ModSecurity as Universal Cross-platform Web Protection Tool

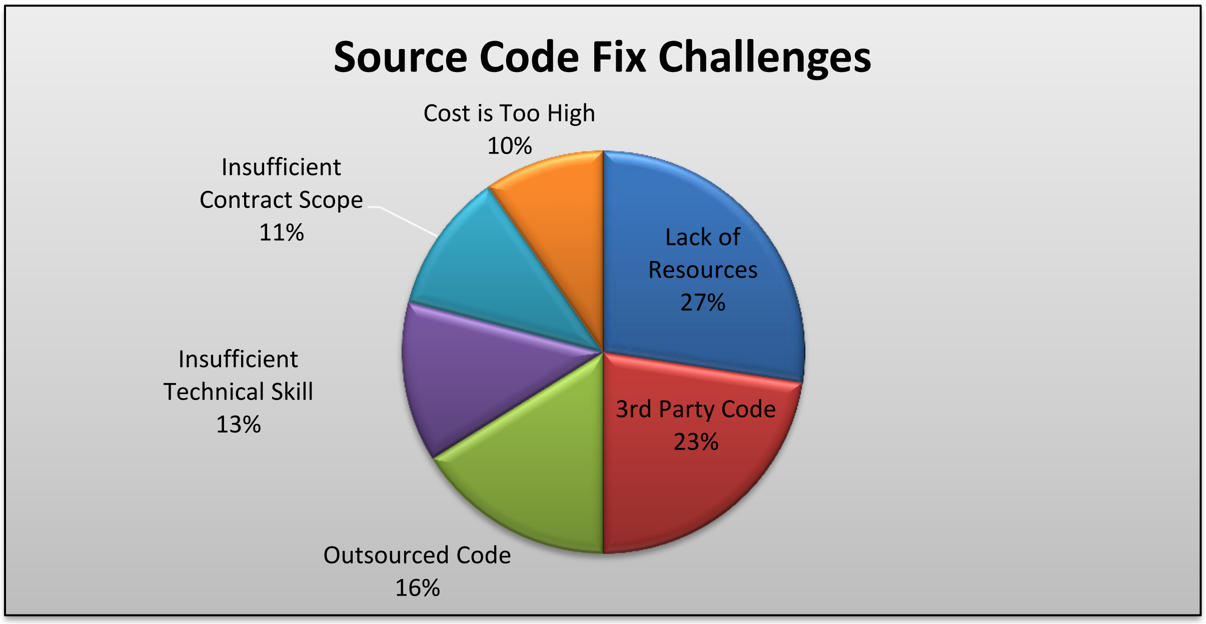
Ryan Barnett
Greg Wroblewski

Abstract

For many years ModSecurity was a number one free open source web application firewall for the Apache web server. At Black Hat USA 2012 we have announced that right now ModSecurity is also available for IIS and nginx servers, making it a first free cross-platform WAF for all three major web server platforms. In this paper we explain how ModSecurity can be plugged in on IIS and nginx and we show how it can be used in early detection of attacks and mitigation of vulnerabilities affecting web infrastructure. We also explain how virtual patching works and how Microsoft Security Response Center will use it to mitigate attacks and vulnerabilities in Microsoft products.

Protecting Web Applications is Challenging

Both web applications and web server platforms that run them, are a big source of security vulnerabilities. While most of security vulnerabilities can be fixed, developers and organizations face significant problems when following the typical process of fix-verify-test-deploy approach:



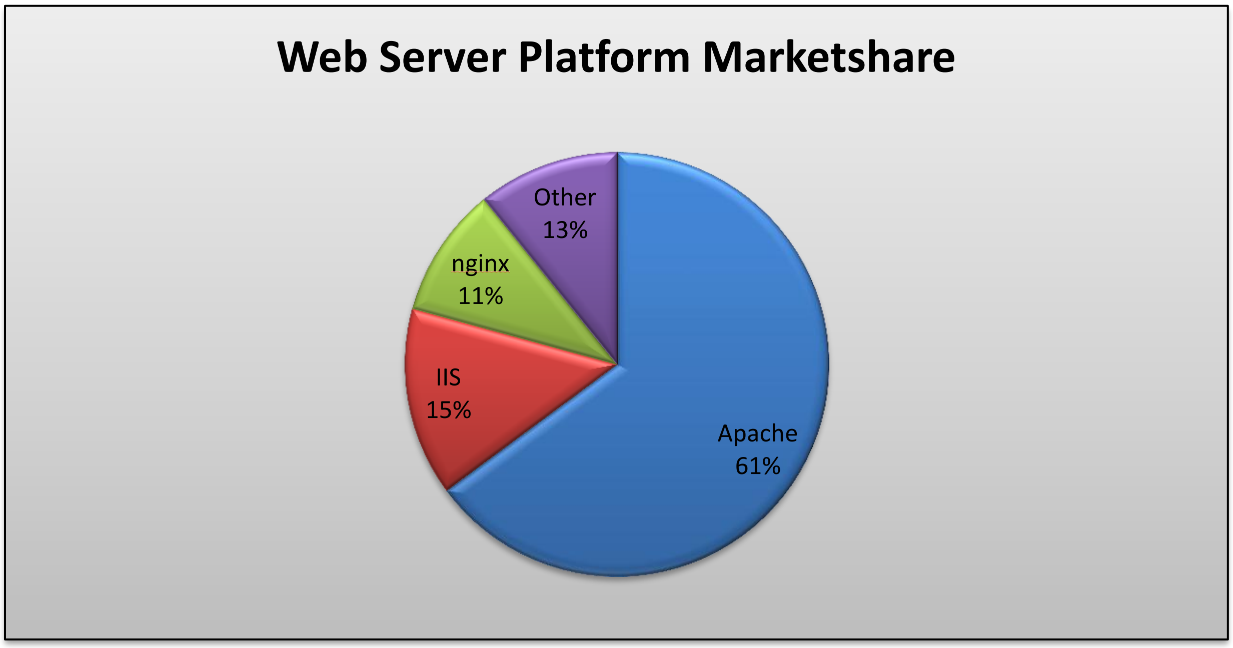
Source: OWASP Web Application Virtual Patching Survey

