

Signature Development

1 IPS Vulnerability Coverage

1.1 Overview

1.2 Criteria for Vulnerability Coverage

1.3 Responsibilities of Vulnerability Monitoring Team

2 Application Control

2.1 Introduction

2.2 Definitions

2.2.1 Category (--app_cat)

2.2.2 Sub-Category (--app_sub_cat) (FortiOS 4.3 and later)

2.2.3 Technology (--technology) (FortiOS 4.3 and later)

2.2.4 Behavior (--behavior) (FortiOS 4.3 and later)

2.2.5 Vendor (--vendor) (FortiOS 4.3 and later)

2.2.6 Popularity (--pop)

2.2.7 Risk (--risk)

2.2.8 Primary Language (--language)

2.2.9 Weight (--weight)

2.2.10 Depend-on (--depend-on)

2.2.11 Require ssl deep inspection (--require_ssl_di)

2.2.12 Scan-range (--scan-range)

2.2.13 Application default port(--app_port)

3 Industrial package

3.1 SCADA signatures guidelines

4 ISTAME

4.1 Adding or Modifying a Vulnerability

4.2 Adding or Modifying an Application

4.3 Adding or Modifying a Signature

5 Signature Naming Convention

5.1 IPS signature

5.2 AppCtl Signature

6 Vulnerability Severity and Reference

6.1 Severity

6.2 Reference

7 Encyclopedia

7.1 IPS Encyclopedia

7.1.1 Description

7.1.2 Vulnerability

7.1.3 Impact

7.1.4 Recommended Action

7.2 Application Control Encyclopedia

7.2.1 Description

7.2.2 Vulnerability

7.2.3 Impact

7.2.4 Recommended Action

7.2.5 Example

7.3 Encyclopedia Template for Zero-Day Vulnerabilities

7.3.1 Template for Zero-Day Vulnerability Received from Vendor/3rd-Party

7.3.2 Template for Zero-Day Vulnerability Discovered by Ourselves

8 Things need to know about signature

8.1 IPS Engine Logic

8.1.1 Service Trees

8.1.2 Signature Matching

8.1.3 Application Control signatures selection process

8.1.4 Ignore_content

8.1.5 Custom Application Control signatures

8.1.6 Industrial Application Control signatures

8.1.7 Application Whitelisting

8.1.8 Whitelisting Logic

8.1.9 UDP and ICMP Sessions

8.1.10 Quarantining Attackers

8.1.11 Traps

8.2 Public Tag

8.3 file_type

8.4 SSL Inspection

8.4.1 Install FortiGate SSL Proxy certificate

8.4.2 Get Decrypted Content

8.4.3 Reminders

- 8.4.4 SSL session logic
- 8.5 SNI verification
- 8.6 Best Practices
- 8.7 Signature Templates
- 9 Signature Quality Assurance
 - 9.1 Overview
 - 9.2 Review Team
 - 9.3 False Negatives
 - 9.4 False Positives
 - 9.4.1 Case study - MS.WinVerifyTrust.Signature.Validation.Remote.Code.Execution
 - 9.4.2 Case study - MS.IE.MSXML.Object.Handling.Code.Execution
 - 9.5 Process for Signature Quality Control
 - 9.5.1 New Regular Signature Creation
 - 9.5.2 Public Signature Modification
 - 9.5.3 Hidden (tag-set) Signatures
 - 9.5.4 skip-after Signatures
 - 9.5.5 Signature Status and Action Change
 - 9.5.6 Pre-release notification of AppCtrl package
 - 9.5.7 Signature Pullout & Re-enable
- 10. Systems for Quality Assurance
 - 10.1 Pcap Management System(PMS)
 - 10.1.1 Review high alert beta signature
 - 10.1.2 Handle False Positive
 - 10.1.3 Change the default action to drop
 - 10.2 FRS
 - 10.2.1 FDS statistics
 - 10.2.2 Evaluate a signature

1 IPS Vulnerability Coverage

1.1 Overview

With the increasing focus on Cybersecurity, software vulnerabilities discovery is becoming increasingly popular today. With how reliant on technology the world is today, the amount of software vulnerabilities found and disclosed can only increase rapidly. Coupled with old vulnerabilities, this amounts to a huge number of vulnerabilities that security companies must manage. However, the amount is too big, and no security vendor can cover all of them. On the contrary, the IPS leaders today can only cover up to between ten to twenty thousand vulnerabilities. Some of the reasons are as follows:

- Performance degradation as the number of vulnerabilities increases.
- Many vulnerabilities are local exploits, so network based IPS cannot provide protection against those exploits.
- Some vulnerabilities are technically impossible to support.
- Vendors cannot obtain the required details of the published vulnerability, that is, no PoC or specific information about the exploit can be obtained.
- As time moves on, a lot of old vulnerabilities become irrelevant. For example, vulnerabilities affecting MS-DOS are irrelevant mostly due to its usage being rare today.
- Other reasons such as limited resources, time constraints, etc.

In our signature development, it is crucial to select the most important and most appropriate vulnerabilities to best meet the security needs of our customers. The standards established in this document are a balance between the actual requirements of our customers and the existing resources of our IPS service team.

1.2 Criteria for Vulnerability Coverage

We only support vulnerabilities that fall into the following categories.

- Vulnerabilities supported by known testing tools, including:
 - Metasploit
 - Exodus
 - Trend Micro
 - BreakingPoint
- Vulnerabilities supported by our competitors, including:
 - Sourcefire
 - Palo Alto
 - Checkpoint
- Vulnerabilities from Microsoft patches, typically Super Tuesday every month, and Adobe patches.
- Vulnerabilities coverage requested by our customers or SEs, either through email to "vulwatch@fortinet.com" or tickets created on FortiCare.
- Vulnerabilities found by our vulnerability monitoring team and fit our criteria for adding a signature.

1.3 Responsibilities of Vulnerability Monitoring Team

The Vulnerability Monitoring Team consists of members of the signature development team. This team's objective is to monitor various sources of vulnerabilities and select the vulnerabilities we will need to support. These vulnerabilities will then be entered into the **IPS Signature Tracking and Management Environment (ISTAME)** system, with signatures added.

The selection principles are as follows:

- The vulnerabilities should be on popular hardware platforms, popular Operating Systems or popular applications. An exception is when the choice is made based on impact or media attention.
- OT related vulnerabilities
- Information from external sources such as the following websites indicates a vulnerability we should cover:
 - [sec-consult](#)
 - [Talos](#)
 - [Tenable](#)
 - [PoC in Github](#)
 - [zeroscience](#)
 - [packetstorm](#)
 - [seebug \(Chinese site\)](#)
 - [CISA](#)
 - [ExploitDB](#)

2 Application Control

2.1 Introduction

Application Control (AC) can accurately identify and control application traffic going through a network, so that network administrators can create policies to allow or deny the usage of specific network applications. There are more than 1000 applications that are supported in FortiOS 7.6. To make management easier for users, some attributes are added to each application signature.

2.2 Definitions

2.2.1 Category (--app_cat)

In order to effectively manage application signatures, we group them into these categories. Each application signature must belong to one of the following categories.

Categories 4.0	Categories 5.0	Descriptions
IM	N/A	The IM category consists of IM (Instant Messaging) software and online chat applications, which can establish real time text-based communication between two or more people over a network.
P2P	P2P	The p2p category consists of P2P (Peer to Peer) applications and associated P2P protocols, which can establish a P2P network to provide fast data sharing.
VOIP	VOIP	The voip category consists of voice communication software using VoIP (Voice Over Internet Protocol) technologies (e.g. SIP, H.323 etc.), which involve voice communication over a network.
Media	Video/Audio	The media category consists of video/audio streaming applications and associated protocols, which can provide online video/audio services to users.
Proxy	Proxy	The proxy category consists of proxy software, VPN softwares and websites, which allows for indirect network connections to other networks

		and bypassing of firewall policies.
Remote.Access	Remote.Access	The remote-access category consists of remote management software and associated protocols, which can be used to log into and control a remote machine.
Game	Game	The game category consists of signatures detecting all kinds of games
Web	General.Interest	This category consists of websites and browser-based applications.
Network.Service	Network.Service	The network-service category consists of application layer protocols, over TCP or UDP.
Business	Business	This category covers applications which are critical for company operation. Example of apps to be included: HR, CRM, Sales, Marketing, Tracking, e-Procurement, Analytics, ERP, Finance, Logistics
Update	Update	The update category refers to the self-update function of a particular software application or system, which could be automated or scheduled.
Botnet	Botnet	The botnet category consists of signatures detecting various botnets.
Email	Email	The email category consists of signatures detecting various email protocols and applications
Storage.Backup	Storage.Backup	The Storage.Backup category consists of applications and online services, which provide file storage or backup.
Social.Networking	Social.Media	This category consists of online services, which help build social networks or social relations among people.
N/A	N/A	File.Sharing category has been deprecated
N/A	Web.Others	The Web.Others category consists of applications that are web-based but do not fit into any other category. In general, we do not add any signature to this category. Most if not all applications should fit into the other categories

		on the list.
N/A	Industrial	The Industrial category consist of applications that related to industrial control system
N/A	N/A	The special category has been deprecated
N/A	Collaboration	This Collaboration category consist of applications that help people involved in a common task achieve goals
N/A	Business	The Business category consists of enterprise applications, e.g. SAP, Salesforce
N/A	Cloud.IT	This category mostly covers IaaS and PaaS applications that provide generic cloud IT without specific business function or end user visibility. Example of apps to be included: cloud networking, cloud IT management, cloud security, cloud infrastructure
N/A	Mobile	This category consists of applications that detect traffic from mobile devices, For example Apple.iPhone.
N/A	Unknown Applications	This category is only used for logging purposes on the Fortigate. No signatures should be added to this category.
N/A	IoT	This category consists of all IoT applications.
N/A	OT	This category consists of all OT applications.

Note.

1. N/A means this category is not available for this FortiOS version. Analysts need to select other categories available for this FortiOS version best fit this application.
2. Botnet signatures have been moved to the IPS database since FortiOS 5.6.

2.2.2 Sub-Category (--app_sub_cat) (FortiOS 4.3 and later)

The Sub-Category attribute was introduced to further enhance the grouping of application categories. It is only available in FortiOS 4.3 and later versions prior 5.0. The sub-category is optional, do not specify it if it is not applicable.

Under Media, there are three sub-categories:

Sub-Category ID	Sub-Category Name	Description
2	Audio	This sub-category consists of audio streaming applications and associated protocols, which provides online audio streaming services.
3	Video	This sub-category consists of video streaming applications and associated protocols, which provides online video streaming services.
4	Photo	This sub-category consists of applications which provide online photo management or sharing services.

Under web, there are 9 sub-categories:

Sub-Category ID	Sub-Category Name	Description
1	Web.Browsing	This sub-category consists of web-based application access.
5	Social.Network	This sub-category consists of social-networking websites.
6	Business	This sub-category consists of websites which provide services to meet the business interest of people online.
7	Personal	This sub-category consists of websites which provide services to meet the personal interest of people online.
8	Web.Conference	This sub-category consists of web applications which provide online conference services.
9	Search.Engine	This sub-category consists of websites which provide Internet search services
10	Web.Mail	This sub-category consists of websites which provide web-based email services.
14	File.Sharing	This sub-category consists of web applications which provide file sharing services.
15	Toolbar	This sub-category consists of third-party toolbars for browsers.

16	HTTP.File.Download	This sub-category consists of file download through the HTTP protocol
----	--------------------	---

Sub-category is deprecated since 5.0

2.2.3 Technology (--technology) (FortiOS 4.3 and later)

The Technology attribute was introduced to describe the communication technology used by an application. It is only available in FortiOS 4.3 and later versions. Multiple values can be set for one application.

Technologies	Descriptions
Network-Protocol	Applications implement all kinds of Internet protocols, such as DNS, DHCP, etc.
Browser-Based	Applications using web browser-based communications, such as Facebook, Google, etc.
Client-Server	Applications using client - server model, such as Google.Talk, Adobe.Update, etc.
Peer-to-Peer	Applications using p2p-based communications, such as eMule, BitSpirit, etc.

2.2.4 Behavior (--behavior) (FortiOS 4.3 and later)

The Behavior attribute was introduced to describe an application's behavior. It is only available in FortiOS 4.3 and later versions. Multiple values can be set for one application. If it is not specified, the default value is "Other".

#	Behaviors 4.0	Behaviors 5.0	Behaviors 5.4	Descriptions
1	Reasonable	N/A	N/A	The application's behavior is reasonable when appearing in a company network. Many applications in Business, Collaboration, Email, General.Interest, Network.Service, Remote.Access, Social.Media, Storage.Backup, Update, Web.Others should be in this category.

2	Botnet	Botnet	Botnet	The application is usually a malware which is installed without the owner's knowledge. It allows attackers to control the infected device by communicating with the botnet C&C servers.
3	Evasive	N/A	Evasive	The application supports evasion techniques which can be used to bypass general firewall filtering, such as Skype and other p2p applications. Most decoder signatures are evasive because they require a decoder to block, which means they have multiple evasion means. Some Proxy applications that constantly release new updates to bypass firewalls are in this category too like Ultrasurf, Psiphon, Freegate and Hotspot.Shield.
4	Productivity-Loss	N/A	N/A	The application could cause a productivity loss when used in a company network. For now, only games are put in this category. We want to avoid being put in a position where certain IM, Social Media are allowed in certain organizations and not allowed in other organizations.
5	Excessive-Bandwidth	Excessive-Bandwidth	Excessive-Bandwidth	The application consumes excessive bandwidth, such as p2p applications, video and audio calls, uploads and downloads.
6	Tunneling	N/A	Tunneling	The application supports tunneling techniques, such as proxy applications.

7	Reconnaissance	N/A	N/A	The application has scanning functionality and can obtain information from other systems.
8	Encrypted-Tunneling	N/A	N/A	The application supports encrypted tunneling techniques, such as proxy applications like UltraSurf, FreeGate, etc.
9	N/A	N/A	Cloud	<p>The application enables user collaboration on a cloud server and facilitates remote access through cloud servers. It should ideally be browser-based or at least offer browser compatibility. For instance, applications that permit users to upload and download files at any time are classified as exhibiting cloud-based behavior</p> <p>Note.</p> <ol style="list-style-type: none"> 1. The application is part of SaaS, IaaS or PaaS. We do not follow the exact definition of SaaS. That would mean many applications will have this behavior. 2. IM apps with file transfer are not considered to have Cloud behavior. 3. The exception is the Cloud deep-app control signatures - we put them under Cloud. E.g. Facebook, Gmail and Instagram.
0	Other	N/A	N/A	Other behavior than the above.

Many behavior values have been deprecated in 5.0 for easy maintenance.

2.2.5 Vendor (--vendor) (FortiOS 4.3 and later)

The Vendor attribute was introduced to highlight applications produced by the most well-known vendors. It is only available in FortiOS 4.3 and later versions. The following table contains the well-known vendor list. It is based on the number of application signatures in our database. If it is not specified, the default value is "Other".

#	Vendors	Descriptions
1	Other	
2	Microsoft	
3	Google	
4	Facebook	
5	Yahoo	
6	Tencent	
7	AOL	
8	Apple	
9	RealNetworks	
10	Netease	
11	Sina	
12	Zoho	
13	Adobe	
14	Shanda	
15	Sohu	
16	Alibaba	
17	Cisco	
18	IBM	
19	Symantec	
20	MySpace	

21	Baidu	
22	EBay	
23	Sun	
24	Citrix	
25	LogMeIn	
26	Amazon	
27	SAP	

2.2.6 Popularity (--pop)

The Popularity attribute provides a rating of how widely the application is used. If it is not specified, the default value of Popularity is "Low".

#	Popularity 4.0	Popularity 5.0	Descriptions
1	High	5	
2	High	4	
3	Medium	3	
4	Medium	2	
5	Low	1	

5.0 introduced 5 popularity levels.

After the signature is released, ISTATE will adjust the popularity level for AppCtrl signature based on FDN statistics automatically.

2.2.7 Risk (--risk)

The Risk attribute rates the risk level when the application is running in a local network. If the risk level is not specified, the default value of Risk is "Low". The risk level of an application will be basically based on its 5.0 category as shown in following table:

#	Risk 4.0	Risk 5.0	5.0 Categories
1	High	5	Botnet, Proxy

2	High	4	P2P, Remote.Access
3	Medium	3	Email, Storage.Backup, Cloud-IT
4	Medium	2	Game, Collaboration, Social.Media, Video/Audio, Voip, Industrial, General.Interest, Network.Service, Web.Client, Mobile
5	Low	1	Update, Business

Applications in the same category may have different risk level so analyst need to adjust the risk level base on following considerations:

- Known to be used by Malware, For example QQ
- Has Known Vulnerabilities, For example HTTP.BROWSER_IE
- Enterprise based proxy should not have a high risk

Generally, the adjustment will be one step only. For example, QQ belongs to category IM which has a default risk level 3. Analysts should adjust its risk level to 4 considering QQ is known to be used by Malware and has known vulnerabilities.

