-multipage/05-actions.html

[6] Variáveis do ModSecurity: http://modsecurity.org/documentation/modsecurity-apache/2.1.0/html-multipage/05-variables.html

[7] Nikto: http://cirt.net/nokto2

[8] Projeto da OWASP de regras padrões do ModSecurity: http://www.owasp.org/index.php/

[9] Blog do ModSecurity, "Correções virtuais durante reposta a incidente: Deface das Nações Unidas": http://blog.modsecurity.org/2007/08/27/

[10] Conferencias do Secretário Geral das Nações Unidas: http://www.un.org/apps/news/infocus/sgspeeches/

[11] Serviço de prevenção à fraudes GeoIP Maxmind: http://www.maxmind.com/

[12] Bloqueando tráfego específico de países com o Apache ModSecurity e o GeoIP por Suvabrata Mukherjee: http://linuxhelp123.wordpress.com/2008/12/11/apache