An alternative solution to full scale fixing is usage of security layers separating trusted environment from untrusted input and user interactions. In the world of web applications such solutions are usually called web application firewalls (WAF). Among the many WAF products available, the ModSecurity module became the most popular choice in the space of open source projects.

Why ModSecurity?

As it was described in [9]: "*While there are other web application firewall applications, ModSecurity is uniquely qualified as the premier option. This is mainly attributed to two factors. First, ModSecurity is an open source, free web application firewall. The fact that there is no cost associated with its use is primarily why it is the most widely installed WAF with more than 10,000 installations woldwide. Second, it boasts a robust rules language and has a number of unique capabilities (outlined below) which allows it to mitigate complex vulnerabilities.*" The advantages of using ModSecurity have been described in numerous publications (see [9] for more examples).

From the beginning of the project until 2012, ModSecurity was only available as an extension module for Apache web server. While the share of Apache was always highest among the most popular web server platforms, in recent years it became clear that if one would like to protect majority of web applications infrastructure, one would have to support two other popular platforms: Microsoft IIS and nginx.

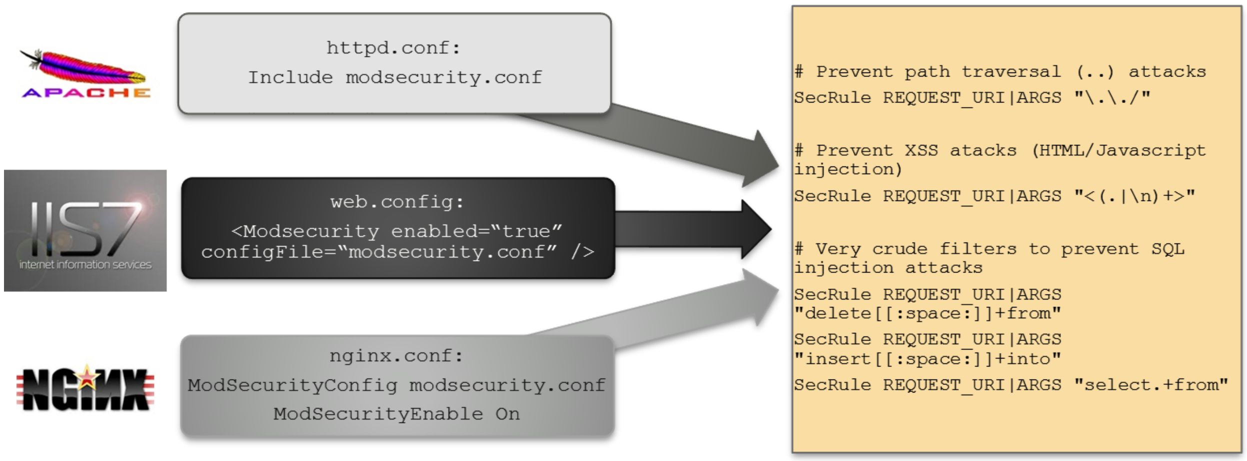


Source: Netcraft: July 2012 Web Server Survey

ModSecurity 2.7.0 – First Multi-platform Release

The discussions about bringing ModSecurity to other web server platforms have started a few years ago, and eventually a community effort made it possible to make ModSecurity version 2.7.0 a first multi-platform release.

There are many advantages of having a uniform security solution covering three most popular web servers. The most obvious one is common format of the definitions specifying web application security policies and protection rules for the entire organization or web data center:



```
httpd.conf:
Include modsecurity.conf
```

```
web.config:
<Modsecurity enabled="true"
configFile="modsecurity.conf" />
```

```
nginx.conf:
ModSecurityConfig modsecurity.conf
ModSecurityEnable On
```

```
# Prevent path traversal (..) attacks
SecRule REQUEST_URI|ARGS "\.\./"

# Prevent XSS atacks (HTML/Javascript
injection)
SecRule REQUEST_URI|ARGS "<(.|\n)+>"

# Very crude filters to prevent SQL
injection attacks
SecRule REQUEST_URI|ARGS
"delete[[:space:]]+from"
SecRule REQUEST_URI|ARGS
"insert[[:space:]]+into"
SecRule REQUEST_URI|ARGS "select.+from"
```

Thanks to a robust design of the porting layer architecture, the major changes between platform-specific versions of the module are limited to operating system or server-specific behavior. For an example, Apache server administrator on a Linux box would typically install the module with the distribution of the OS or compile it using sources from:

http://sourceforge.net/projects/mod-security/files/modsecurity-apache/2.7.0-rc2/

while Windows Server 2008 administrator would use single-file standard MSI installer to extend IIS server on his box with ModSecurity module, downloading it from:

http://sourceforge.net/projects/mod-security/files/modsecurity-iis/2.7.0-rc2/ModSecurityIIS_2.7.0-rc2.msi