2.2.8 Primary Language (--language)

Primary language attributes indicate the regions where an application is widely used. Some of the choices are:

- Multiple (applications like google or Facebook which serve different languages)
- N/A (applications like network protocols, Botnet which do not have a GUI)

The full list can be found on the ISTANCE application setting page.

It is key to judge the **primary language**. Following two web sites are recommended when setting primary language for web-based applications and mobile applications:

1. **Alexa** (<http://www.alexa.com/>)

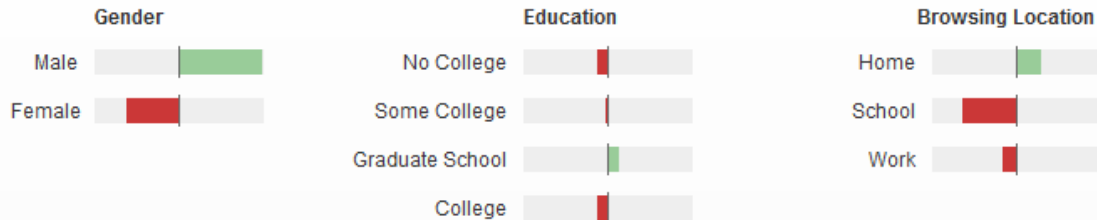
Alexa is a California-based subsidiary company of Amazon.com which provides commercial web traffic data. It can provide basic information about website geographical visits. For example, try it with domain "Speedtest.net" you should see a similar result as shown in following image.

Who visits speedtest.net?



Audience Demographics

How similar is this site's audience to the general internet population?



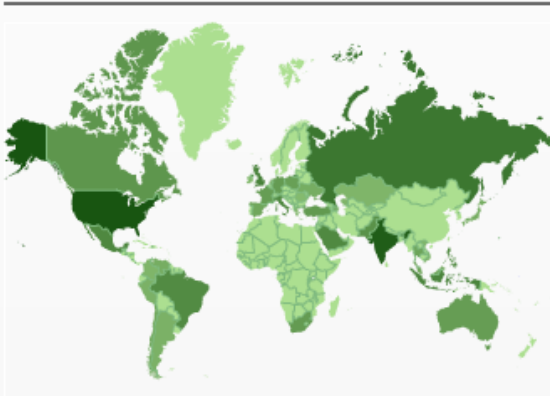
Subscribe to the **Alexa Pro Insight Plan** to view all demographics including age, income, ethnicity and children.

[View More](#)

Audience Geography

Where are this site's visitors located?

Visitors by Country



Country	Percent of Visitors	Rank in Country
United States	19.3%	441
India	14.4%	233
Russia	5.6%	374
United Kingdom	3.8%	315
Indonesia	3.4%	184
Mexico	3.0%	290
Saudi Arabia	2.9%	129
Italy	2.9%	357
Philippines	2.7%	76
Brazil	2.7%	500

[Less](#)

As you can see, the United States, which is the top 1 country, only takes up 19.3% in terms of the percentage of visitors. In language settings, 80% and above are recommended to be considered as primary. As a result, the primary language of signature "Speedtest" should be "Multiple".

There are cases where one appctrl signature, for example "Yandex.Search", covers multiple domains. Yandex is the largest internet search engine based in Russia. Yandex mainly has four domains. They are "www.yandex.ru" (Russian version), "www.yandex.ua" (Ukrainian version), "www.yandex.com" (English version), and "www.yandex.com.tr" (Turkish version). The global traffic ranks for these domains are listed in the following table according to Alexa at the time of writing.




Domain	Global Rank
yandex.ru	20
yandex.ua	364
yandex.com	1586
yandex.com.tr	686

As you can see, the rank of domain yandex.ru is much higher than the other three domains. So, we only consider domain yandex.ru when deciding the primary language of signature “Yandex.Search”. If you try Alexa with domain yandex.ru you will find more than 80% visitors are from Russia. As a result, the primary language of signature “Yandex.Search” should be “Russian”.

2. **App Annie** (<http://www.appannie.com/>)



App Annie is a business intelligence company and analyst firm. It produces a range of business intelligence tools and market reports for the apps and digital goods industry. Login is required to view data mentioned in the following section. We registered an account “applicationcontrollanguage@gmail.com/FortinetTest1” for this purpose.

Try it with “WeChat” in Google Play store. You should see a similar part of “Highest Ranks” page under “Daily Ranks” as shown in the following image.

 WeChat		Overall	Games	Applications	Communication	
	# of countries - rank 1 reached	9	0	9	9	
	# of countries - rank 5 reached	17	0	17	25	
	# of countries - rank 10 reached	21	1	22	31	
	# of countries - rank 100 reached	34	2	36	51	
	# of countries - rank 500 reached	47	2	51	51	
	# of countries - rank 1000 reached	48	2	51	51	
 AppAnnie.com	Country					
	China	1 Jan 10, 2013	-	1 Jan 21, 2012	1 Jan 10, 2013	
	Hong Kong	1 Aug 29, 2013	-	1 Feb 03, 2013	1 Feb 03, 2013	
	Taiwan	1 Nov 15, 2012	-	1 Nov 15, 2012	1 Nov 15, 2012	
	Malaysia	1 Apr 01, 2013	-	1 Apr 01, 2013	1 Apr 01, 2013	
	India	1 May 16, 2013	-	1 May 16, 2013	1 May 16, 2013	
	Philippines	1 May 17, 2013	-	1 May 17, 2013	1 May 16, 2013	
	Italy	1 Jul 18, 2013	-	1 Jul 18, 2013	1 Jul 18, 2013	
	Indonesia	1 Mar 04, 2013	-	1 Mar 03, 2013	1 Mar 04, 2013	
	Mexico	1 May 18, 2013	-	1 May 18, 2013	1 May 18, 2013	
	Spain	2 Jul 19, 2013	-	2 Jul 19, 2013	2 Jul 19, 2013	
	Singapore	3 Jan 10, 2013	9 Feb 14, 2014	3 Jan 10, 2013	2 Jan 10, 2013	
	Argentina	3 Jul 21, 2013	85 Feb 14, 2014	3 Jul 21, 2013	3 Jul 16, 2013	
	South Africa	3 Jul 15, 2013	-	3 Jul 12, 2013	2 Jul 12, 2013	
	Turkey	3 Jul 10, 2013	-	3 Jul 09, 2013	2 Jul 09, 2013	
	Brazil	4 Jul 28, 2013	-	3 Jul 28, 2013	2 Jul 27, 2013	

In “Communication” category, we can observe that WeChat has been ranked top 1 in nine places among free communication apps. Obviously multiple languages are spoken in these nine places. As a result, we set “Multiple” as the primary language for WeChat application.

As another example, try it with “QQ” in Google Play store. The highest daily ranks are as below.

 <p>QQ</p> <p>App Details</p> <p>Intelligence</p> <p>Daily Ranks</p> <p>Rank History</p> <p>NEW Keywords / ASO</p> <p>Featured</p> 	Daily Ranks				
	Ranks		Highest Ranks ?		
		Overall	Games	Applications	Communication >
	# of countries - rank 1 reached	1	0	1	1
	# of countries - rank 5 reached	1	0	1	2
	# of countries - rank 10 reached	1	0	1	4
	# of countries - rank 100 reached	4	0	4	34
	# of countries - rank 500 reached	10	1	13	51
	# of countries - rank 1000 reached	10	1	13	51
	Country				
	China	1 Feb 04, 2012	-	1 Feb 02, 2012	1 Jan 02, 2012
	Hong Kong	25 Jul 29, 2012	-	16 Mar 31, 2012	3 Mar 27, 2012
	Singapore	81 Jul 05, 2013	129 Feb 14, 2014	62 Feb 05, 2013	10 Feb 02, 2014
	Taiwan	88 Jul 29, 2012	-	59 Jul 29, 2012	8 Apr 02, 2012
	New Zealand	347 Nov 30, 2012	-	206 Nov 30, 2012	20 May 21, 2014
	Australia	366 May 07, 2013	-	233 May 08, 2013	20 Dec 11, 2012
	Malaysia	440 Nov 29, 2012	-	235 Nov 30, 2012	21 Oct 02, 2013
	Russia	440 Nov 29, 2012	-	235 Nov 30, 2012	28 Nov 30, 2012
	South Korea	482 Jul 30, 2012	-	351 Jul 30, 2012	25 Jul 29, 2012
	Canada	492 Mar 18, 2014	-	319 Mar 18, 2014	23 Mar 18, 2014
	Netherlands	-	-	342 May 07, 2014	25 Apr 02, 2014

In the “Communication” category, QQ ranked the first place for daily ranks among free communication apps in China. As a result, we set “Chinese” as the primary language for QQ application.

2.2.9 Weight (--weight)

Option ‘weight’ was introduced in IPS engine 3.49. **Every AppCtrl signature needs to have a ‘weight’ value.** With this information, the IPS engine can pick up the best (heaviest) signature for a session when multiple signatures are triggered. Following please find the guidelines of setting weight value:

1. Network protocols: 1-9

Examples:

```
SSH:1
HTTP.BROWSER:1
HTTP.BROWSER_IE: 2
HTTP.Segmented.Download:2
```

2. Main apps: 10-19

Examples:

Facebook:10
Sharepoint :10
Globat :10

3. Sub apps: 20-199

Examples:

Facebook.App: 20
Facebook.App_TexasHoldem :30
Sharepoint_Calendar:20

4. Botnet: 250

Examples:

Conficker.Botnet: 250
Zeus.Botnet :250

Analysts need to set the weight value in the application setting page as shown in the following image:

← → ↻ 172.16.77.29/wsgi/istame/vulnerability_application_setting_edit/15832/

ISTAME Forticare PMS IPS stats Evaluate FortiWiki Mantis TSL Mu URL De

Vulnerabilities, Signatures, Rule Files, Descriptions, PCAPs, Documents, Lists, Icons, Account,

+ [Icons] [Link] [File] [Edit] [Comment] [Share] [Settings] [Calendar] [Trash]

Vulnerability

15832

Facebook

Icons

[Facebook icon] [facebook] [Trash] [Add]

Popular: 5 - High ▼

Risk: 3 - Medium ▼

Language: Multiple ▼

Vendor: Facebook ▼


Vendor name: Facebook

Vendor url: www.facebook.com

Application category: Social.Networking ▼

Application category5: Social.Media ▼

Application sub category: ----- ▼

Weight: 10 

Note.

Weight only differentiate signatures which have the same action. For example following signatures are all set to allow:

HTTP.BROWSER:1

HTTP.BROWSER_IE: 2

Global: weight 10

Engine will pick up the signature Global for sessions to the site <http://www.globat.com/> because it has the heaviest weight value. If you change the action of HTTP.BROWSER to block and still want the engine to pick up Global you need the keyword depend-on.

2.2.10 Depend-on (--depend-on)

The **depend-on** keyword was introduced in IPS engine 3.49 to support whitelisting in AppCtrl. Analysts use this keyword to specify the dependency relationship between applications. This keyword is optional.

The dependency relationship is transitive. For example to whitelist “Globat”, analysts only need to make it depend-on Web.Browsing.

--depend-on 38941;

At the time of writing Web.Browsing(vid 38941) was set to depend-on following signatures:

34038	HTTP.BROWSER_IE
34039	HTTP.BROWSER_Chrome
34040	HTTP.BROWSER_Safari
34041	HTTP.BROWSER_Opera
34050	HTTP.BROWSER_Firefox
38783	Wget
38996	HTTP.BROWSER_UCBrowser
40568	HTTPS.BROWSER
40469	HTTP.BROWSER_Edge
41392	HTTP.BROWSER_Opera.Mini
107347980	Proxy.HTTP
15862	WebDAV

Following signatures were set to depend-on HTTP.BROWSER


22933: HTTP.BROWSER_Opera.Mini
34038: HTTP.BROWSER_IE
34039: HTTP.BROWSER_Chrome
34040: HTTP.BROWSER_Safari
34041: HTTP.BROWSER_Opera
34050: HTTP.BROWSER_Firefox
22933: Opera.Turbo

To whitelist Globat, you can configure the action for Globat as 'pass,' while setting all other signatures to 'block. When you access site <http://www.globat.com/> using IE the first HTTP request will match following signatures:

HTTP.BROWSER_IE
HTTP.BROWSER
Globat

Engine will pick up Globat based on the dependency relationship and pass this packet.

Analysts need to set the depend-on values at the application setting page as shown in following image:

Vulnerability	
38941	
Web.Browsing	
Icons	
+	
Popular:	1 - Low ▼
Risk:	3 - Medium ▼
Language:	Multiple ▼
Vendor:	Other ▼
Vendor name:	<input type="text"/>
Vendor url:	<input type="text"/>
Application category:	Network.Service ▼
Application category5:	Web.Others ▼
Application sub category:	----- ▼
Weight:	1 <input type="text"/>
Depend on:	22933,34038,34039,34040,34 
Behavior list:	

To input more than 1 vid, please use comma as a separator.

2.2.11 Require ssl deep inspection (--require_ssl_di)

As some applications require ssl-deep inspection, and others don't. We introduce the attribute require_ssl_di to add granularity to application control configuration.

In IPS Engine 3.286, signatures that have "--req_ssl_di "Yes"" will be discarded if deep-inspection is not enabled. For applications that could use HTTP or HTTPS, we should put them as "does not require deep-inspection".

We can add the following note in Recommended action:

Note: This application may be accessed through HTTP or TLS sessions. In a TLS session, this signature requires "deep-inspection" in FortiOS 5.0 and above.

2.2.12 Scan-range (--scan-range)

The new application control hierarchical design was released in IPS Engine 3.286. With this new engine build, the --depend-on and --scan-range syntaxes are very important for the IPS Engine to identify a session with a signature. **The scan-range option is mandatory while the depend-on is optional.**

In general, the most basic guidelines which would apply to most signatures are:

- a) HTTP signatures that do not require deep-inspection with flow from_client should be fine with a 1k scan-range.
- b) SSL signatures that only check for the hostname should be using a scan-range value of 1. **The engine will not offload a session until the key exchange is done.**
*******Note:** In Mantis 437696, the IPS Engine will add a new logic to optimize the scanning for SSL sessions. Signatures with --service SSL, --context host and scan-range 1, will make the engine stop scanning the session once the server's name is detected either in the Client Hello packet or the Server Certificate packet.

Hence, any simple service SSL signature that checks only --context host should have a scan-range of 1 for the best performance.

- c) If a signature has only one depend-on value, use **--scan-range xx,all;** instead of **--scan-range xx,yy;**. It makes modifying the depend-on value easier next time. **Scan-range value modification needs to go through the beta process.**
- d) The scan-range value for parent apps is what affects performance the most. The current largest scan-range for TCP parent app is 25k while the largest scan-range for UDP parent app is 20k. For parent apps, analysts should avoid using a large scan-range to avoid performance impact.

Important changes about IPS Engine 3.286 and above:

- a) Depend-on values are **very important** now. Wrong depend-on values will result in missed detections.

Example:

Correct

Web.Browsing (38941) → Google.Services (42533) → Gmail (15817)
→ Google.Drive

Vs

Incorrect

Web.Browsing (38941) → Google.Services (42533) → Google.Drive
→ Gmail (15817)

In the past, we put most signatures to depend-on Web.Browsing. The hierarchical design will only try to match the signatures in the nodes directly below the current node. E.g. if we have Gmail (let's say the signature checks flow from_server) that depend-on Web.Browsing and in a session, Google.Services triggers on the first client packet, Gmail will be discarded because it does not

depend-on Google.Services. This situation is likely to happen in applications that use Google, Amazon, Microsoft or other cloud server providers like Ultrasurf, Psiphon, some VPN/Proxy software, etc. It could also happen with children signatures of a parent.

- b) We have a special signature “Web.Browsing” which is a slint signature, i.e it will never trigger. One of the limitations of this engine is that the scan-range is updated based on the node **directly below** the currently triggered signature.

Example:

HTTP.Browser → HTTP.Browser_Chrome → Web.Browsing → Google.Services

For almost all of our HTTP signatures, it can be detected within a 2k scan-range value. Therefore, it will bypass the HTTP.Browser, HTTP.Browser_Chrome and Web.Browsing signatures and trigger Google.Services or other similar signatures and take on the scan-range values of the signatures below their tree.

There could be problems with flow from_server signatures. If the signature checks the body of a server reply, there could be cases where the client packet + server header is larger than 2k which would cause a missed detection. In such a session, usually the matched signature is HTTP.Browser or HTTP.Browser_Chrome. Since the scan-range value will then be HTTP.Browser_Chrome or Web.Browsing due to them being the signature directly below the current signature, it will not use the scan-range value of the signature that is supposed to check the flow from_server packet. In cases like this, try to create signatures that check the header of a server reply instead of the body. If that is not possible, we could use skip-after to get around the problem.

HTTP.Browser(1k)→HTTP.Browser_Chrome(1k)→Web.Browsing(2k)→Server.Body(10k)

In the example above, since Web.Browsing does not trigger, the engine will not use the scan-range of the server’s body. It will use at most 2k from Web.Browsing. The number of signatures in such a case should be small.

- c) Signatures that have children that could trigger without first triggering the parent signature would not be able to tell the engine to update the scan-range value.

Example:

Fortiguard (2k) → Fortiguard_Web.Filtering (200k)

If a user visits <http://www.fortiguard.com/webfiltering> as the first URL instead of going to <http://www.fortiguard.com> first and then only clicking on the link to /webfiltering, the engine will not be able to update the scan-range to 200k. **This is because the engine only takes the largest scan-range value of its immediate children.** There are no more children after Fortiguard_Web.Filtering. Thus, the scan-range remains unchanged - default value.

To get around this limitation, we use the “--skip-after” keyword to increase the scan-range.

2.2.13 Application default port(--app_port)

Numerous networking protocols and applications typically operate on specific ports. For

instance, TELNET uses port 21, HTTP uses port 80, and Facebook uses ports 80 and 443, among others. Some customers have a requirement to enforce strict port restrictions within their environment, meaning that, for instance, TELNET running on a non-standard port like 2121 should be blocked. To facilitate this enforcement, we have introduced the 'app_port' metadata parameter.

Format

--app_port "<protocol>/port_number[,<protocol>/port_number]*"

The protocol can be either TCP or UDP

The port_number is an integer within [1,65535]. It can also be a range which is indicated by a '-' character between two integers.

Examples of usage:

--app_port "UDP/1111"; //udp port 1111

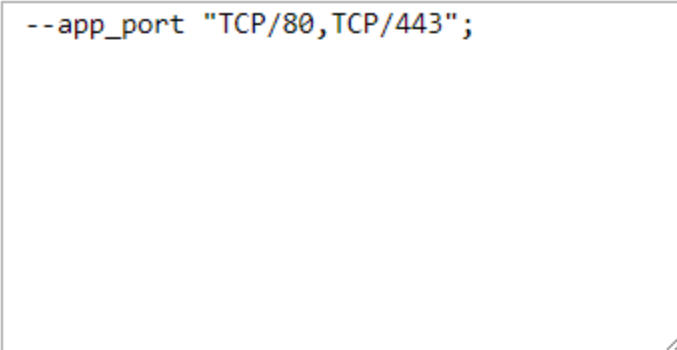
--app_port "TCP/8801-8810"; //tcp port range 8801-8810

--app_port "TCP/80,TCP/443"; //tcp port 80 and 443

The IPS engine stores unique values for this metadata. To minimize memory usage, it is essential to adhere to the following rules:

1. TCP port lists should precede UDP port lists whenever both are present.
2. Whenever feasible, consolidate port ranges. For example, use 'TCP/8801-8802' instead of listing 'TCP/8801' and 'TCP/8802' separately
3. Arrange smaller port numbers before larger ones.
4. Check the list of unique app_ports on [App Ctrl Quality.xlsx](#) and reuse one of the existing app_port if possible. If it is needed to create a list of app_ports that is not on the sheet, please double check with the review team.

To add the app_port metadata, analysts need to manually input it in the body section of the application setting page as below.

Body: 

Note. You don't need to provide the app_port metadata if the application or protocol doesn't have a default port or it doesn't run on TCP or UDP protocols at all.

3 Industrial/OT package

With the release of 5.6, SCADA or industrial signatures have been moved to a separate package from IPS and application control signatures. The package, termed industrial or OT package in later versions of FortiOS, consists of IPS signatures in the group SCADA and appctrl signatures in category Industrial. As this package is sold separately from the IPS package, we have a few guidelines when handling SCADA related signatures.

3.1 SCADA signatures guidelines

When determining if a signature should be added to ISTANCE as a SCADA signature, analysts should determine the following.

- a) Is the product used exclusively in the industrial environment? If it can only be found in regular households or offices, then it should be placed in regular IPS or application control packages.

For all SCADA IPS reports, the App List under OS and APP on ISTANCE should be set to "SCADA".

4 ISTANCE

ISTAME is the **IPS Signature Tracking and Management Environment**. It is used to manage the creation and updating of IPS signatures by the analysts. The signatures managed by ISTANCE are extracted by Signature QA to build the release packages. ISTANCE is also used for Application Control (AC) development.

The system is located at [ISTAME2](#). The development team is consistently updating this system to satisfy the requirements of analyst and QA. As ISTANCE will be used regularly by all analysts, there are certain good practices that an analyst should try to follow.

4.1 Adding or Modifying a Vulnerability

Before adding a vulnerability, analysts need to check ISTANCE to see if a report has already been created for the vulnerability. One way is to search with reference, for example CVE number, or other related information such as keywords. Avoid adding duplicate reports if possible as this leads to redundancy and sometimes confusion when there are a few reports created for a single vulnerability. There are exceptional cases though.

To increase default drop rate for some testing tools, for example, BP/Trend Micro, we may

create a duplicate report for a vulnerability. This is usually because the original report is not appropriate for a drop action. In such a case, the original report containing generic signatures will provide a good coverage for the vulnerability while the duplicate report can provide a high default drop rate with signatures based on PoC for the vulnerability. Do not add more than one duplicate report unless necessary. As a guideline, there should be at most one report for generic signatures and one report for drop signatures for each vulnerability.

Since 1.230, the rate value has become unique per vuln_id. It may be necessary to create a duplicate report if you want to write signatures using different rate values.

When adding new exploit traffic and signature for an existing vulnerability, find the correct report to add them in. That is, the report should correspond with the traffic and signature you are adding in. Do not place exploit/signature in a different vulnerability, for example reports with a different CVE number from your vulnerability, just because it has default drop action or because they are from the same testing tool.

Analyst should avoid using one report to cover multiple vulnerabilities unless:

1. These vulnerabilities are very similar or closely related so that it is not possible or appropriate to cover them with different signatures.
2. This report is for generic signatures like Cross.Site.Scripting

Here are the procedures to follow when adding/modifying a vulnerability:

1. Provide the appropriate "**Vuln Name**" (not the signature name).
 - If the original vulnerability report contains multiple vulnerabilities, for example for a single Microsoft bulletin that contains multiple vulnerabilities, it should be broken into equal amounts of vulnerability reports based on the CVE numbers matching the multiple vulnerabilities.
2. Provide the **CVE, MS numbers** and other references if available.
 - They will be shown in the IPS encyclopedia ([Fortiguard Encyclopedia](#)).
 - For applications, provide a link to the vendor's website.
3. Set the appropriate "**Severity**".
 - It will be shown on the Fortigate WebUI .
 - Refer to chapter 6.1 for information on setting the severity.
4. Choose a proper "**group**" for the vulnerability.
 - Choose '**backdoor**' for Botnet signatures
 - Choose '**SCADA**' for industrial control systems vulnerabilities
 - Choose '**web_client**' for Browser-based vulnerabilities
 - Choose '**apache**' for Apache's vulnerabilities
 - Choose '**web_server**' for web servers' vulnerabilities except Apache

- Choose '**web_app**' for web applications vulnerabilities
 - Choose '**operating_system**' for vulnerabilities targeting any operating system
- Analysts need to choose the group correctly to ensure that it is consistent amongst our signatures. For example, Oracle WebLogic is under web_server on ISTANCE, so any newly added reports for it should be under the web_server group as well. When unsure analysts should search for similar products on ISTANCE and see which group they are associated with.
5. Mark "test source list" accordingly
 6. Set "viewable" state for MAPP or 0day vulnerability
 7. Analyze the vulnerability and **fill in the "Analysis" section**. The following information should be included:
 - What is the cause of the vulnerability? Please give your understanding of the cause.
 - What did you select as the detection condition in the signature? Please explain why, if it is not obvious. This is especially important for a new analyst if the detection condition is not an obvious one as an explanation will help the signature review team in understanding your detection logic.

You may leave "Analysis" section empty when creating the report and fill it out later. ISTANCE does not allow you to release a signature with "Analysis" section not filled.

8. Find and upload as many **exploits (PoC)** as possible and set their attributes. This is very important when we review the report in the future.
 - Provide a link to where the exploit can be found, in the analysis section.
 - If the exploit comes from Metasploit, Trend Micro or Exodus, please mark them accordingly.
 - If the exploits can be covered by an existing signature, please declare this in the analysis section.
9. If necessary, add a **new signature** to cover the vulnerability, and set its **attributes**. This is very important when we review the report in the future.
 - If the signature is converted from a Snort rule, please mark it.
 - If the signature is for ICSA/NSS, please mark them.
 - If the signature covers Metasploit, BP, Trend Micro or Exodus, please mark them accordingly.
 - If the signature is exploit based, please explain this.
10. Upload any **attachments**, such as pcaps, and set the correct attributes, including their "**Type**" and "**Source**". This is very important when we review the report in the future. If the source is from Trend Micro or Exodus, the report they provide should be uploaded as well. Listed below are the requirements for the traffic file:
 - Currently only pcap, pcapng and zip (contains only pcap/pcapng files) format file are acceptable. Don't use zip format unless it contains more than 3 traffic files.

- Analysts should capture the decrypted traffic for signatures that need deep inspection. This can be done with tools like Fiddler and fiddler2pcap.
- Analysts should leave a note in the “Notes” section on ISTANCE if no traffic file can be provided. The note can be as simple as “Can’t provide pcap for aid ###”.

11. Set the **OS** and **Application**.

OS and Application settings are shown on the Fortigate GUI for our customers.

Many of our customers use the settings for configuring their sensor. Analysts should always set the OS to the correct setting when creating a report on ISTANCE.

- They will be shown on the Fortigate WebUI.
- For PHP applications, set OS to “[Windows/Linux/BSD/Solaris/MacOS](#)”.
- **For the setting of “All” applications, do not set any other OS, as the setting includes all OS.**

12. Set **Vulnerability Type**

Vulnerability Type will be used to generate reports on Fortigate and Fortianalyzer.

Currently we have following vulnerability types:

- N/A
- DoS
- BufferErrors (CWE-119)
- NumericErrors (CWE-189)
- CSRF (CWE-352)
- XSS (CWE-79)
- PathTraversal (CWE-22)
- CodeInjection (CWE-94)
- SQLInjection (CWE-89)
- FormatString (CWE-134)
- Malware
- Anomaly
- ImproperAuthentication (CWE-287)
- InformationDisclosure (CWE-200)
- Permission-Privilege-AccessControl (CWE-264)
- OSCommandInjection (CWE-78)
- ResourceManagementErrors (CWE-399)
- Other

This list is mainly based on CWE (Common Weakness Enumeration) maintained by the MITRE Corporation. The vulnerability type followed by a CWE id are from CWE and you can find their definition at <http://nvd.nist.gov/cwe.cfm#cwes>. Other types are quite straightforward:

N/A: Hidden IPS signatures for example vid 15306

DoS: This is in fact an impact type not vulnerability type. I added it in the list because It is easier for customers to understand the vulnerability.

Malware: IPS signatures for Backdoor, Worm, and Trojan

Anomaly: IPS signature for protocol anomaly and suspicious traffics for example scanning and brute force attack.

Other: IPS signatures that don't fit any other types

13. Fill in all sections in the **encyclopedia description** and set the description status to **Releasable**.

- This will be shown in the IPS Encyclopedia. See **IPS Vulnerability Description** in the **Encyclopedia** section for details.

14. Set the **Default Status** and **Default Action**.

- The Default Action when the signature is in the **beta state** should be **pass**. Please give your suggested action in the comment, so that we can set it based on your suggestion.

15. Select the signature's release status from the **Signature Status** list.

- **"Releasable"** means it is ready for release. Release QA will test and release it.
- **"New Engine Testing"** means the signature will only work with a new engine version. It will be tested and released when the new engine version is released. Please add the engine version required in the comments to assist QA in knowing when to release the signature.

16. Set the **status of the Vulnerability report**.

- **"DONE"** means it has been completed, and we do not need to monitor it.
- **"MONITOR"** means that the analysis is not completed for some reason, and we need to monitor it.

To modify a vulnerability, the relevant information listed above should be completed. If a signature is modified or a new signature is added, **reasons should be given as a note**. To maintain consistency, signatures should be modified on all platforms if they have the same patterns. When an old signature is modified, please make sure that it can cover all relevant exploits that the signature used to cover.

4.2 Adding or Modifying an application

1. Provide the appropriate **application name**.
2. Provide **a link to the vendor's website**.
3. Set the appropriate **Severity**.
 - For application control the severity should be **"info"**.
4. Analyze the application and **fill in the Analysis section**. It should contain the following:
 - What is the version of the application you analyzed, if applicable?
 - What did you select as the detection condition in the signature? Please explain

why, if it is not obvious.

You may leave the “Analysis” section empty when creating the report and fill it later. ISTANCE does not allow you to release a signature with the “Analysis” section not filled.

5. If necessary, **add a new signature** to cover the application.
6. Upload any **attachments** and set the correct attributes including their “**Type**”. Listed below please find requirements for the traffic file:
 - The attachment purpose should be set to “**application traffic**” for the application (including Botnet) traffic file.
 - Currently only pcap, pcapng and zip (contains only pcap/pcapng files) format files are acceptable. Don’t use zip format unless it contains more than 3 traffic files.
 - Analysts should capture the decrypted traffic for signatures that need deep inspection. This usually can be done with tools like Fiddler and fiddler2pcap.
 - Analysts should leave a note on ISTANCE if no traffic file can be provided.
7. Set the **OS** and **Application**.
 - This information will be shown on the Fortigate WebUI.
8. Fill in all sections in the **encyclopedia description** and set the description status to **Releasable**.
 - This information is shown in the IPS encyclopedia. See **IPS Vulnerability Description** in the Encyclopedia section for details.
9. Set the **Default Status** and **Default Action**.
 - The **Default Action** should be set to **Pass**, and the **Disable checkbox** should be selected to set the default status.
10. Select the signature’s release status from the **Signature Status** list.
 - “**Releasable**” means it is ready for release. Release QA will test and release it.
 - “**New Engine Testing**” means the signature will only work with a new engine version. It will be tested and released when the new engine version is released. Please add the engine version required in the comments to assist QA in knowing when to release the signature.

It is **required** to leave a comment for the new added or updated signature if the report contains multiple signatures. The comment should explain why this change is necessary and contain related application version information.

11. Add Icons

Following please find the requirement of app icon:

1. Format: PNG file

2. Dimension: 96x96 for large icons, 32x32 for medium icons and 16x16 for small icons (Please reduce image from 96x96 downwards, instead of expanding the image from 16x16 upwards as we want to maintain the quality of the image throughout the pixel size changes)
3. Depth: RGB color 8 bits
4. Background: Transparent

Following free tools are available for use:

<http://www.coolutils.com/online/image-converter/>

This site can resize icons or change their formats.

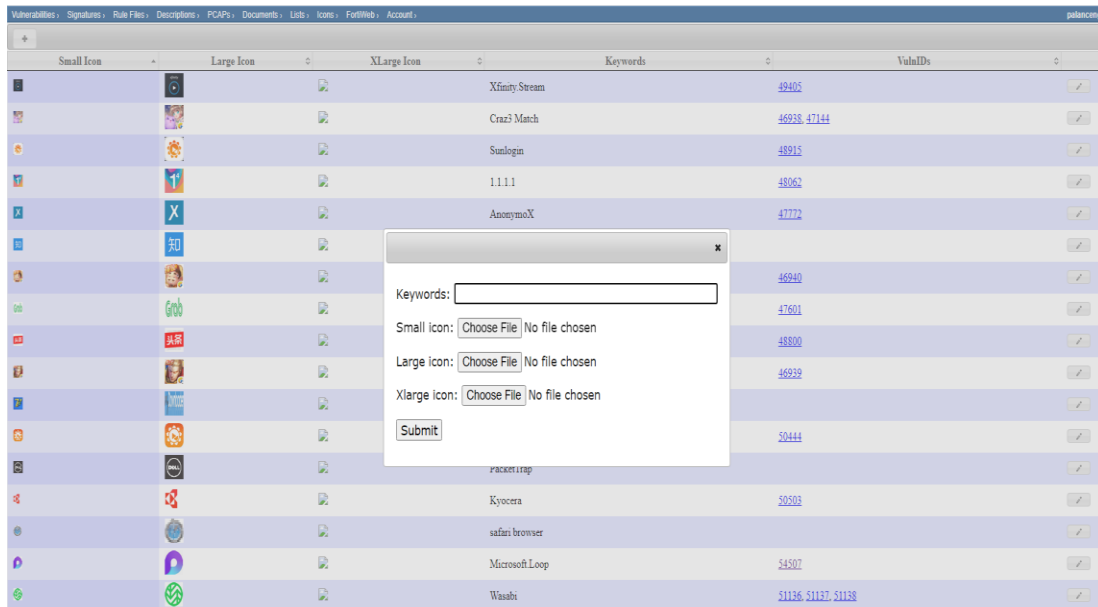
- Select browse to choose the picture to upload
- Select a format to convert into (we use PNG for icons.)
- On the right-side panel under “options”, click on “resize”
- Input the width and height that you want (32x32 for large icons, 16x16 for small icons and 96x96 for xlarge icons)
- Download the converted file

http://www.adobe.com/downloads/cs2_downloads/index.html

We can create an RGB 8bit PNG image with transparent background in Photoshop.