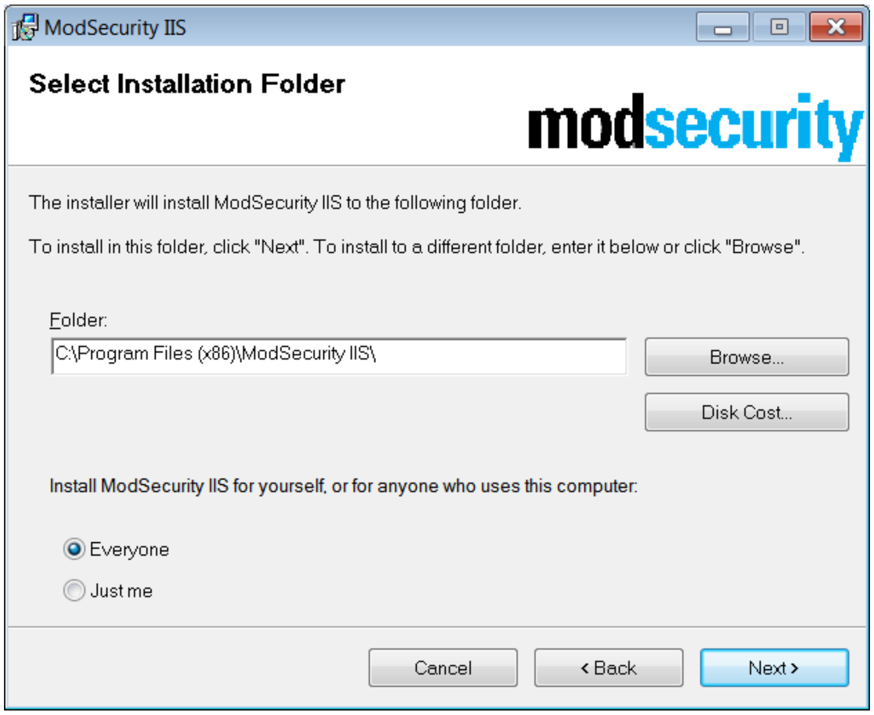
Availability of the module for IIS platform will let Microsoft Security Response Center start publishing ModSecurity rules for vulnerabilities in Microsoft products. It can be also expected, that IIS/ASP/ASP.NET/nginx specific ModSecurity rule sets will be created by community effort, just like it happened for other web application security areas.

ModSecurity for IIS

Although the source code of ModSecurity's IIS components is fully published and the binary building process is described (see mod_security/iis/winbuild/howto.txt), it is highly not recommended to build the module for non-research or non-development purpose.

A standard MSI installer of ModSecurity for IIS is available from SourceForge files repository of ModSecurity project and in the future designated maintainers will be keeping it updated with latest patches and minor versions of the module.

The installation process on IIS is very simple and straightforward. Once installed, the module can be uninstalled. The module binary and schema files are installed in Windows directories, while the `mlogc` tool is installed in Program Files folder.



The installer will install ModSecurity IIS to the following folder.

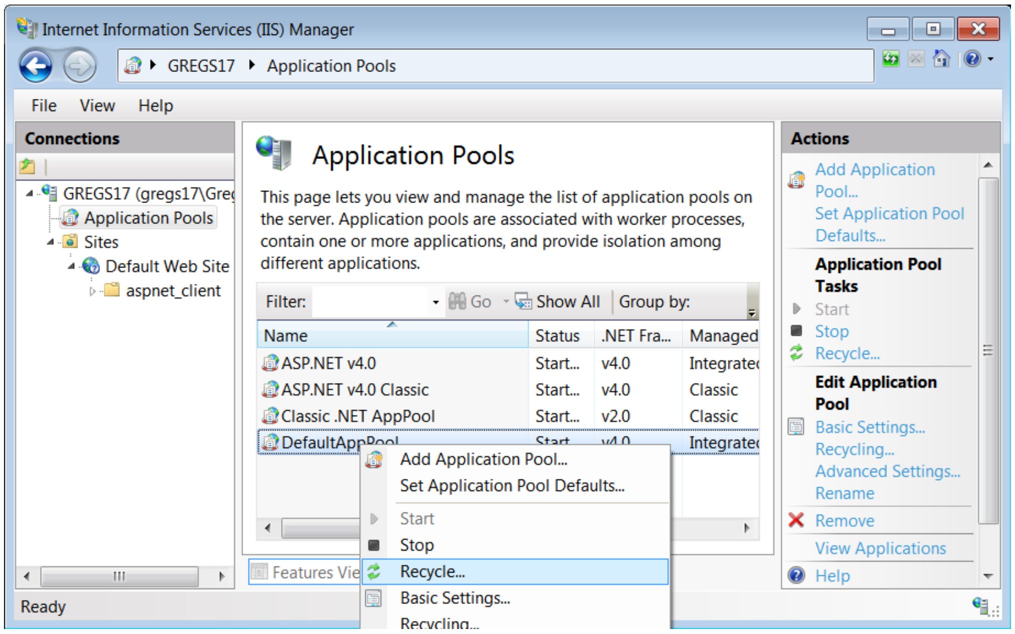
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Program Files (x86)\ModSecurity IIS\