- File->Open and select your picture
- Ctrl+a and Ctrl+c to select all and copy
- File->new. You will see a setting panel for the new document.
- Set width and height to 32, 16, or 96 pixels, set color mode to RGB Color 8 bit, and set background contents to Transparent
- Ctrl+v to paste the image
- In your toolbox, click the magic wand icon.
- Use the magic wand to click on any background that you want to remove. Use delete key to clear the background color.
- When you are done, File->save. Give the file a name and select the format PNG.

Upload generated icon file to ISTAME as shown in following image



12. Edit application setting

- Select Icons for the application, for example the one added in last step
- Set Popularity and Risk level
- Select Vendor and set Vendor name and URL
- Select application category for 4.0 and 5.0 respectively
- Select sub-category for 4.0 if necessary
- Mark Behavior and technology list properly
- Set require_ssl_di
- Add the other metadata, for example, the app_port, in the body section if needed as below.

Body:

```
--app_port "TCP/80,TCP/443";
```

Only after you added the application setting, the signature became an AppCtrl signature!

13. Set the **status of the Vulnerability report**.

- **“Done”** means it is completed, and we do not need to monitor it.

- **“Monitor”** means that analysis is not completed for some reason, and we need to monitor it.

4.3 Adding or Modifying a Signature

1. To start, click the "+" button in the signature section on the main page for the vulnerability or application.
2. Add signature body and/or comments.

For **vulnerability signatures** include **--protocol** and the **detection options**.

3. Mark platforms and other flags accordingly
ISTAME 2.1 support five platforms:
 - **3.0**: use 1.0 IPS engine (Deprecated)
 - **4.0**: below 4.3.12. uses 1.0 IPS engine
 - **4.3**: above 4.3.12. use 2.1 IPS engine
 - **5.0**: use 2.1 IPS engine
 - **5.0 extended**: use 2.1 IPS engine and above

No more signatures should be added for the FortiOS 3.0 platform.

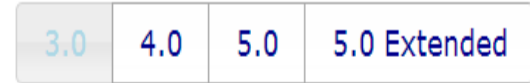
- Default setting (4.0/5.0/5.0 extended/4.3) is good for most cases
- Always set 5.0 and 5.0 extended for new IPS/AppCtrl signatures. This is to guarantee an effective beta process. the 5.0 extended database is only supported in the few high-end FGT devices (the smallest model supports the extended database is 100D) while a lot of our customers use low-end models, like 60C, 60B, 80C, etc. **New signature that doesn't have a 5.0 flag set will be rejected by QA!** If a signature has high risk of FP, it will be added into the beta signature review list on [PMS](#) after the beta process. You can unset 5.0 flag after you review this beta signature on PMS.

4. Click the **Submit** button to save the signature.

Status: Protocol: Is pending: ☐ Is urgent: ☐ Is disable: ☒ Is hidden: ☐ Is forced: ☐ Is evaluate: ☐

Test source list:

Firmware list:



Name:

Body:

```
--service HTTP; --flow from_client,reversed; --parsed_type HTTP_GET; --pattern "/getfile.php?r="; --context uri;  
20,context; --pattern "&p="; --context uri; --within 20; --pattern "!&"; --context uri; --distance 0;
```

Comment:

it's created to replace the signature with attack_id=22783(change flow reversed flag)

Figure 3.3

5 Signature Naming Convention

5.1 IPS signature

We use the following naming convention for IPS signature:

[CompanyName.]ProductName.VulnerableFile.VulnerabilityType

MalwareType.MalwareName

ProtocolName.fieldname.issuename

[CompanyName.]ProductName.VulnerabilityName.VulnerabilityType

[CompanyName.]ProductName.VulnerableFunction.VulnerabilityType

The signature name should be standardized across all the signatures on ISTANCE. When a new signature is added, and the analyst is unsure of the name to use, they should check ISTANCE to see if there are previously added signatures for the Company which he is adding.

For example, 7-Technologies, for reports related to this Company, we had a few different names.

7Technologies.IGSS.SCADA.System.Directory.Traversal

7T.IGSS.ODBC.Server.Memory.Corruption

IGSS.SCADA.System.Memory.Corruption

The third signature name is obviously wrong as it does not contain the company name. Besides that, the variants in names lack professionalism in our work when our customers look at them. This sort of issue should be avoided by checking ISTANCE for the signature name.

When unsure of the company, analysts should do a search on the web for information on the company and product.

Since 2014, many researchers have been naming their vulnerabilities findings to increase its marketing value. For example, Shellshock, Heartbleed, etc. For such cases, the name can be added to the signature so customers can search for it on Fortiguard. A caveat is that such a name should make sense to appear in the signature name. For example, Wannacry, the malware that was terrorizing major organizations in 2017. It is a malware that spread through a SMB vulnerability in Microsoft windows. For such cases, it would not make sense to add it to our common name.

Notes:

1. Company name can be omitted if it is same as product name
2. Only alphabetic, numeric, dot and hyphen are allowed
3. Following abbreviations can be used:
 - MS (Microsoft), IBM, SUN, HP, SAP, CA (Computer Associates)
 - IIS, IE, DB2
 - SQL, OS, BSD
 - CGI, ASP, PHP, HTML (All file extensions should be in uppercase)
 - ActiveX, COM, DCOM
4. Following vulnerability/malware types are suggested:

- Buffer.Overflow, Integer.Overflow
 - Format.String
 - XSS
 - SQL.Injection
 - Directory.Traversal
 - Remote.File.Inclusion
 - Double.Free
 - xxx.Bypass
 - Command.Execution, Code.Execution
 - Memory.Corruption
 - Information.Disclosure
 - DoS
 - Overlong
 - Brute.Force
 - Spoofing
 - Insecure.Library.Loading
 - Arbitrary.File.Upload
 - Use.After.Free
 - Backdoor/Worm/Trojan
 - SSTI (For Server Side Template Injection)
 - SSRF or CSRF (For Server Side or Cross Site Request Forgery)
5. The vulnerable function name should be indicated exactly as found in official documentation for the function.
 - Flash.attachMovie (not Flash.AttachMovie)
 6. If there are no details being provided about the vulnerability and the analyst is unable to come up with a signature name. The naming convention can be done using CVE.
 - Adobe.Flash.CVE-2017-3068.Memory.Corruption
 - MS.IE.CVE-2017-8618.Memory.Corruption
 7. For vulnerabilities with marketing names, when there are 2 or more names associated with the report, a regular common name should be used. For example, meltdown and spectre were both covered in vid 45413. We use the name "CPU.Speculative.Execution.Timing.Information.Disclosure" instead of "CPU.Meltdown.Spectre.Information.Disclosure".

Examples:

Adobe.Reader.	(not Adobe.Reader.Acrobat or Adobe.Reader.And.Acrobat)
Adobe.Flash.	(not Adobe.Flash.Player)
MS.Windows.	(not Windows., for example, MS.Windows.PnP.DoS)
MS.Word.	(not Word. or MS.Office.Word.)
MS.PowerPoint.	(not Power.Point. or MS.Power.Point.)
MS.Excel.	(not Excel.)

MS.Office.	(for example, MS.Office.Malformed.String.Buffer.Overflow)
MS.SQL.Server.	(not SQL.Server. or MS.SQLServer.)
MS.Messenger.	(not MS.MSN. or Messenger.)
MS.Outlook.	(not Outlook.)
MS.IIS.	(not IIS.)
MS.IE.	(not IE.)
MS.Browser	If it affects both IE and Edge
MS.Exchange.Server.	(not MS.Exchange.)
MS.Frontpage.	(not Frontpage.)
MS.Publisher.	(not Publisher. because Publisher is not a known product.)
MS.MediaPlayer.	(not Media.Player. or MediaPlayer.)
MS.SMB.	(not SMB.)
MS.RPC.	(not RPC.)
Mozilla.Firefox.	(not FireFox.)
Mozilla.Graphics.Features.Integer.Overflow	(not Mozilla.Products.Graphics.)
Apache.	
MySQL.	
Oracle.	
TrendMicro.	(not Trend or Trend.Micro)
CGI.	(for cgi web app, e.g. CGI.A1stats.A1disp.Directory.Traversal)
PHP.	(for php web app, e.g. PHP.AttilaPHP.global.php3.SQL.Injection)
Virus.	(for viruses)
Worm.	(for worms, for example, Worm.Blaster)
Trojan.	(for trojans)
Backdoor.	(for backdoors, for example, Backdoor.Glacier)
HTTP.Host.Header.Too.Long	
IMAP.Select.Command.Directory.Traversal	
FTP.STOR.Command.Format.String	
SCADA	(not scada or Scada, should all be in uppercase)
App	(not APP, it can be lowercase if it is a function or file name)
7-Technologies	(not 7T, 7Technologies)
3S-Smart	(not 3S, 3S.Smart)
JavaScript	(not javascript or Javascript)
H.264	(not H264)
Quicktime.STSD	(not stsd or Stsd)
TrueType.Font	(not TrueTypeFont or True.Type.Font)
UTF-16	(not UTF16)

5.2 AppCtl Signature

We use the following naming convention for application signatures.

Main App: [VendorName.][ProductName.]AppName[.Feature]

Sub App: MainAppName_Feature

Botnet: Botnetname.Botnet

Notes

1. Underscore character is used to indicate the relationship between main app and sub app

Examples

Fetion

Baidu.Hi

MS.Office.Communicator -> main app

MS.Office.Communicator_Audio -> sub app

MS.Office.Communicator_Video -> sub app

Rediff.Messenger -> main app

Rediff.Messenger_File.Transfer -> sub app

Rediff.Messenger_Video.Chat -> sub app

Morto.Botnet

Night.Dragon.Botnet

6 Vulnerability Severity and Reference

6.1 Severity

One of the tasks to be completed when working on a vulnerability is selection of the severity level. A severity level must be determined for each vulnerability. Its purpose is to help our customers to understand the risk that the vulnerability can deal to their IT infrastructure and to assist them in planning their updates accordingly.

For many years, the severity rating had to be determined by each analyst, based on their own analysis, experience and skills. However, this changed in 2005 with the creation of [CVSS](#) (Common Vulnerability Scoring System).

The CVSS provides a universal, open and standardized method for rating the severity of IT vulnerabilities. It outlines a way to calculate a severity rating as a numerical score that is independent of the analyst. The latest CVSS v3 scoring standard was released in June 2015, available at <https://www.first.org/cvss/specification-document>.

The severity level of vulnerabilities in our system should be based on their CVSS v3.1 score. Currently we have five severity levels: Critical, High, Medium, Low and Info. The guide below shows how we decide on a severity level for a vulnerability.

This is how we map the **CVSS base score** to our five severity levels:

- **Critical** 9 <= CVSS base score <= 10
- **High** 7 <= CVSS base score < 9
- **Medium** 4 <= CVSS base score < 7
- **Low** 0 < CVSS base score < 4

- **Info** `CVSS base score == 0`

Using the above metrics, the CVSS High rating is broken into our High and Critical, and the CVSS Low rating is not changed.

If no CVSS score is available, the National Vulnerability Database **CVSS calculator** can be used to get the **Base Score**, so that the corresponding severity level can be found. The calculator tool is available at the link below:

<https://www.first.org/cvss/calculator/3.1>

Note.

1. If a CVE covers multiple vulnerabilities, for example CVE-2016-1000113 covers SQL injection and XSS vulnerabilities in the Joomla! Huge-IT Image Gallery Extension, while our signature, for example, Cross.Site.Scripting, only covers partial of them, care should be taken to make the appropriate severity be selected.
2. If a report has multiple CVEs which have different scores, choose the highest one for the severity setting.

6.2 Reference

Analysts can add one reference when creating a report. More references can be added later by clicking "Add Foreign Reference" in the report. You can only input "**Reference Number**" for following report source:

CVE: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=xxxx-xxxx>

For example, to add CVE reference 2011-0611:

Report Source: CVE

Reference Number: 2011-0611

ISTAME will automatically generate the link <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2011-0611> for you.

Note.

The CVE number is added into signature in 5.0 platform to make CVE ID searchable on Fortigate. For example,
F-SBID(--vuln_id 26538; --cve 20110611; ...)

Analysts need to provide reference links for other report sources.

7 Encyclopedia

For each IPS or App Control signature we release, their corresponding ISTANCE report requires an encyclopedia entry along with it. The encyclopedia is part of the service that we provide to our customers. The encyclopedia provides them with summarized information of the vulnerability to assist them in decision making.

Multiple signatures covering the same vulnerability or application can have the same encyclopedia entry if they are in the same report on ISTANCE, as the encyclopedia information is entered through ISTANCE. There are four distinct text areas to fill in: **Description**, **Vulnerability**, **Impact** and **Recommended Action**. Their contents will be described in detail below. When all fields are filled, the status should be set to **Releasable**. The entry will then be reviewed and approved, before the description is submitted to Fortiguard Center.

Although we can use competitors' descriptions as reference, we need to use our own words in the encyclopedia entry. **To avoid causing legal issues over plagiarism, it is forbidden to copy directly from other websites or advisories.**

For more information on plagiarism, please refer to:

http://www.plagiarism.org/learning_center/what_is_plagiarism.html

On the website, "plagiarism" is defined as:

1. To steal and pass off (the ideas or words of another) as one's own.
2. To use (another's production) without crediting the source.
3. To commit literary theft.
4. To present as new and original an idea or product derived from an existing source.

If quoting statements from a website is required, it is necessary to cite the source and acknowledge that these statements were copied from the website.

7.1 IPS Encyclopedia

7.1.1 Description

For vulnerability detection signatures, the Description section is used to provide details of a vulnerability or exploit. This section requires the following contents:

- The **name** of the vulnerable software, but in this section do not include version numbers with the name. Version numbers should be in the **Vulnerability** section.
- The **type** of vulnerability (e.g. buffer overflow, SQL injection, cross-site scripting, format string, remote file inclusion, etc.)
- The **cause** of the vulnerability (e.g. failure to perform boundary checks on user-supplied data, a logic error, design error, etc.)
- The **exploit method** or condition (e.g. the vulnerability can be exploited by a local or remote, authenticated or unauthenticated attacker to ... via ...)
- Exploit **impact** (e.g. denial of service (crash), arbitrary code execution, sensitive information leak, etc.)

- If the signature is rate-based, the default rate setting should be mentioned in the description so customers can adjust it if needed

Note: The names of specific functions, methods, header fields, etc. should be put in quotes to make it clear that these are proper names and not general descriptions.

Below are some example descriptions that can be used as templates.

1. **Buffer overflow example:**

This indicates an attack attempt against a Buffer Overflow vulnerability in Microsoft DirectX Media SDK.

The vulnerability is caused by an error when the vulnerable software handles a malicious "SourceUrl" property. An attacker can trick an unsuspecting user into visiting a malicious webpage and execute arbitrary code within the context of the application.

2. **ActiveX example:**

This indicates an attack attempt to exploit a Memory Corruption vulnerability in RealNetworks' RealPlayer.

The vulnerability, which is found in the "rmoc3260.dll" ActiveX control, can be exploited through misuse of the "Control" property. It may allow remote attackers to execute arbitrary code in the context of the application, using the affected ActiveX control. Failed exploit attempts will likely cause the program to crash, resulting in a denial-of-service condition.

3. **RPC example:**

This indicates an attack attempt against a heap-based Buffer Overflow vulnerability in Samba server.

The vulnerability is caused by improper bounds checking in the "lsa_io_privilege_set" function. By sending a specially crafted RPC request to the LSA RPC interface, a remote attacker could overflow a buffer and execute arbitrary code on a vulnerable system.

4. **Web application example:**

This indicates an attack attempt against a remote Command Execution vulnerability in Raxnet's Cacti software.

The vulnerability is due to the application's user input filters failing to properly sanitize the "graph_start" parameter value that is passed to "graph_image.php" and "graph.php". An attacker may include shell commands by supplying an injection string through the URL and a good string through the request method POST or the header field COOKIE.

5. SQL Injection example:

This indicates an attack attempt to exploit a SQL Injection vulnerability in SaphpLesson.

The vulnerability is a result of the application's failure to properly sanitize user input before using it in a SQL query. As a result, a remote attacker can send a crafted query to execute SQL commands on a vulnerable server.

6. File based examples:

a.

This indicates an attack attempt to exploit a remote Code Execution vulnerability in Adobe Flash Player 9.

The vulnerability results from insecure code in the DLL responsible for parsing SWF tags. It can be exploited via a crafted Flash file (SWF), leading to remote code execution.

b.

This indicates an attack attempt to exploit a Double Free vulnerability in Microsoft Word.

The vulnerability is caused by an error that occurs when the vulnerable software handles a malicious DOC file. A remote attacker may exploit this to execute arbitrary code via a crafted DOC file.

7. Virus example:

This indicates detection of network traffic originating from a computer infected with the Krackin version of the Storm worm.

Storm worm is a virus that uses P2P networks and other methods to spread and launch spamming attacks. The Krackin version communicates through encrypted P2P traffic.

8. Cross-Site Scripting example:

a.

This indicates an attack attempt to exploit a Cross-Site Scripting vulnerability in WordPress Super Cache Plugin.

The vulnerability is due to insufficient sanitization of user-supplied inputs in the application. A remote attacker may be able to exploit this to inject malicious script code into the web pages. When an unsuspecting victim views these crafted web pages via a web browser, the injected script code will be automatically executed.

b.

This indicates an attack attempt to exploit a Cross-Site Scripting vulnerability in

Microsoft Sharepoint.

The vulnerability is due to insufficient sanitization of user supplied inputs in the application when handling a crafted HTTP request. A remote attacker may be able to exploit this to execute arbitrary script code within the context of the application, via a crafted request.

9. Same Domain Policy example:

This indicates an attack attempt to exploit a Security Bypass vulnerability in Adobe Reader and Acrobat.

The vulnerability is due to an error when the vulnerable software handles a maliciously crafted PDF file. A remote attacker may be able to exploit this to bypass the Same Origin Policy of the application, via a crafted PDF file.

10. Information Spoofing example:

a.

This indicates an attack attempt to exploit an Information Spoofing vulnerability in Microsoft Internet Explorer.

The vulnerability is due to an error when the vulnerable software attempts to handle a maliciously crafted web page. An attacker can exploit this by tricking a user into visiting a malicious webpage and serving spoof contents to the unsuspecting user.

b.

This indicates an attack attempt to exploit an Information Spoofing vulnerability in Microsoft Outlook.

The vulnerability is due to an error when the vulnerable software attempts to handle maliciously crafted email. An attacker can exploit this by tricking a user into opening a crafted email and redirect the user to another site, leading to further attacks.

11. Rate Based example

This indicates an attack attempt to exploit a Denial of Service vulnerability in Microsoft Office 2010 beta.

The vulnerability is a result of the application's failure to properly sanitize user-supplied inputs, which allows remote attackers to cause a denial of service (memory consumption) via many SIP INVITE requests. The signature detects 100 malicious requests within 5 seconds. The threshold is configurable based on the user's

environment.

12. Use-After-Free example:

This indicates an attack attempt to exploit a Use-After-Free vulnerability in Microsoft Internet Explorer.

The vulnerability is due to an error when the vulnerable software tries to access a deleted object. An attacker can exploit this by tricking an unsuspecting user into visiting a malicious webpage and execute arbitrary code within the context of the application.

Notes:

- Do not include detailed information on how we detect the vulnerability. We do not want to let the attackers know how to avoid detection.
- Do not include PoC or code. We do not want to inform the public of how to exploit the vulnerability.
- Do not write anything that you cannot verify from your own analysis. You may ask for help from other analysts, and you can add a note in your report to give your thoughts to the IPS Review Team.
- Do not include vendor/product links in this section. They should be added to **References**.
- Provide some external references to CVE whenever possible through the **References** part of the report. These are provided to help our customers to cross-check the online descriptions and get more detailed information regarding the vulnerability.
- Keep your sentences concise
- Make sure the common name aligns with the description. If the common name is XXX.YYY.Directory.Traversal, the description should not be XXX.YYY.Memory.Corruption.

7.1.2 Vulnerability

For attack signatures, list all known vulnerable software versions and put each item on a separate line. For example:

Haudenschilt Family Connections 0.1.2
Haudenschilt Family Connections 0.1.1
Haudenschilt Family Connections 0.8
Haudenschilt Family Connections 0.6
Haudenschilt Family Connections 0.5

Or if there are too many, you can follow one of the examples below:

Akamai Download Manager ActiveX Control versions prior to 2.2.3.6

HP OpenView Network Node Manager 7.53 and later versions

SCO OpenServer 5.0.5 and earlier versions

If the vulnerability only occurs on a given OS platform (name/version), we should specify it in this section. For example:

Microsoft Office 2003 Service Pack 2

7.1.3 Impact

For attack signatures, provide the impact on the system if the vulnerability is successfully exploited. The recommended format is **Type [Description]**:

Where **Type** contains one of:

- System Compromise
- Denial of Service
- Information Disclosure
- Privilege Escalation
- Security Bypass
- Information Spoofing

And **Type: Description** can be used as follows:

- **System Compromise:** Remote attackers can gain control of vulnerable systems.
- **System Compromise:** Remote attackers can execute arbitrary script code within the context of the target user's browser
- **System Compromise:** Remote attackers can execute arbitrary script code in the context of the affected site
- **System Compromise:** Remote attackers can add, view, delete or modify data in the database of the affected application
- **System Compromise:** Remote attackers can access or modify data in the database of the affected application
- **Denial of Service:** Remote attackers can crash vulnerable systems.
- **Information Disclosure:** Remote attackers can gain sensitive information from vulnerable systems.
- **Privilege Escalation:** Remote attackers can leverage their privilege on the vulnerable systems.
- **Security Bypass:** Remote attackers can bypass security features of vulnerable systems.
- **Information Spoofing:** Remote attackers can spoof data of vulnerable systems.
- **Information Spoofing:** Remote attackers can serve spoof contents to unsuspecting targets.

Note: We only provide the most severe impact in this section.

7.1.4 Recommended Action

For attack signatures, we should provide a patch or update information if a fix is available. If there is no fix available currently for the issue, we should, if possible, provide a

recommendation. For example,

For **vendor patched or workaround** cases:

- Upgrade to the latest version, available from the website.
<http://xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
- Apply patch, available from the website.
<http://xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
- Apply the latest update from the vendor.
<http://xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
- Apply the most recent upgrade or patch from the vendor.
<http://xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
- Refer to the vendor's website for suggested workaround.
<http://xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

For **unpatched** cases:

- Disable xxxxxxxxxxxx if not needed.
- Disable this ActiveX Control by setting its kill bit, by the method shown on the website:
<http://support.microsoft.com/kb/240797>.
- Use AntiVirus software to scan and clean the system.
- Currently we are unaware of any vendor supplied patch for this issue.

We should try to **use "N/A" as little as possible**. In most cases it is possible to write something a little more informative. Here are some examples:

- There is no vulnerability associated with this signature.
- This signature's action can be set to "Block" to protect against this threat.
- This may indicate an attempted probe or attack.
- This indicates detection of traffic that does not comply with the protocol standard.
Monitor the traffic from that network for any suspicious activity.

When adding signatures for old vulnerabilities, always check if there is a patch or update for it even if the old description says, "we are unaware of any patch or update for this issue". As the vulnerability is old, it may have been fixed but we did not update the description. If there is a patch or fix for it, please update the Recommended Action appropriately.

7.2 Application Control Encyclopedia

7.2.1 Description

For Application Control signatures, the Description section is used to provide information on an application or protocol, not a vulnerability or an exploit. This section requires the following:

- The name of the application or protocol, but do not include version numbers in this section.
- A brief introduction to the application or protocol and its uses.

The **example description** below can be used as the template.

This indicates an attempt to access Gmail.

Gmail is a web-based email service provided by Google. It has a free version and a business version - the business version with the email extension @gmail.com replaced with @xxxxx.com. It is an advertising supported email service and can be accessed via secure webmail or through the IMAP/POP3 protocols. Within the Gmail webpage, users can use the Google Hangout service to chat with other contacts. It is one of the most popular email service providers in the world.

Always try to write a more descriptive description. If possible, write up a description based on your understanding of the application. Otherwise, we can look up Wikipedia and do a summary. For the parent app, we should write a longer description. For the sub apps like Gmail_Attachment, we don't need to repeat the description.

The **example sub app description** below can be used as the template.

This indicates an attempt to download or upload attachments on Gmail.

Gmail Attachment detects if a file is being downloaded or uploaded by the user in a Gmail message.

For a **DAC signature**, we need to add an extra section. **Example template** is as below.

This indicates an attempt to download attachments from Gmail.

Gmail Attachment Download detects and logs the name of the file downloaded by the user in an email message.

Information Displayed:

User's IP/Username/Email Address depending on traffic

File Name

Again, we follow the template for the first line, "**This indicates an attempt to** <do something>." The second line should have the word "**detect(s) and log(s)** <something>". Finally, the third line should include the information displayed. The possible options are:

CASI		Description	Information Displayed
Control		Displays the username, client IP or userID of the client.	User's IP/Username/Email Address depending on traffic

File_info		Display the file name.	File Name
Video_info		Display the video name.	Video Name
Message		Display email subjects, messages, chats.	Message Information/Email Subject
Search_phrase		Display search text string.	Search Phrase

The DAC information can be obtained at Application Settings.

For an **Industrial signature**, we use the following template:

This indicates detection of the MMS protocol.

For an **Industrial signature subcommand**, we use the following template:

This indicates detection of the MMS Read Coil command.

For a less widely known application or protocol, a brief description of it can help customers to decide if it is relevant to their business. For example, Thunder, aka, Xunlei, is only well known in China. We also should give a short explanation of its purpose.

Note:

- Our descriptions have a standard format to it. So, the first line should start with “**This indicates an attempt to <do something>.**” or “**This indicates detection of <protocol name>**”
- **Try our best to eliminate spelling mistakes. Always proofread what you wrote. Typos and mistakes lower our credibility and make us look unprofessional.**
- For AppControl signatures, **do not put** "this does not indicate an attack or exploit" or similar sentences in this section.
- Do not include vendor or production links in this section. They should be added to the **References** section.

7.2.2 Vulnerability

This will be renamed in the future. For now, we list only the parent app. For example:

1. **Gmail example:**
Gmail
2. **Gmail Attachment Upload example:**
Gmail
3. **Gmail Attachment example:**
Gmail
4. **BitTorrent example:**

BitTorrent
BitComet
Azureus
Utorrent

Note:

- We do not need to list the versions of the apps anymore like before.
- For unique protocols like BitTorrent which is used by multiple applications, we list the application names.
- Capitalize the first character and **do not add a full stop after the name**.

7.2.3 Impact

For AppControl signatures, we can list the characteristics of the application or protocol which could affect a customer's business. For example:

- Network bandwidth consumption.
- Unexpected network communication.
- Firewall policy avoidance.

7.2.4 Recommended Action

For Application Control signatures, we can use one of these:

- If required, this signature's action can be set to "Block" to block this application.
- You can set this signature's action to "Block" if this application is not allowed in your network.
- The signature can be set to "Block" if access to the site is against network usage policy.

For Application Control signatures that require deep inspection, please add the following line to the **Description section**:

- Note: This signature requires "deep-inspection" enabled on the Fortigate.

7.2.5 Example

Gmail

Description:

This indicates an attempt to access Gmail.

Gmail is a web-based email service provided by Google. It has a free version and a business version - the business version with the email extension @gmail.com replaced with @xxxxx.com. It is an advertising supported email service and can be accessed via secure webmail or through the IMAP/POP3 protocols. Within the Gmail webpage, users can use the Google Hangout service to chat with other contacts. It is one of the most popular email service providers in the world.

Vulnerability:

Gmail

Behavior list:

Reasonable

Gmail_Attachment

Description:

This indicates an attempt to download or upload attachments on Gmail.

Gmail Attachment detects if a file is being downloaded or uploaded by the user in Gmail message.

Vulnerability:

Gmail

Behavior list:

Excessive-Bandwidth

Gmail_Attachment.Upload

Description:

This indicates an attempt to upload attachments on Gmail.

Gmail Attachment Upload detects and logs the name of the file uploaded by the user in an email message.

Information Displayed:

User's IP/Username/Email Address depending on traffic

File Name

Vulnerability:

Gmail

Behavior list:

Excessive-Bandwidth

Gmail_Attachment.Download

Description:

This indicates an attempt to download attachments from Gmail.

Gmail Attachment Download detects and logs the name of the file downloaded by the user in an email message.

Information Displayed:

User's IP/Username/Email Address depending on traffic

File Name

Vulnerability:

Gmail

Behavior list:

Excessive-Bandwidth

Gmail_Send.Message

Description:

This indicates an attempt to send an email from Gmail.

Gmail Send Message detects and logs the name of the recipient and subject of an email that is sent by the user.

Information Displayed:

User's IP/Username/Email Address depending on traffic

File Name

Vulnerability:

Gmail

Behavior list:

Reasonable

In Gmail_Attachment, Gmail_Attachment.Download and Gmail_Attachment.Upload, since it involves file transfer, we classify it as Excessive-Bandwidth. Things might be harder to decide with Gmail since it is the parent signature of the sub signatures. If customers do not use deep-

inspection, all Gmail_Attachment will trigger as Gmail and we could say it is Excessive-Bandwidth too. **So, another assumption we make when selecting the behavior is: deep-inspection is enabled.** That way, Gmail won't cover the file transfer part and so it is not under Excessive-Bandwidth.

7.3 Encyclopedia Template for Zero-Day Vulnerabilities

When releasing signatures for zero-day (0day) vulnerabilities, we need to hold back the details until the affected vendors release their patch, or until the details are publicly disclosed by a third party. While information is being withheld, the analyst should use the following template for the encyclopedia.

7.3.1 Template for Zero-Day Vulnerability Received from Vendor/3rd-Party

[Signature Name]:

VendorName.0day.IstameReportID

[Description]:

This indicates detection of a Zero-Day vulnerability protected by a signature from Fortinet's FortiGuard Labs.

This signature should help mitigate the threat proactively both prior to, and after an official statement is available from the vendor.

Once an official advisory or statement is available from the vendor, the signature name and its description will be updated to provide more details regarding this vulnerability.

Further details may also be made available in an advisory on FortiGuard Center (<http://www.fortiguards.com>).

.

[Vulnerability]:

This is a Zero-Day (unpatched) vulnerability that is currently being investigated by Fortinet's FortiGuard Labs.

[Impact]:

Any Zero-Day vulnerabilities can have a large impact because they remain unpatched, making exploit attempts more effective since the given attack vector remains open.

[Recommended Action]:

N/A

7.3.2 Template for Zero-Day Vulnerability Discovered by Ourselves

[Signature Name]:

FG-VD-xx-xxx-VendorName (FG-VD-xx-xxx is our Vulnerability Discovery Tracking Number)

[Description]:

This indicates the detection of a Zero-Day vulnerability discovered by Fortinet's FortiGuard Labs.

This signature should help mitigate the threat proactively - both prior to, and after an official fix is available from the vendor.

Once this official fix is available, further details about our discovery will be made available in an advisory on our FortiGuard Center (<http://www.fortiguards.com>). The signature name and description will also be updated then.

[Vulnerability]:

This is a Zero-Day (unpatched) vulnerability that has been discovered by Fortinet's FortiGuard Labs.

[Impact]:

Any Zero-Day vulnerabilities can have a large impact because they remain unpatched, making exploit attempts more effective since the given attack vector remains open.

[Recommended Action]:

N/A

8 Things need to know about signature

8.1 IPS Engine Logic

8.1.1 Service Trees

In IPS Engine 3.0, if a signature uses the "**--service**" keyword it will be added into **its own service tree**. If there is no "**--service**" keyword, but it has the "**--port**" keyword, it will be added into the **unknown_service** tree. If neither of these are used, it is called a **generic signature**, and will be added to all service trees and the **unknown_service** tree. Our IPS engine has multiple service trees (http, smtp, pop3, dns, etc.) and one **unknown_service** tree.

If a packet is marked by a protocol dissector as some type of service, for example HTTP, the packet is only inspected by signatures in the HTTP service tree. The problem is, if a signature uses "**--dst_port 80**" without "**--service HTTP**", then it won't get matched. **Therefore, analysts must ensure that all signatures for known service types use the "--service" keyword.**

Since we cannot force external users to follow this rule, **custom signatures** of **unknown_service** will be added to all service trees.

8.1.2 Signature Matching

The IPS engine uses a two steps system to do signature matching:

- **pre-match** - a preliminary test to pick possible match candidates from a group of signatures. Only patterns are used in this test, including strings found in PCRE if there are any. For example, if the PCRE is `/ABCD\s*\w+\d{2}/`, the engine will extract “ABCD” and use it in the pre match phase. The seq, ack, tag, data_size, byte_test, byte_jump keywords are not used. All patterns except the following types are used in this step.
 - Complete PCRE
 - Patterns with “!”...”
 - Patterns that are too short, like one byte patterns.
 - Multi-search
- **full match** - a full and accurate match of these candidates. All conditions are inspected in the following order, regardless of their positions in the signature.
 - Service/port/protocol options.
 - Tag test options.
 - Non-pattern options including seq, data_size, data_at, etc.
 - Pattern options such as pattern, PCRE, byte_test and byte_jump.
 - Tag set/clear options.
 - Rate/track options.

Currently, signature matching consumes 70% of the total IPS performance. Step 1 (pre-match) accounts for two thirds of the signature matching time, and step 2 (full match) accounts for one third.

If a signature is picked as a candidate in step 1, it is hit once. If a signature contains no pattern in it, it is always a hit in step 1 so it will be included in step 2.

We can improve signature matching performance in two areas:

- **Reduce the pre-match time:**
 - Try to add more patterns with appropriate lengths.
 - Try to increase the pattern length so that it contains more than one byte.
 - The goal is to increase the number of candidates filtered out by the pre-match.
- **Reduce the full match time:**
 - Try to add non-pattern options if possible. For example, if the signature looks for a very long field, it can use data_size to check the packet size first.
 - Try to make the first pattern long and uncommon.
 - Try to give location limits for all patterns including PCRE, especially short patterns.
 - Try to keep PCRE patterns as simple as possible. If other patterns in the signature are easily matched, do not put them in the PCRE pattern.
 - One-byte patterns should not be used as the first pattern for any signature. However, they can be used if they are not the first pattern.