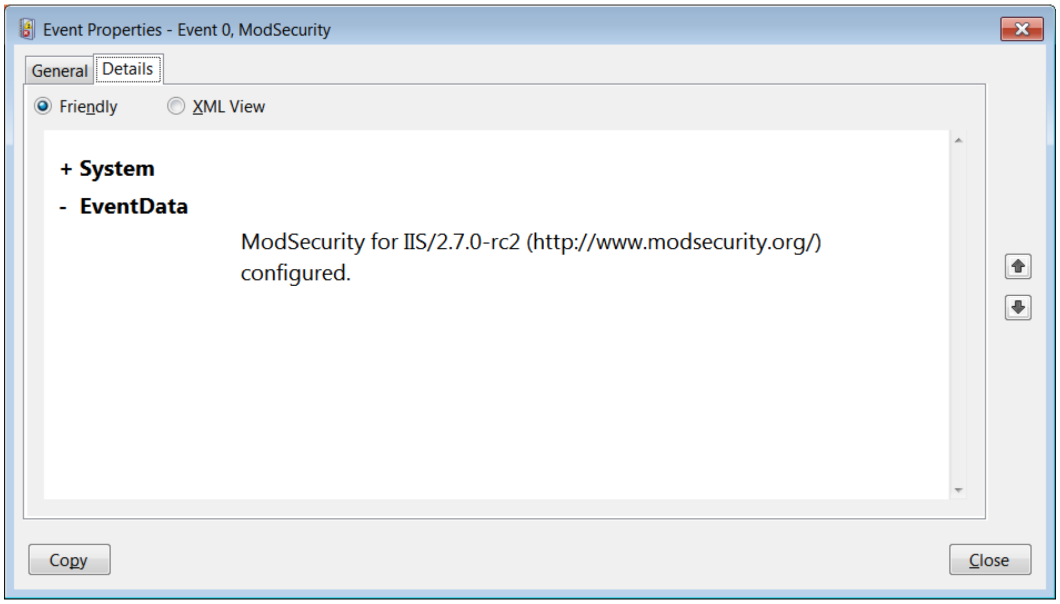
Install ModSecurity IIS for yourself, or for anyone who uses this computer:

[✓ Everyone] Just me

The IIS installer does not interfere with currently running web applications. This implies that the installation process must be followed by an application pool restart or recycling in order to load the new module into the application pool process. For the RC2 version of the module the restart/recycle step is also highly recommended each time a ModSecurity configuration file has been changed:


Internet Information Services (IIS) Manager
GREGS17 ▸ Application Pools
File View Help
Connections
GREGS17 (gregs17\Gre
Application Pools
Sites
Default Web Site
aspnet_client
Application Pools
This page lets you view and manage the list of application pools on the server. Application pools are associated with worker processes, contain one or more applications, and provide isolation among different applications.
Filter: Go Show All Group by:
Name Status .NET Fra... Managed
ASP.NET v4.0 Start... v4.0 Integrated
ASP.NET v4.0 Classic Start... v4.0 Classic
Classic .NET AppPool Start... v2.0 Classic
DefaultAppPool Start v4.0 Integrated
Add Application Pool...
Set Application Pool Defaults...
Start
Stop
Recycle...
Basic Settings...
Recycling...
Features View
Ready
Actions
Add Application Pool...
Set Application Pool Defaults...
Application Pool Tasks
Start
Stop
Recycle...
Edit Application Pool
Basic Settings...
Recycling...
Advanced Settings...
Rename
Remove
View Applications
Help

After successful load of the module into the application pool process a series of informational events is recorded in the application event log. These are the same header lines that are sent to the log file of the Apache web server:



ModSecurity for IIS/2.7.0-rc2 (http://www.modsecurity.org/) configured.

Runtime messages and notifications generated during operational phase, both coming from the user-defined rules and system specific events or errors, are sent to the same application event log repository.