8.1.3 Application Control signatures selection process

The engine will start from the first condition, going down the list in descending order - higher priority at the top.

1. Signatures configured first have higher priority over the next ones.

E.g.

```
config application list
edit "default-5-6"
  set other-application-log enable
  config entries
    edit 1
      set application 1229 4427 8296 8960 16435 16714 16996
    next
    edit 2
      set application 7705 88777 88778 88779 88780
      set action pass
    next
    edit 3
      set category 6
    next
  end
next
End
```

Signatures in entries “edit 1” have higher priority over signatures in “edit 2” and “edit 3”.

2. Custom signatures have higher priority over DAC signatures.

3. DAC signatures have higher priority over regular signatures.

4. Higher weight app picked over lower ones.

5. Non hidden rules > hidden rules.

6. Rules with more patterns selected over those with less patterns.

7. Rules with higher attack IDs selected over smaller IDs. (Newer rules chosen first).

*******Extra note: Industrial signatures and DAC signatures trigger on a per-packet basis - the signatures can trigger multiple times in a session.**

E.g.

```
00000000 05 01 19 00 63 c7 c8 de 00 95 00 00 00 01 00 01 ....c... .....
```

```
00000010 00 01 00 00 00 00 00 00 01 00 00 00 02 00 00 00 ..... .....
```

```

00000020 40 59 00 00 00 00 00 00 40 59 00 00 00 00 00 00 @Y..... @Y.....
00000030 40 59 00 00 00 00 00 00 00 00 00 00 00 00 00 00 @Y..... .....
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 ..... .....
00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
00000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
00000090 00 00 00 00 00 .....
00000095 05 01 19 00 63 c7 c8 de 00 95 00 00 00 01 00 01 ....c... .....
000000A5 00 01 00 00 00 00 00 00 02 00 00 00 02 00 00 00 ..... .....
000000B5 40 59 00 00 00 00 00 00 40 59 00 00 00 00 00 00 @Y..... @Y.....
000000C5 40 59 00 00 00 00 00 00 00 00 00 00 00 00 00 @Y..... .....
000000D5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 ..... .....
000000E5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
000000F5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
00000105 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
00000115 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
00000125 00 00 00 00 00 .....
0000012A 05 01 19 00 63 c7 c8 de 00 95 00 00 00 01 00 01 ....c... .....
0000013A 00 01 00 00 00 00 00 00 03 00 00 00 02 00 00 00 ..... .....
0000014A 40 59 00 00 00 00 00 00 40 59 00 00 00 00 00 00 @Y..... @Y.....
0000015A 40 59 00 00 00 00 00 00 00 00 00 00 00 00 00 @Y..... .....
0000016A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 ..... .....
0000017A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
0000018A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
0000019A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
000001AA 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....
000001BA 00 00 00 00 00 .....

```

The above is 3 packets in the same session.

Scenario 1: --app_cat 15;

```

F-SBID( --vuln_id 44993; --attack_id 63927; --name "DIS"; --group SCADA; --protocol udp; --
default_action pass; --revision 12277; --app_cat 15; --vendor 0; --technology 2; --pop 1; --risk 2;
--language "N/A"; --weight 10; --require_ssl_di "No"; --severity info; --status disable; --dst_port
3000; --flow bi_direction; --byte_test 2,=$PKT_SIZE,8; --scan-range 2k,none; --date
20171207; )

```

The logs would show:

```

[attack] v_id=44993 a_id=63927 src=175.100.189.174:51000 dest=60.217.235.139:3000
SCADA: DIS (#1 in pkt 2)

```

Scenario 2: --app_cat 26;

```
F-SBID( --vuln_id 44993; --attack_id 63927; --name "DIS"; --group SCADA; --protocol udp; --
default_action pass; --revision 12277; --app_cat 26; --vendor 0; --technology 2; --pop 1; --risk 2;
--language "N/A"; --weight 10; --require_ssl_di "No"; --severity info; --status disable; --dst_port
3000; --flow bi_direction; --byte_test 2,=,$PKT_SIZE,8; --scan-range 2k,none; --date
20171207; )
```

The logs would show

```
[attack] v_id=44993 a_id=63927 src=175.100.189.174:51000 dest=60.217.235.139:3000
SCADA: DIS (#1 in pkt 2)
[attack] v_id=44993 a_id=63927 src=175.100.189.174:51000 dest=60.217.235.139:3000
SCADA: DIS (#1 in pkt 3)
[attack] v_id=44993 a_id=63927 src=175.100.189.174:51000 dest=60.217.235.139:3000
SCADA: DIS (#1 in pkt 4)
```

Scenario 3: --deep_ctrl set,\$R2-="870";

```
F-SBID( --vuln_id 44993; --attack_id 63927; --name "DIS"; --group SCADA; --protocol udp; --
default_action pass; --revision 12277; --app_cat 15; --vendor 0; --technology 2; --pop 1; --risk 2;
--language "N/A"; --weight 10; --require_ssl_di "No"; --severity info; --status disable; --deep_ctrl
set,$R2-="870"; --dst_port 3000; --flow bi_direction; --byte_test 2,=,$PKT_SIZE,8; --scan-range
2k,none; --date 20171207; )
```

The logs would show

```
[attack] v_id=44993 a_id=63927 src=175.100.189.174:51000 dest=60.217.235.139:3000
SCADA: DIS (#1 in pkt 2)
[attack] v_id=44993 a_id=63927 src=175.100.189.174:51000 dest=60.217.235.139:3000
SCADA: DIS (#1 in pkt 3)
[attack] v_id=44993 a_id=63927 src=175.100.189.174:51000 dest=60.217.235.139:3000
SCADA: DIS (#1 in pkt 4)
```

8.1.4 Ignore_content

In IPS Engine 3.402 and above, a new feature to adaptively skip pattern matching for certain content was added. The feature is as follow:

1. In signature loading time, scan all the rules under one app ID for the following options.
--context *FILE*
--context *BODY*
--context *HEADER*
--context *PACKET* (default for --pattern and --pcre options)
2. A different bit flag is used for each context type. If all bits are clear, the signatures do not care about the content. Otherwise:

3. If the *FILE/BODY* bit is clear, the content decoder can be skipped, and no *BODY/FILE* buffer is needed for this signature.
4. The engine will check the children's signature content scanning flag and propagate the flag to the parent.

The information can be identified in the debug log as following:

```
[8633/0]appctrl_process_match: set ignore_content to true by 6664  
Youtube.Channel.Specific_Custom (features=0004, buffer_usage=0000,0000)
```

E.g. (I simplified the signatures by removing the other syntax.)

Scenario 1:

```
F-SBID( --vuln_id 10000; --attack_id 10000; --name "Facebook"; --protocol tcp; --service HTTP;  
--flow from_client; --pattern "facebook.com"; --context host; )
```

When the engine scans the first HTTP packet, since there is no signature by the children of Facebook or Facebook itself that checks one of the context mentioned in 1., the engine will set the `ignore_content` flag to true.

Scenario 2:

```
F-SBID( --vuln_id 10000; --attack_id 10000; --name "Facebook"; --protocol tcp; --service HTTP;  
--flow from_client; --pattern "facebook.com"; --context host; )  
F-SBID( --vuln_id 10000; --attack_id 10001; --name "Facebook"; --protocol tcp; --service HTTP;  
--flow from_server; --pattern "facebook"; --context body; )
```

In this case, the engine will not set the `ignore_content` flag because `attack_id 10001` checks the body context in the server HTTP response.

Scenario 3:

```
F-SBID( --vuln_id 10000; --attack_id 10000; --name "Facebook"; --protocol tcp; --service HTTP;  
--flow from_client; --pattern "facebook.com"; --context host; )  
F-SBID( --vuln_id 10001; --attack_id 10001; --name "Facebook_Child"; --depend-on 10000; --  
protocol tcp; --service HTTP; --flow from_server; --pattern "facebook"; --context body; --scan-  
range 0,all; )
```

In this case, the engine will not set the `ignore_content` flag too because the engine would scan the children of Facebook and check if it can ignore content scanning for the children signature. Since `Facebook_Child` checks for the body context in the server's HTTP response, the `ignore_content` flag will not be set. Since `Facebook_Child` is a child of Facebook, the IPS Engine will not set the `ignore_content` flag when Facebook triggers.

8.1.5 Custom Application Control signatures

In IPS Engine 3.432 and above, custom signatures are no longer filtered out based on the

dependency information like the pre-defined signatures. In addition to this, if a custom signature is triggered, the session will not be offloaded like sessions triggered by pre-defined signatures that have no children or parent.

Custom signatures will not have the ignore_content flag set to true (in future IPS Engine - refer to Mantis ID 475749).

*******Refresher:** Signatures that do not have any children or parent that trigger on a session will cause the session to be offloaded immediately.

8.1.6 Industrial Application Control signatures

Signatures with “app_cat 26”, 26 for industrial, are treated like IPS signatures from the filtering point of view. The dependency tree is still used to filter out other Application Control signatures and to set the scan-range.

Since Industrial sessions can usually trigger the DAC signatures on the first packet, the best practice when writing Industrial signatures is to add a skip-after 0 in the parent signature. If a DAC signature triggers on the first packet, it may be able to modify the scan-range from the base value to the newer value - could be 10M, could be 0. Therefore, the parent Industrial signature should be generic enough for the skip-after to take effect when the DAC signatures trigger on the first packet.

8.1.7 Application Whitelisting

In IPS Engine 3.4, the engine uses the syntax --depend-on and --parent together to decide if a session would be deferred upon triggering the **first signature** in a session. (If a parent signature trigger after some other signature has triggered, the deferral works as usual.)

Key point: **The engine does not support sub-feature whitelisting. The engine identifies a sub-feature signature with the --parent syntax.** In other words, signatures of a parent, for e.g. Facebook_Chat of Facebook if set to Allow/Monitor with Facebook set to Block will not work. When the engine sees the Facebook signature trigger on one packet, it will drop the session immediately because Facebook_Chat has Facebook as its parent.

If, hypothetically, we want the engine to defer the detection after triggering Facebook, our Facebook_Chat signature needs to remove Facebook as its --parent while keeping --depend-on as Facebook. This will remove the parent-children relationship between Facebook and Facebook_Chat, allowing the engine to defer.

*******Note:** In FortiOS 5.0 which uses IPS Engine 2.xxx, application whitelisting is not supported. Signatures configured in a higher entry will have higher priority.

8.1.8 Whitelisting Logic

The IPS Engine whitelisting logic is as follow: (starting in IPS Engine in Mantis 474479)

Deferral logic:

1. If a signature that triggers have the action “pass”, it will not defer. (Deferral only works if the initial signature is set to block)
2. If `total_stream_size > scan_range + prolog_size` (size of SSL handshake), the IPS engine will not defer.
3. If `total_stream_size < scan_range`, the IPS engine will defer.
4. If a session is SSL, the IPS engine will defer during the handshake procedure.
5. If a session has scan-range 0, the IPS engine will not defer.

Signature selection:

When we already have a rule deferred (A) and consider *deferring* another rule (B):

- * If weight of the B is higher, defer it and replace A
- * Otherwise ignore and keep A deferred

When consider *triggering* new match (B):

- * If there's no deferral detection, trigger right away
- * If B has action pass, configured explicitly (doesn't come from other-app setting), and has a dependency on deferred rule (A), and configured in a higher entry, then trigger B and allow the traffic
- * Check if B itself should be deferred (see above)
- * If B does NOT have a dependency on A or its weight is higher than A's, trigger A and block the traffic
(here the weight check is needed to avoid SSL - HTTPS - SSL detections)

8.1.9 UDP and ICMP Sessions

In addition to TCP, the Fortigate also maintains sessions for UDP and ICMP.

- The session is established when the first packet is received, and it is closed by timeout (IPS:60 seconds, Firewall: 300 seconds).
- **"seq"** can be used. It accumulates the packet sizes from one side.
- **"drop_session"** can be used on UDP and ICMP.
- **"flow"** can be applied to UDP sessions.

Some engine dissectors reassemble TCP packets before signature matching. So **"seq"**, **"data_at"** and **"data_size"** could produce different results from a reassembled packet than from the original packet. Therefore, It is not recommended to use **"data_at"** for **TCP sessions** for a TCP session because a TCP session is regarded as a stream.

Note: UDP sessions are identified on the “per packet” basis as opposed to the “per session” basis for TCP sessions.

The key point with “per packet” detection is if a UDP session has one of its packets identified by a signature that is set to Block, subsequent packets after the dropped packet will not be dropped unless those packets match the signature again.

In contrast, in a TCP session, once a packet triggers a signature that is set to Block, the packet and the entire session will be set to drop all subsequent packets.

E.g.

In a VoIP session, the usual order of procedure is a STUN bind request/response to negotiate the protocol for the session, followed by RTP packets.

Scenario 1:

STUN - pass

RTP.Video - pass

RTP.Audio - block

Scenario 2:

STUN - pass

RTP.Audio - block

RTP.Video - pass

In Scenario 1 and 2, the blocked packets will not affect subsequent RTP.Video packets that are sent. This is because of the “per packet” detection logic of the IPS Engine.

8.1.10 Quarantining Attackers

In **FortiOS 3.0 MR6**, IPS sensors were introduced to help with IPS configuration. In each IPS sensor, users can add filters to select predefined signatures that they need. One of the filter options is **Target**, which means attack target, where the target is either the client or the server. The target is automatically identified by way of the **flow keyword** in the signature. If the flow is from_client , the target is the server; if the value is from_server , the target is the client.

In the **FortiOS 4.0** IPS sensors setup, users can select the **Quarantine Attackers** option in IPS Filter or IPS Override. When an attack occurs, the FortiGate will then automatically block subsequent packets from the attacker for a specified period.

The quarantine can be based on:

- Attacker's Source IP.
- Attacker's Source IP and Target's Destination IP.
- Attacker's Incoming Interface.

The attacker and the target are automatically identified by the source and destination IP of the packet that triggers the signature.

The above method of target and attacker identification works when the signature detects a packet from the attacker to the victim. But if the signature detects instead a packet from a victim

to an attacker, then the target and the attacker will be identified wrongly. The signature will then be wrongly filtered or missed in the IPS filter, and the target will be quarantined as the attacker. For example, the following signature detects FTP brute force login attacks against an FTP server by inspecting the FTP server's response.

```
F-SBID( --name FTP.Brute.Force.Login; --protocol tcp; --service FTP; --flow
from_server; --pattern "530 User "; --within 9,packet; --pattern "cannot login.[0d 0a]"; --
within 50; --rate 100,60; --track dst_ip; )
```

The target it protects is shown as the client on the GUI. Hence, quarantining the attacker blocks the FTP server, which is the victim of the attack. This is the opposite of what we want to achieve in this case. To correctly identify the target and the attacker in this situation, the flow keyword introduced the parameter **reversed** in Engine 1.046 . So the correct signature should be

```
F-SBID( --name FTP.Brute.Force.Login; --protocol tcp; --service FTP; --flow
from_server, reversed; --pattern "530 User "; --within 9,packet; --pattern "cannot
login.[0d 0a]"; --within 50; --rate 100,60; --track dst_ip; )
```

Therefore, for signatures detecting packets from victim to attacker, their flow keyword needs to contain the **reversed** modifier.

8.1.11 Traps

Sometimes IPS engines or other systems/tools may behave not as expected or do something tricky. It will be wise to know these traps before getting caught.

1. Null byte is dropped in request URI silently

The null byte in request URI as shown in following image will be dropped by engine after 2.167 silently.

Source	Destination	Protocol	Length	Info
192.168.1.100	192.168.1.101	TCP	62	xiostatus > 6014 [SYN] Seq=0 w
192.168.1.101	192.168.1.100	TCP	62	6014 > xiostatus [SYN, ACK] Se
192.168.1.100	192.168.1.101	TCP	54	xiostatus > 6014 [ACK] Seq=1 A
192.168.1.100	192.168.1.101	HTTP	208	GET \000\005\0031A HTTP/1.0
192.168.1.100	192.168.1.101	TCP	54	xiostatus > 6014 [FIN, ACK] Se

Analyst will find following pattern did not work:

```
--pattern "|00 05 03 31 41|"; --context uri; --within 10, context;
```

And there is no clue in engine debug information. In fact the debug information is rather misleading:

```
[17354/0]ips_http_handler_uri: enter
[17354/0]ips_http_handler_uri: confirm HTTP
uri[0](17)=GET 1A HTTP/1.0
```

The fact is the engine dropped null character only. Following pattern will work for you:

```
--pattern "|05 03 31 41|"; --context uri; --within 10, context;
```

Engine team is considering correcting this behavior in the future. But no fix is available at the time of writing.

2. Missing character '&' in the crafted URL for HTTP POST

As we know, the engine will insert the HTTP body into the URI line for some HTTP POST request. Engine will simply combine the decoded URI with POST data and does not put a '&' sign between them. Using HTTP request shown in following image as an example.



```
POST /webGoat/attack?Screen=1104&menu=1200 HTTP/1.1\r\n
[truncated] Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash,
Referer: http://172.22.5.152:8006/webGoat/attack?Screen=1104&menu=1200\r\n
Accept-Language: zh-cn\r\n
Content-Type: application/x-www-form-urlencoded\r\n
UA-CPU: x86\r\n
Accept-Encoding: gzip, deflate\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; windows NT 5.1; .NET CLR 2.0.50727; CIBA; .NET CLR 3.0.4
Host: 172.22.5.152:8006\r\n
Content-Length: 150\r\n
Connection: Keep-Alive\r\n
Cache-Control: no-cache\r\n
Cookie: JSESSIONID=E5B43F5ED0E5058986A0C913871A6286\r\n
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=\\r\\n
\r\n
[Full request URI: http://172.22.5.152:8006/webGoat/attack?Screen=1104&menu=1200]
Line-based text data: application/x-www-form-urlencoded
account_number=101+AND+%28ascii%28+substr%28%28SELECT+first_name+FROM+user_data+WHERE+userid%3D15613%29
```

The crafted URI will be:

```
uri[1](173)=POST /WebGoat/attack?Screen=1104&menu=1200account_number=101 AND
(ascii( substr((SELECT first_name FROM user_data WHERE userid=15613) , 1 , 1) ) < 77 );
&SUBMIT=Go! HTTP/1.1
```

Avoid checking character '&' in this case.

3. 200K limitation

The IPS engine will ignore non-HTTP sessions after a set amount of (200k bytes by default) traffic has passed.

Only checking the first 200k bytes increases throughput dramatically. Unfortunately, it is possible the pattern we want to detect appears only after 200k bytes in a session. In this case, our signature will be bypassed. This feature has been leveraged to bypass our signature for PDF exploit in a public demo. As a compromise solution, we added a signature with `--skip-after 0`; for all HTTP sessions. On top of that, IPS engine will also scan the entire session if an exploitable file such as FLASH, PDF, JS, VBS, MS Office, PE, etc, is detected.

For more information regarding this, please refer to the document “Intelligent Mode Summary”

4. P2P/IM Decoders

You may need to test following decoder signatures when handling AppCtrl related issues:

```
AIM 1
ICQ 2
BitTorrent 6
Gnutella 8
Kazaa 9
Skype 10
Winny 1
```

The number following the signature name is its small vid.

Remember to use option '-d im' or '-d p2p' when test these decoders with iscan:

For 1.0 engine

```
./iscan -r nids-400-4.260.txt -f bt.pcap -d p2p
```

For 2.0 engine

```
./iscan -r nids-500-4.260.bin bt.pcap -d p2p
```

Following AppCtrl signatures are merged with these decoder signatures in 2.0 engine to reduce confusion:

```
Skype_Communication (30197) -> Skype
BitTorrent_HTTP.Track (26038) -> BitTorrent
Edonkey_Handshake (29100) -> Edonkey
Gnutella_Download (30620) -> Gnutella
MSN.Messenger_Communication (30266) -> MSN.Messenger
ICQ_Communication (30588) -> ICQ
QQ_Sign.In (30436) -> QQ
```

To test these signatures with 2.0 version iscan, you have to use option '-d im' or '-d p2p' too!

```
./iscan -r nids-500-4.364.bin icq8-login.pcap -d im
```

IPS Offline Scanner v2.1.158

Load signature 04.364

Startup time: 558.841 Mcycles

*[11464/0]ips_handle_engine_rule: engine rule id **30588**, attack id **39863**, ICQ_Communication
pkt 6*

[11464/0]ips_alert_im_ex: IM.aim, 172.22.5.223:12480 -> 94.100.186.23:443 (pkt 6)

[11464/0]ips_alert_im_ex: icq session

==> /home/bingliu/fortigate/test2/icq8-login.pcap (66.720 Mcycles)

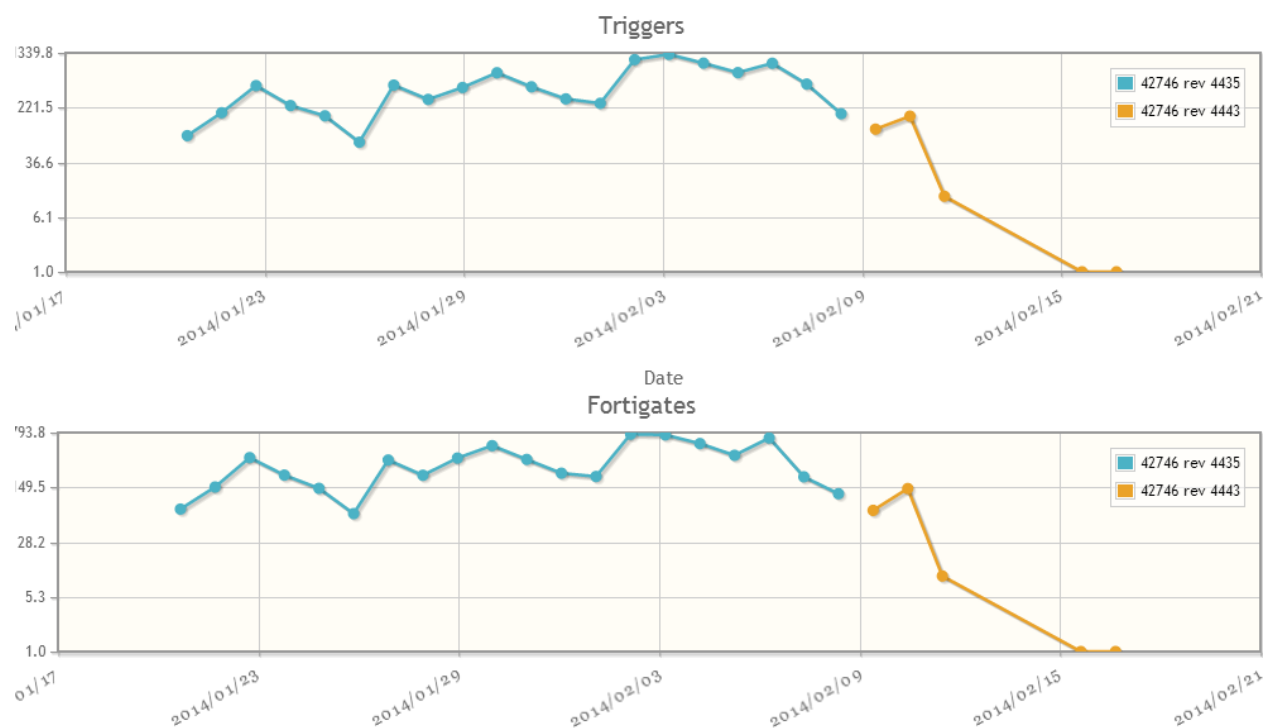
5. Signature revisions

IPS signatures need to go through the beta process. During this process analysts may improve the signature and ISTATE will record these changes in signature revision history. If the signature passed the beta process without any change, ISTATE will release the signature with a **new** revision number and the old revision signature will disappear from the signature history! Following is an image of signature history of aid 42746.

Vulnerabilities > Signatures > Rule Files > Descriptions > PCAPs > Documents > Lists > Icons > Account >		
Analyst	Modified Date	Rule
nanxiao	Feb. 7, 2014, 12:05 p.m.	F-SBID(--vuln_id 37853; --attack_id 42746; --name "Neutrino.Pack.Exploit.Kit"; --group backdoor; --protocol tcp; --default_action pass; --revision 4443; --severity critical; --default#VML"; --context body; --pattern "document.createStyleSheet().cssText"; --context body; --distance -256; --within 512; --pattern "behavior."; --context body; --within within 128; --date 20140213;)

Date	User	
Feb. 7, 2014, 12:05 p.m.	ips_qa_scripts	Released from Beta
Jan. 30, 2014, 4:39 p.m.	palanceng	Unset Evaluate
Jan. 30, 2014, 4:39 p.m.	palanceng	Set Evaluate
Jan. 27, 2014, 8:46 a.m.	ips_qa_scripts	Unset Evaluate
Jan. 20, 2014, 3:25 p.m.	palanceng	Modified Status from New to Releasable
Jan. 20, 2014, 3:22 p.m.	palanceng	Set Evaluate
Jan. 20, 2014, 3:21 p.m.	palanceng	Added new Signature

Signature aid 42746 was released as beta in package 4.435. You can't find it in the signature history after it was officially released in package 4.443. ISTEAM developers think this is reasonable because revision 4.435 is the same as 4.443 and the date of "Added new Signature" is helpful to recover the history. But this history is a little confusing to analysts. To make things worse, The FDS system (refer 9.2.1 for details) treated them as different revisions and discarded the alerts of 4437 when 4450 was released. Following is an image of the FDS Statistic of aid 42476. As you can see there is a sharp drop after revision 4.443 was released. But this is simply because Many FGTs did not update to revision 4.450 yet and FDS discarded the alerts of revision 4437.



6. File Decompression

As of 3.0 engine, support for decompressing some files has been added into the engine. The engine currently (as of v3.030) supports decompression of files with the following file extensions:

.swf	CWS format Flash file
.xap	Sliverlight
.jar	Java
.xlsx	OOXML
.docx	OOXML
.pptx	OOXML

Update.

Engine added support of decompressing ZWS format flash file in 3.49.

Engine extend the support of decompressing OOXML files to following file extensions in 3.60:

- .docm
- .dotx
- .dotm
- .pptm
- .potx
- .potm
- .ppam
- .ppsm
- .sldx
- .sldm
- .thmx
- .xlsm
- .xltx
- .xltm
- .xlsb
- .xlam

Engine added support for decoding PDF streams in 3.62.

Analysts should note that the engine only decompresses the file if it sees the above file extension. This is due to concerns regarding the engine's performance. Hence, if there is a Java file that has its file extension as something else other than jar, the engine will not decompress it. Analysts should take note when adding signatures regarding these few files as some attack tools use random file extension for those files.

7. Malformed HTTP Chunked-Body

HTTP messages that use chunked transfer encoding include "Transfer-Encoding: chunked" as one of their headers, while the message body has the following format:

```
Chunked-Body = *chunk
                last-chunk
                trailer
                CRLF
chunk = chunk-size [ chunk-extension ] CRLF
```

```

        chunk-data CRLF
chunk-size = 1*HEX
last-chunk = 1*("0") [ chunk-extension ] CRLF
chunk-extension= *( ";" chunk-ext-name [ "=" chunk-ext-val ] )
chunk-ext-name = token
chunk-ext-val = token | quoted-string
chunk-data = chunk-size(OCTET)
trailer = *(entity-header CRLF)

```

The chunk encoding support was introduced in IPS engine 1.63. When the engine encounters an error during parsing a chunked-body, it will simply discard it. So, it is impossible for analysts to detect a malicious chunked-body in many cases. To solve this problem engine 3.31 decided to keep the malformed chunked-body in the http body buffer. This was later found to be a bad decision when signatures checking malformed chunked-body matched normal http message body unexpected. We need a better place to store malformed chunked-body to avoid FPs.

Engine 3.66 decided to append a malformed chunked-body in the http header buffer. For example given following request:

```

0000 47 45 54 20 2F 50 6C 65 72 53 5A 74 53 64 76 6F  GET /PlerSZtSdvo
0010 4A 64 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73  Jd HTTP/1.1..Hos
0020 74 3A 20 4C 55 31 69 68 43 42 76 6A 72 4E 69 77  t: LU1ihCBvjrNiw
0030 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F  ..User-Agent: Mo
0040 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 69 6E 64 6F  zilla/5.0 (Windo
0050 77 73 20 4E 54 20 36 2E 31 3B 20 57 69 6E 36 34  ws NT 6.1; Win64
0060 3B 20 78 36 34 3B 20 72 76 3A 32 2E 30 2E 31 29  ; x64; rv:2.0.1)
0070 20 47 65 63 6B 6F 2F 32 30 31 30 30 31 30 31 20  Gecko/20100101
0080 46 69 72 65 66 6F 78 2F 34 2E 30 2E 31 0D 0A 41  Firefox/4.0.1..A
0090 63 63 65 70 74 3A 20 2A 2F 2A 0D 0A 54 72 61 6E  ccept: /*..Tran
00A0 73 66 65 72 2D 45 6E 63 6F 64 69 6E 67 3A 20 63  sfer-Encoding: c
00B0 68 75 6E 6B 65 64 0D 0A 0D 0A 48 45 7A 42 6F 7A  hunked....HEzBoz
00C0 46 70 57 6C 54 4F 5A 49 51 71 44 52 72 4F 78 6F  FpWITOZIqQDRrOxo
00D0 41 47 51 69 4E 76 6C 69 6B 5A 71 47 41 73 67 6F  AGQiNvlikZqGAsgo

```

Engine will append the malformed chunked-body "HEzBozFpWITOZIqQ...." to header buffer. You can find this info in the iscan debug output(available in engine 3.68):

```

[32403/0]decode_data_chunk: chunk: 0
[32403/0]append_chunk_length_to_header: append chunk length to header(len=1274):
[32403/0]append_chunk_length_to_header: ips_pkt_id: 8
0000 48 45 7A 42 6F 7A 46 70 57 6C 54 4F 5A 49 51 71  HEzBozFpWITOZIqQ
0010 44 52 72 4F 78 6F 41 47 51 69 4E 76 6C 69 6B 5A  DRrOxoAGQiNvlikZ
0020 71 47 41 73 67 6F 69 71 65 69 66 70 58 7A 48 6A  qGAsgoiqeifpXzHj
0030 75 55 52 50 6E 59 54 64 79 72 49 77 48 49 4C 44  uURPnYTdyrlwHILD

```

Analyst can locate the malformed chunked-body in header buffer with pattern "[0d 0a 0d 0a]".

Please keep in mind the engine appends the chunked-body only when it encounters an error during parsing or the chunk-size plus chunk-extension in a chunk is longer than 24 bytes. Decoder HTTP.Chunk.Length.Invalid will be triggered in the former case.

It is possible the chunked-body spread in multiple packets. Engine ONLY appends the chunked-body part in the current packet when it encounters an error. For example following POST request spread in two packets:

```
PACKET id:4 len:155 vf:0 fw:0 view:31 derived:0 log:(traffic:0 pre:0 post:0)
  imp2p:0xff proxy:0x0 features:0x1 flowutm:1
  172.16.8.164:44419 -> 172.16.8.230:80 protocol:6
  IP length:141b, header:20b, ttl:64, tos:0, id:10621
  TCP payload:89b, header:32b
  TCP seq:2192889060, ack:1924685001, win:115, flags:***AP***
[32539/0]ips_run_decode: ips_pkt_id: 4
0000 00 50 56 B4 0E 7F 00 50 56 B4 03 53 08 00 45 00 .PV....PV..S..E.
0010 00 8D 29 7D 40 00 40 06 A7 43 AC 10 08 A4 AC 10 ..})@. @..C.....
0020 08 E6 AD 83 00 50 82 B4 D4 E4 72 B8 5C C9 80 18 .....P....r.\..
0030 00 73 37 72 00 00 01 01 08 0A 54 A1 FD 98 1D 3E .s7r.....T....>
0040 3E 3D 50 4F 53 54 20 2F 20 48 54 54 50 2F 31 2E >=POST / HTTP/1.
0050 31 0D 0A 48 6F 73 74 3A 20 31 37 32 2E 31 36 2E 1..Host: 172.16.
0060 38 2E 32 33 30 0D 0A 54 72 61 6E 73 66 65 72 2D 8.230..Transfer-
0070 45 6E 63 6F 64 69 6E 67 3A 20 63 68 75 6E 6B 65 Encoding: chunke
0080 64 0D 0A 0D 0A 41 3B 61 3D 62 0D 0A 68 65 6C 6C d....A;a=b..hell
0090 6F 77 6F 72 6C 64 0D 0A 30 0D 0A oworld..0..

PACKET id:6 len:1514 vf:0 fw:0 view:31 derived:0 log:(traffic:0 pre:0 post:0)
  imp2p:0xff proxy:0x0 features:0x1 flowutm:1
  172.16.8.164:44419 -> 172.16.8.230:80 protocol:6
  IP length:1500b, header:20b, ttl:64, tos:0, id:10622
  TCP payload:1448b, header:32b
  TCP seq:2192889149, ack:1924685001, win:115, flags:***A****
[32539/0]ips_run_decode: ips_pkt_id: 6
0000 00 50 56 B4 0E 7F 00 50 56 B4 03 53 08 00 45 00 .PV....PV..S..E.
0010 05 DC 29 7E 40 00 40 06 A1 F3 AC 10 08 A4 AC 10 ..)~@. @.....
0020 08 E6 AD 83 00 50 82 B4 D5 3D 72 B8 5C C9 80 10 .....P...=r.\..
0030 00 73 9B 11 00 00 01 01 08 0A 54 A1 FD 98 1D 3E .s.....T....>
0040 3E 3D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D >=.....
0050 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D .....
```

As you will see in the debug output, engine only append the chunked-body part in packet 6 to the header buffer.

```

[32539/0]decode_data_chunk: chunk: 0
[32539/0]append_chunk_length_to_header: append chunk length to header(len=1448):
[32539/0]append_chunk_length_to_header: ips_pkt_id: 6
0000 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D .....
0010 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D .....
0020 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D .....
0030 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D .....