To apply a ModSecurity configuration file to a web application or a path, one has to use IIS configuration schema extension, like in the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <ModSecurity enabled="true"
configFile="c:\inetpub\wwwroot\test.conf" />
    </system.webServer>
</configuration>
```

The `c:\inetpub\wwwroot\test.conf` config file is a regular ModSecurity configuration containing the same directives as used on the Apache web server.

ModSecurity for nginx

The extensibility model of the nginx server does not include dynamically loaded modules, thus ModSecurity must be compiled with the source code of the main server. Since nginx is available on multiple Unix-based platforms (and also on Windows), for now the recommended way of obtaining ModSecurity for nginx is compilation in the designated environment.

The first step in obtaining nginx server with built-in ModSecurity module is building of standalone library containing full ModSecurity with a set of intermediate API (this layer is a common base for IIS version, nginx version, and server-less command line version of ModSecurity). It is recommended to follow the general steps of preparing build environment for ModSecurity and then follow with three simple commands:

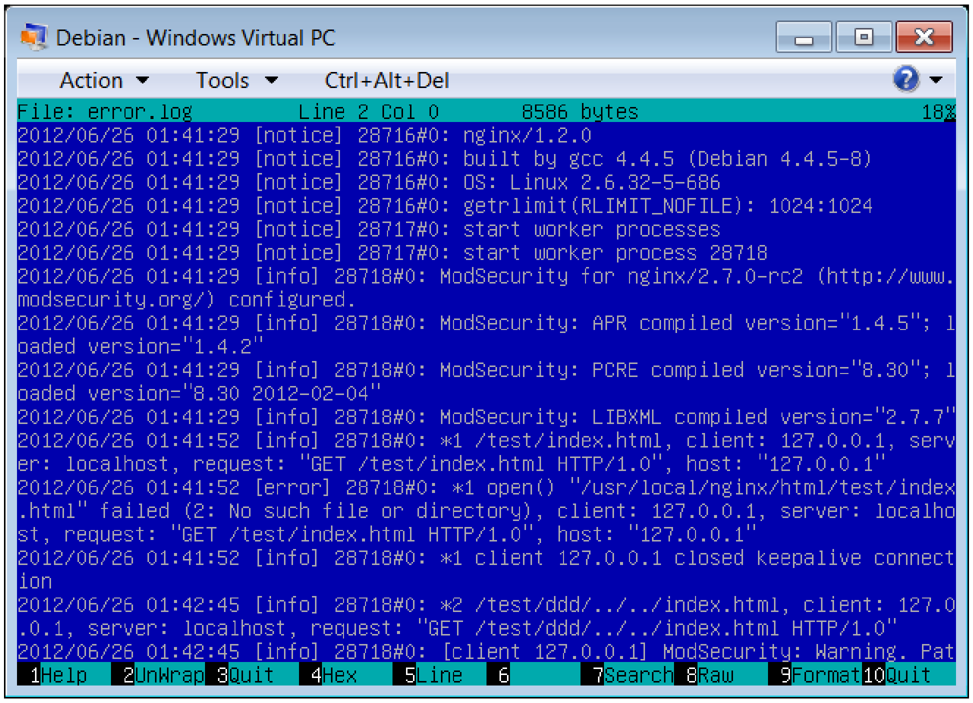
```
~/mod_security# ./configure
~/mod_security# cd standalone
~/mod_security/standalone# make
```

Once the standalone library is built successfully, one can follow with building the nginx server, following the steps from the nginx build tutorial:

```
~/nginx-1.2.0# ./configure --add-module=../mod_security/nginx/modsecur
ity
~/nginx-1.2.0# make
~/nginx-1.2.0# make install
```

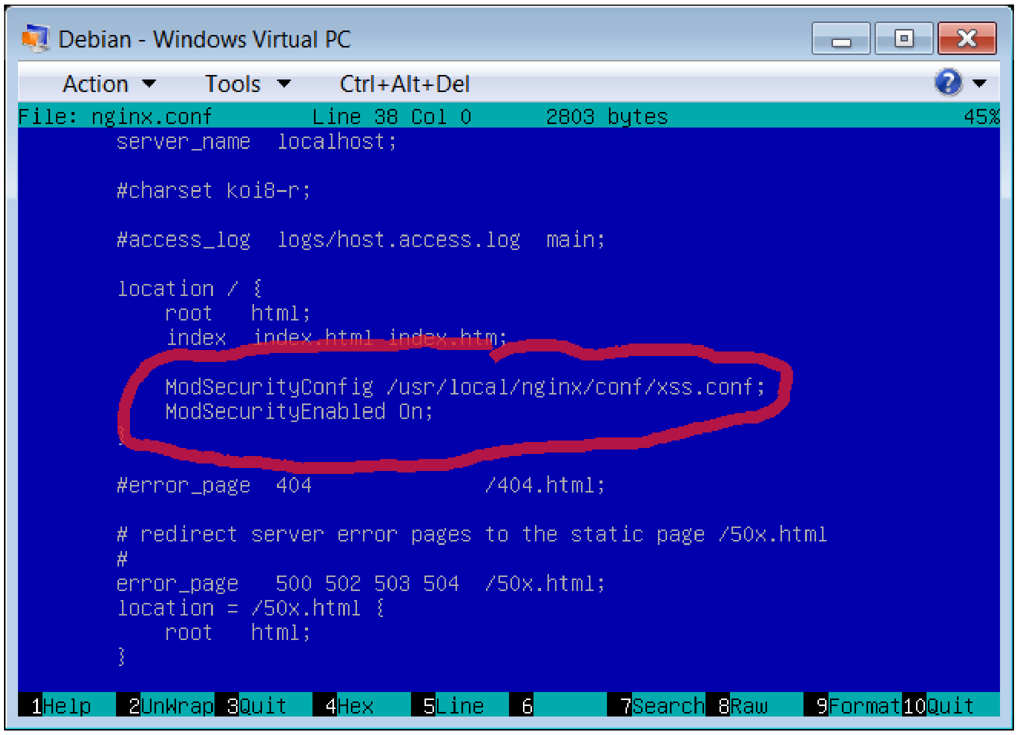
The last command performs server installation on the local machine, which can be either customized or omitted with built binaries packaged or moved to alternative server.

After installation and server start, ModSecurity header lines should appear in nginx's `error.log` file:



```
File: error.log          Line 2 Col 0        8586 bytes                   18%
2012/06/26 01:41:29 [notice] 28716#0: nginx/1.2.0
2012/06/26 01:41:29 [notice] 28716#0: built by gcc 4.4.5 (Debian 4.4.5-8)
2012/06/26 01:41:29 [notice] 28716#0: OS: Linux 2.6.32-5-686
2012/06/26 01:41:29 [notice] 28716#0: getrlimit(RLIMIT_NOFILE): 1024:1024
2012/06/26 01:41:29 [notice] 28717#0: start worker processes
2012/06/26 01:41:29 [notice] 28717#0: start worker process 28718
2012/06/26 01:41:29 [info] 28718#0: ModSecurity for nginx/2.7.0-rc2 (http://www.
modsecurity.org/) configured.
2012/06/26 01:41:29 [info] 28718#0: ModSecurity: APR compiled version="1.4.5"; l
oaded version="1.4.2"
2012/06/26 01:41:29 [info] 28718#0: ModSecurity: PCRE compiled version="8.30"; l
oaded version="8.30 2012-02-04"
2012/06/26 01:41:29 [info] 28718#0: ModSecurity: LIBXML compiled version="2.7.7"
2012/06/26 01:41:52 [info] 28718#0: *1 /test/index.html, client: 127.0.0.1, serv
er: localhost, request: "GET /test/index.html HTTP/1.0", host: "127.0.0.1"
2012/06/26 01:41:52 [error] 28718#0: *1 open() "/usr/local/nginx/html/test/index
.html" failed (2: No such file or directory), client: 127.0.0.1, server: localho
st, request: "GET /test/index.html HTTP/1.0", host: "127.0.0.1"
2012/06/26 01:41:52 [info] 28718#0: *1 client 127.0.0.1 closed keepalive connect
ion
2012/06/26 01:42:45 [info] 28718#0: *2 /test/ddd/../../index.html, client: 127.0
.0.1, server: localhost, request: "GET /test/ddd/../../index.html HTTP/1.0"
2012/06/26 01:42:45 [info] 28718#0: [client 127.0.0.1] ModSecurity: Warning. Pat
 1Help  2UnWrap 3Quit   4Hex    5Line   6       7Search 8Raw    9Format10Quit
```

The ModSecurity configuration file must be linked in `nginx.conf` file using two directives defined by nginx's ModSecurity extension module:



```
File: nginx.conf         Line 38 Col 0       2803 bytes                   45%
        server_name  localhost;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        location / {
            root   html;
            index  index.html index.htm;

            ModSecurityConfig /usr/local/nginx/conf/xss.conf;
            ModSecurityEnabled On;
        }

        #error_page  404              /404.html;

        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }

 1Help  2UnWrap 3Quit   4Hex    5Line   6       7Search 8Raw    9Format10Quit
```

Examples of Virtual Patching

A virtual patching can be defined as a security policy enforcement layer which prevents the exploitation of a known vulnerability. In the scenario with ModSecurity module extending basic functionality of web servers, ModSecurity rules are the enforcement layer specification, which is executed and enforced by the module loaded into the server processes. Using two recent vulnerabilities as an example we show how this layer can be specified in an environment with IIS server running with ModSecurity.

CVE-2011-3414

In December 2011 vulnerability has been discovered in ASP.NET allowing attackers to cause excessive processor load on most ASP.NET web applications:

A denial of service vulnerability exists in the way that ASP.NET Framework handles specially crafted requests, causing a hash collision. An attacker who successfully exploited this vulnerability could send a small number of specially crafted requests to an ASP.NET server, causing performance to degrade significantly enough to cause a denial of service condition.

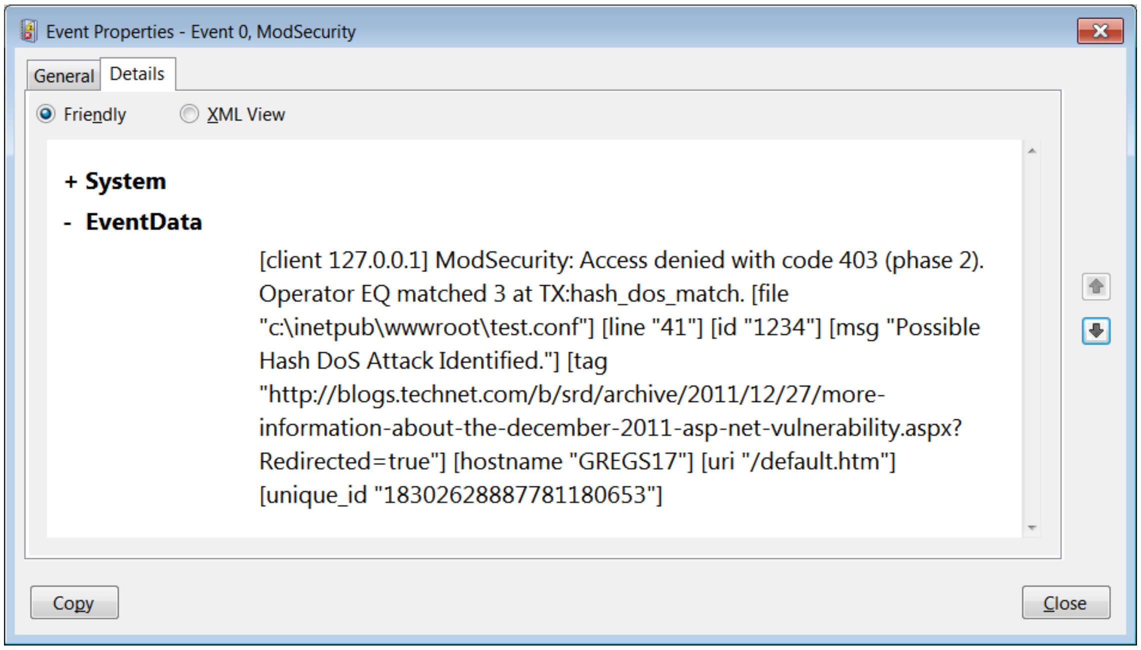
The hash collision issue required attacker to send a large (typically 1MB or 4MB) POST request to the server, with tens of thousands of arguments with specially crafted names. There are at least four ways to mitigate this kind of attack:

- Restrict the request body size.
- Restrict the number of arguments.
- Identify repetitive payloads.
- Check arguments names against PoC data.

The approach checking for the presence of repetitive payload is the most sophisticated one and it can be implemented in ModSecurity using the following chain of rules:

```
SecRule &ARGS "@ge 1000" "chain,id:1234,phase:2,t:none,deny,msg:'Possib
le Hash DoS Attack Identified.',tag:'http://blogs.technet.com/b/srd/ar
chive/2011/12/27/more-information-about-the-december-2011-asp-net-vuln
erability.aspx?Redirected=true'"

  SecRule REQUEST_BODY "^\w*?=(.*?)&\w*?=(.*?)&\w*?=(.*?)&\w*?=(.*?)&"
"chain,capture"

    SecRule TX:1 "@streq %{tx.2}" "chain,setvar:tx.hash_dos_match=+1"

    SecRule TX:2 "@streq %{tx.3}" "chain,setvar:tx.hash_dos_match=+1"

    SecRule TX:3 "@streq %{tx.4}" "chain,setvar:tx.hash_dos_match=+1"

      SecRule TX:HASH_DOS_MATCH "@eq 3"
```

When this rule is loaded into an IIS server configuration and the attack is launched on the protected path, Windows application event log will record an access denied message from ModSecurity:



```
[client 127.0.0.1] ModSecurity: Access denied with code 403 (phase 2).
Operator EQ matched 3 at TX:hash_dos_match. [file
"c:\inetpub\wwwroot\test.conf"] [line "41"] [id "1234"] [msg "Possible
Hash DoS Attack Identified."] [tag
"http://blogs.technet.com/b/srd/archive/2011/12/27/more-
information-about-the-december-2011-asp-net-vulnerability.aspx?
Redirected=true"] [hostname "GREGS17"] [uri "/default.htm"]
[unique_id "18302628887781180653"]
```

At the same time attacker will see HTTP response 403, stopping the attack before it reaches ASP.NET vulnerable component.

CVE-2012-1859

In July 2012 Microsoft patched a classic case of reflected cross-site scripting vulnerability in Microsoft SharePoint 2010. For the attacks to exploit the vulnerability it was enough to trick user into clicking on a malicious URL, like the one below:

```
http://sharepoint/_layouts/scriptresx.ashx?culture=en-us&name=SP.JSGri
d.Res&rev=laygpE0lqaosnkB4iqx6mA%3D%3D&sections=All<script>alert('Hack
ed!!!')</script>z
```

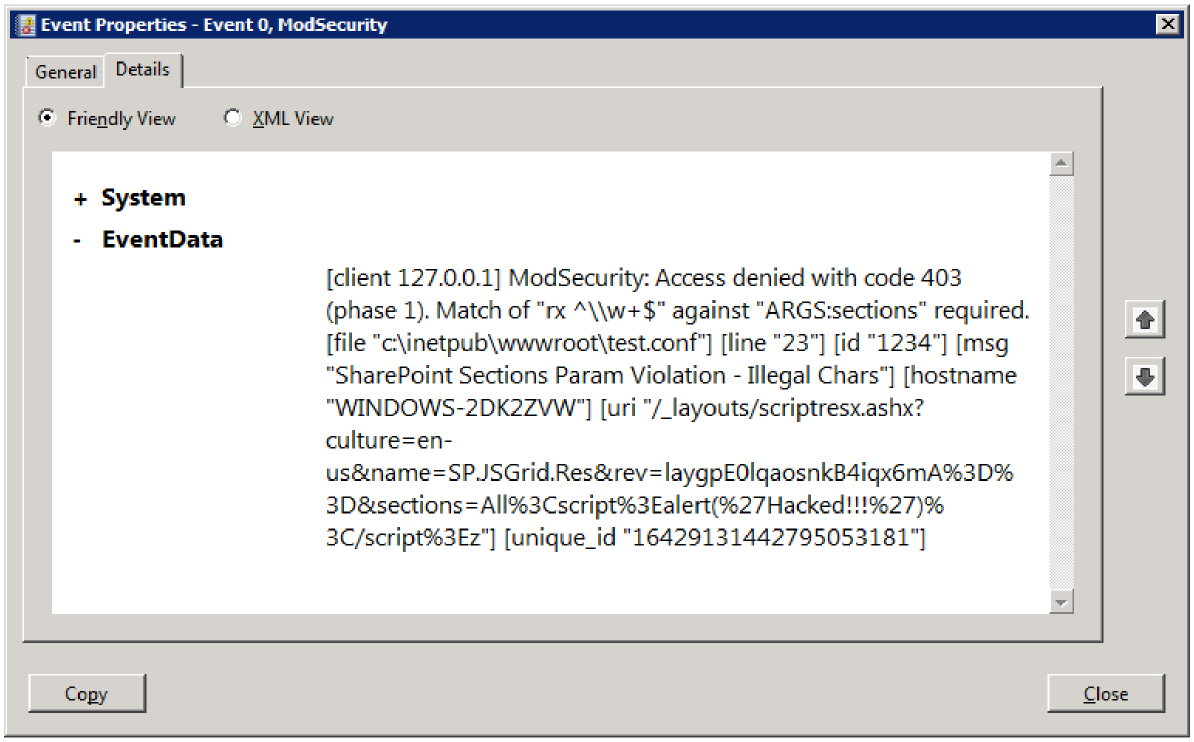
The script injected by attacker could gain access to the entire data set available to the victim through the hacked SharePoint server.

One possible way to block this attack is a whitelist approach: let the URL with `sections` argument that does contain only valid characters pass through, while block all other URLs. Below is a ModSecurity rule implementing this approach for alphanumeric characters:

```
SecRule REQUEST_FILENAME "@contains /_layouts/scriptresx.ashx"
"chain,phase:1,block,msg:'SharePoint Sections Param Violation -
Illegal Chars"
```

```
SecRule ARGS:sections "!@rx ^\w+$"
```

The rule included through ModSecurity config file into the SharePoint `web.config` file, generates the following event when any invalid character (indicating possible attack attempt) is discovered in corresponding SharePoint URL:



```
[client 127.0.0.1] ModSecurity: Access denied with code 403
(phase 1). Match of "rx ^\\w+$" against "ARGS:sections" required.
[file "c:\inetpub\wwwroot\test.conf"] [line "23"] [id "1234"] [msg
"SharePoint Sections Param Violation - Illegal Chars"] [hostname
"WINDOWS-2DK2ZVW"] [uri "/_layouts/scriptresx.ashx?
culture=en-
us&name=SP.JSGrid.Res&rev=laygpE0lqaosnkB4iqx6mA%3D%
3D&sections=All%3Cscript%3Ealert(%27Hacked!!!%27)%
3C/script%3Ez"] [unique_id "16429131442795053181"]
```

Summary

In the presence of widespread vulnerabilities and attacks on web applications the need for robust, universal, portable, flexible and easy to use defensive tools is more than obvious. With ModSecurity module becoming the first open source multi-platform web application firewall, entire IT security community gains a powerful support for both reactive and proactive efforts of protecting the web and Internet users. We also hope that this development will be a seed for further research in the area of web security research.

Project Contributors

The following people have contributed to the multi-platform effort of ModSecurity:

- Microsoft – ModSecurity Port for IIS
  - Greg Wroblewski – Senior Security Developer
  - Suha Can – Security Researcher / Developer
- Trustwave - ModSecurity

    - Ziv Mador – Director of Security Research
    - Ryan Barnett – Security Researcher Lead
    - Breno Pinto – ModSecurity Researcher & Developer
- Open community - Security Port for Nginx
    - Alan Silva - Software Engineer at Alcatel-Lucent

Acknowledgments:

We would like to thank members of the Microsoft's IIS team: Wade Hilmo and Nazim Lala, for their support and help in solving many technical problems.

Resources

[1] ModSecurity home page: http://www.modsecurity.org/

[2] OWASP Core Rule Set for ModSecurity: https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project

[3] MSRC blog: http://blogs.technet.com/b/srd/

[4] Trustwave SpiderLabs blog: http://blog.spiderlabs.com/

[5] Trustwave Commercial Rule Set for ModSecurity: https://www.trustwave.com/modsecurity-rules-support.php

[6] http://blog.modsecurity.org/files/enough_with_default_allow_r1_draft.pdf

[7] http://www.modsecurity.org/documentation/Securing_Web_Services_with_ModSecurity_2.0.pdf

[8] http://www.modsecurity.org/documentation/Ajax_Fingerprinting_and_Filtering_with_ModSecurity_2.0.pdf

[9] https://www.blackhat.com/presentations/bh-dc-09/Barnett/BlackHat-DC-09-Barnett-WAF-Patching-Challenge-Whitepaper.pdf

About Authors

Ryan Barnett

Trustwave's SpiderLabs Research Team
(rbarnett-at-trustwave.com)

Ryan Barnett joined SpiderLabs after a decade in computer security. As Research –Surveillance Team Leader, he leads the SpiderLab team which specializes in application defense. This includes SPAM filtering, network IDS/IPS and web application firewalls. His main area of expertise is in application defense research.

Barnett is renowned in the industry for his unique expertise. He has serves as the Open Web Application Security Project (OWASP) ModSecurity Core Rule Set Project Leader and Project Contributor on the OWASP Top Ten and AppSensor Projects. He is a Web Application Security Consortium (WASC) Board Member and Project Leader for the Web Hacking Incident Database (WHID) and the Distributed Web Honeypot Projects. He is also a Certified Instructor at the SANS Institute.

Barnett is regularly consulted by industry news outlets like Dark Reading, SC Magazine and Information Week. He is the author of Preventing Web Attacks with Apache (Addison-Wesley Professional, 2006.) Key industry events he has addressed include Blackhat, SANS AppSec Summit and the OWASP Global Summit.

Greg Wroblewski

Microsoft
(gwroblew-at-microsoft.com)

Greg Wroblewski, PhD, CISSP, is a senior security researcher at Microsoft's Trustworthy Computing Security group. Over the last 8 years he worked in many areas of security response, presenting part of his work at BlackHat 2007. At Microsoft he focuses on security problems in on-line services, detection of attacks and pentesting. In the past he was responsible for the technical side of patches in over 50 Patch Tuesday bulletins as well as hardening products like Windows and Office 2007. Recently he lead development effort to port ModSecurity module to IIS and nginx servers.