```

You will not be able to detect the string "A;a=b" in this case.

Starting from 3.6.508, the ips engine appends 'IPS-NEG-CHUNK-SIZE:' to the header buffer when a negative chunk length is met. You should be able to see something like below in the debug log:

```

[26627/0]append_chunk_length_to_header: append chunk length to header(len=19):
[26627/0]append_chunk_length_to_header: ips_pkt_id: 6
0000 49 50 53 2D 4E 45 47 2D 43 48 55 4E 4B 2D 53 49 IPS-NEG-CHUNK-SI
0010 5A 45 3A ZE:

[26627/0]append_chunk_length_to_header: append chunk length to header(len=15):
[26627/0]append_chunk_length_to_header: ips_pkt_id: 6
0000 2D 33 66 66 66 66 66 66 63 30 30 30 0D 0A -3ffffffc000..

```

So, to detect a negative chunk size, you can simply search the pattern "IPS-NEG-CHUNK-SIZE:" in the context header.

8. Byte_test 32-bit integer overflow

The engine does not support 64-bit variables yet so it may not work as expected given an integer value larger than 0xffffffff. The following issues were found so far:

1. **IPSC** integer overflow when parsing **value** in keyword **byte_test**

```

--pattern "Content-Length: "; --context header; --byte_test
*,>,922337203685477580,0,string,relative;

```

Before the fix, this signature was compiled as the following one which caused a lot of FPs:

```

--pattern "Content-Length: "; --context header; --byte_test
*,>,0xffffffff,0,string,relative;

```

After the fix in 3.39, **IPSC** will give you an error like below when trying to compile a **byte_test** with an overly large value:

Error:

Line 23313 offset 354 option "byte_test" value "",>,922337203685477580,0,string,relative"

--byte_test *,>,922337203685477580,0,string,rela

invalid number value

To work around this issue, you can use a **pcre** to check the big number like below:

--pattern "Content-Length: "; --context header; --byte_test *,>,0x7fffffff,0,string,relative; --pcre "/^Content-Length:\s+[1-9]d{17}/"; --context header; --distance -16;

But even this signature may not work due to the integer overflow issue during the string conversion. See below the details.

2. Engine integer overflow when handling converted string data

Signature

--dst_port 10514; --flow from_client; --pattern "|3c|"; --within 1,packet; --pattern !"|3e|"; --context packet; --within_abs 10; --byte_test *,>,0x7fffffff,0,string,relative;

Missed below traffic on 64-bit system:

3	0.000028	1.1.114.242	1.2.180.215	TCP	62	2008
6	0.002145	1.2.180.215	1.1.114.242	TCP	62	10514
7	0.043305	1.1.114.242	1.2.180.215	TCP	62	2008
8	0.102329	1.1.114.242	1.2.180.215	TCP	189	2008
9	0.146074	1.1.114.242	1.2.180.215	TCP	60	2008
10	0.146148	1.2.180.215	1.1.114.242	TCP	54	10514
11	0.146148	1.2.180.215	1.1.114.242	TCP	54	10514

Frame 8: 189 bytes on wire (1512 bits), 189 bytes captured (1512 bits)			
Ethernet II, Src: MS-NLB-PhysServer-26_c5:01:00:00 (02:1a:c5:01:00:00), Dst: MS-NLB-PhysServer-26_c5:02:00:00 (02:1a:c5:02:00:00)			
Internet Protocol Version 4, Src: 1.1.114.242 (1.1.114.242), Dst: 1.2.180.215 (1.2.180.215)			
Transmission Control Protocol, Src Port: 20088 (20088), Dst Port: 10514 (10514), Seq: 1, Ack: 1, Len: 135			
Data (135 bytes)			

0000	02 1a c5 02 00 00 02 1a c5 01 00 00 08 00 45 00E.
0010	00 af 83 48 00 00 ff 06 0e 34 01 01 72 f2 01 02	...H... .4..P...
0020	b4 d7 4e 78 29 12 b8 d2 28 fd 06 b7 5c a9 50 10	..Nx)... (...\.P.
0030	3f ff 02 ff 00 00 3c 35 35 30 31 38 31 37 32 33	?.....<5 50181723
0040	37 3e 20 41 70 72 20 32 37 20 31 30 3a 34 33 3a	7> Apr 2 7 10:43:
0050	36 20 43 47 68 6a 6f 47 74 6c 4f 47 20 75 6f 4d	6 CGhjoG tlog uoM
0060	43 5a 57 4d 54 50 53 4b 31 74 38 4d 65 58 35 4a	CZWMTPSK 1t8Mex5J
0070	30 30 65 50 63 4d 4e 77 6d 4f 67 4c 38 45 78 5a	00ePCmNw mOgL8ExZ
0080	4c 56 32 50 4f 6d 71 54 6f 77 52 4e 6b 54 39 79	LV2PomqT owRNkT9y
0090	59 75 7a 56 69 50 65 79 53 68 36 51 79 52 45 74	YuzviPey Sh6QyREt
00a0	7a 37 66 6c 4a 45 6a 75 48 58 67 68 62 64 4a 55	z7f1JEju HXghbdJU
00b0	68 79 37 78 55 38 62 6d 78 42 43 61 74	hy7xu8bm xBCat

As you can see, the converted number is 0x147ef1195 in hex which overflowed uint32. The 32-bit Engine will detect this overflow and replace it with 0xffffffff. Unfortunately, on a 64-bit system what is left is 0x47ef1195, leading to this FN.

This issue was fixed 3.072 and the 64-bit Engine will also replace it with 0xffffffff for comparison.

9. Negative search in URI

The IPS engine decodes and normalizes the original URI field, placing the results in three buffers: **origin**, **decoded** and **rmdir**. For example:

origin:

GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+ver

decoded:

GET /scripts/../../winnt/system32/cmd.exe?/c+ver

("\" is also converted to "/" in this phase.)

rmdir:

GET /winnt/system32/cmd.exe?/c+ver

The **decoded** and **rmdir** buffer will be **empty** when their contents are the same as the origin buffer.

The specific contents as parsed by the engine are visible in the debug log as:

uri[0](length)=contents of origin buffer

uri[1](length)=contents of decoded buffer

uri[2](length)=contents of rmdir buffer

They will not be logged if they are empty.

All three buffers are searched for the specified pattern. In the case of *negative search*, as of Mantis 493529, only the **origin** and **decoded** buffers are searched, and the verdict will be a match when the pattern appears in **None** of these two buffers.

In most cases, we should simply use 'context uri' for negative matches to prevent FP/FN issues, but analysts should be aware of how the URI may differ in the three buffers and whether they need to search in a specific one in some cases.

For example,

The following pattern:

`--pattern "jump="; --context uri; --pattern !"0a"; --context uri; --within_abs 1000;`

Can be bypassed if:

1. The 'jump' value contains "%0A" at the beginning
In this case, the jump value contains a '|0a|' at the beginning in the decoded buffer
2. The 'jump' value contains "|F2 AA BE 89|" at the beginning
|F2 AA BE 89| is the UTF-8 encoding for unicode AAF89, which is then stored as |0A AF 89| in the decoded buffer
3. The 'jump' value is percent-encoded

In this case, the jump value can be shorter than 1000 in the decoded buffer

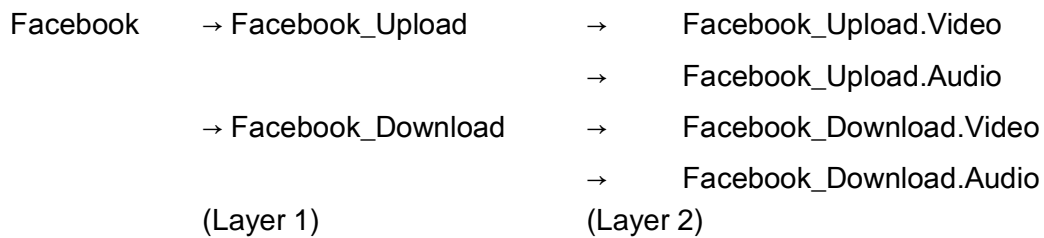
Currently, there is no solution for the first two issues. Analysts can choose to scan the origin buffer only (`--context uri,origin;`) if needed.

To fix the third issue, analysts need to choose a proper value for the `within_abs` option.

10. AppCtrl children signatures get filtered out due to the HAO implementation

The engine introduced the **Hierarchical App Organization (HAO)** feature in 3.286 to improve the performance. Unfortunately, this feature may bring detection loss in some cases. For signatures that have more than one layer of children, once a signature in layer 2 or above triggers, the engine will not be able to traverse the dependency list back to layer 1 and to another layer's children.

E.g.



If a session has triggered any of `Facebook_Upload.Video` or `Facebook_Upload.Audio`, the engine will not be able to trigger any of `Facebook_Download`'s signatures. On the other hand, if the engine only triggers `Facebook_Upload`, it can still trigger `Facebook_Download`.

11. JS normalization

The engine introduced the JS normalization feature in 3.043 which includes following processes:

1. Remove extra whitespace

`var a=`

⇒

`var a=`

2. Replace `'\r'`, `'\n'` and `'\t'` with the space character

`<script>`

var a=

=>

<script> var a=

3. unescape() method

eval(unescape('%66%75%63%74%69%6f%6e

=>

eval(unescape('function

4. fromCharCode() method

[(String.fromCharCode(0151, 0x65))]

=>

["(ie)"]

5. decodeURIComponent() method

s=decodeURIComponent("%41%41

=>

s="AA

6. String concatenation

"G" + "E" + "" + "T"

=>

GET

To reduce the performance impact, the engine enters these normalization processes only if:

- 1.The <script> tag appears within the first 2k of the file
2. The file size is larger than 50 bytes

These conditions may lead to detection loss in some cases. Analysts can write two sets of signatures to detect both the original and normalized content if needed.

The engine removed these conditions In build 3.506.

Following are some known limitations:

1. The engine failed to remove JS comments(mantis bug 0461586)
2. The engine will not normalize JS placed in external files. For example,

```
<script src="myscripts.js"></script>
```

The file myscripts.js will not be normalized even if it has file extension **.js**.

12. Inspects SSL encrypted packets

Only a few IPS and AppCtrl signatures detect the SSL encrypted packets. To optimize the SSL traffic scanning, the IPS engine decided to detach the IPS and AppCtrl features as soon as possible after the key exchange is done. In the build 4.340, the following logic was implemented:

1. Only the first encrypted packet will be scanned for AppCtrl
2. For IPS, a relatively larger scan range (**8k**) is used

This doesn't affect packet scanning for decrypted data sessions. The optimization is applied to the original SSL sessions.

Unfortunately, this optimization may cause some detection loss issues especially for AppCtrl signatures. For example, the following pattern wants to detect the first SSL application data packet:

```
--pattern "|17 03 03|"; --context packet;
```

As expected, it matches packet #12 as shown in the following image.

No.	Time	Source	Protocol	Destination	Length	Info
1	0.000000	192.168.1.110	TCP	205.250.85.73	66	42152 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.003770	205.250.85.73	TCP	192.168.1.110	66	443 → 42152 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_1
3	0.003841	192.168.1.110	TCP	205.250.85.73	54	42152 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
4	0.005471	192.168.1.110	TLSv1.2	205.250.85.73	226	Client Hello
5	0.008190	205.250.85.73	TCP	192.168.1.110	60	443 → 42152 [ACK] Seq=1 Ack=173 Win=30336 Len=0
6	0.012693	205.250.85.73	TLSv1.2	192.168.1.110	1514	Server Hello
7	0.012754	205.250.85.73	TCP	192.168.1.110	1514	443 → 42152 [ACK] Seq=1461 Ack=173 Win=30336 Len=1460 [TCP segmen
8	0.012781	192.168.1.110	TCP	205.250.85.73	54	42152 → 443 [ACK] Seq=173 Ack=2921 Win=65700 Len=0
9	0.015164	205.250.85.73	TLSv1.2	192.168.1.110	581	Certificate, Server Key Exchange, Server Hello Done
10	0.022361	192.168.1.110	TLSv1.2	205.250.85.73	180	Client Key Exchange, <u>Change Cipher Spec</u> , Encrypted Handshake Messa
11	0.024771	205.250.85.73	TLSv1.2	192.168.1.110	296	New Session Ticket, <u>Change Cipher Spec</u> , Encrypted Handshake Messa
12	0.042656	192.168.1.110	TLSv1.2	205.250.85.73	1262	<u>Application Data</u>
13	0.042798	192.168.1.110	TCP	205.250.85.73	1514	42152 → 443 [ACK] Seq=1507 Ack=3690 Win=64928 Len=1460 [TCP segmen

> Frame 12: 1262 bytes on wire (10096 bits), 1262 bytes captured (10096 bits)
 > Ethernet II, Src: Dell_d9:cf:bf (d4:be:d9:d9:cf:bf), Dst: Fortinet_0a:31:9e (90:6c:ac:0a:31:9e)
 > Internet Protocol Version 4, Src: 192.168.1.110, Dst: 205.250.85.73
 > Transmission Control Protocol, Src Port: 42152, Dst Port: 443, Seq: 299, Ack: 3690, Len: 1208
 √ Secure Sockets Layer
 √ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
 Content Type: Application Data (23)
 Version: TLS 1.2 (0x0303)
 Length: 1203
 Encrypted Application Data: 00000000000000001976b877dc88c4d994908fd2f5b27ed4a...

First encrypted packet

But if the Change Cipher Spec and the Encrypted Handshake Message records from the server are sent in separate packets(#18 and #19), as shown in the image below.

No.	Time	Source	Protocol	Destination	Length	Info
10	0.320531	178.62.195.71	TLSv1.2	192.168.106.3	460	Server Key Exchange
11	0.320761	178.62.195.71	TLSv1.2	192.168.106.3	63	Server Hello Done
12	0.322351	192.168.106.3	TCP	178.62.195.71	54	47878 → 443 [ACK] Seq=197 Ack=
13	0.322772	192.168.106.3	TCP	178.62.195.71	54	47878 → 443 [ACK] Seq=197 Ack=
14	0.377059	192.168.106.3	TLSv1.2	178.62.195.71	197	Client Key Exchange
15	0.377309	192.168.106.3	TLSv1.2	178.62.195.71	60	<u>Change Cipher Spec</u>
16	0.377494	192.168.106.3	TLSv1.2	178.62.195.71	99	Encrypted Handshake Message
17	0.531475	178.62.195.71	TCP	192.168.106.3	54	443 → 47878 [ACK] Seq=1497 Ack=
18	0.531692	178.62.195.71	TLSv1.2	192.168.106.3	60	<u>Change Cipher Spec</u>
19	0.531768	178.62.195.71	TLSv1.2	192.168.106.3	99	<u>Encrypted Handshake Message</u>
20	0.535003	192.168.106.3	TCP	178.62.195.71	54	47878 → 443 [ACK] Seq=391 Ack=
21	0.535306	192.168.106.3	TCP	178.62.195.71	54	47878 → 443 [ACK] Seq=391 Ack=
22	0.535494	192.168.106.3	TLSv1.2	178.62.195.71	562	<u>Application Data</u>
23	0.681771	178.62.195.71	TLSv1.2	192.168.106.3	542	<u>Application Data</u>

> Frame 22: 562 bytes on wire (4496 bits), 562 bytes captured (4496 bits)
 > Ethernet II, Src: SamsungE_1a:4c:bf (e8:50:8b:1a:4c:bf), Dst: 00:ff:c8:de:fc:9e (00:ff:c8:de:fc:9e)
 > Internet Protocol Version 4, Src: 192.168.106.3, Dst: 178.62.195.71
 > Transmission Control Protocol, Src Port: 47878, Dst Port: 443, Seq: 391, Ack: 1548, Len: 508
 √ Secure Sockets Layer
 > TLSv1.2 Record Layer: Application Data Protocol: http-over-tls

First encrypted packet

In this case, there is no chance for the pattern to match against the first Application Data packet(#22) because the AppCtrl feature was detached after packet #19 as indicated by the following information in the debug log:

PACKET id:19 len:99 vf:0 fw:0 view:31 derived:0 encap:0 log:(traffic:0 pre:0 post:0)

imp2p:0xff proxy:0x0 features:0x6 flowutm:1

...

[3509/0]evaluate_session_features: ignore_app_session: session_size=1937, prolog_size=1892, ignore_app_after_size=1

If needed, analysts can use the option skip-after to ask the engine to scan more SSL encrypted data for the AppCtrl signatures. Unfortunately, the engine will reset the scan range value when the SSL key exchange is not done(Mantis bug 501556). To work around this bug, the skip-after signature needs to catch the first encrypted packet.

13. Pattern is not found in the context packet

Generally, analysts should always use “context uri” for the patterns in the URL. The HTTP decoder of the IPS engine prepares the URI buffers carefully for signature matching purposes. For example, it inserts the HTTP body into these URI buffers for the below HTTP POST request:

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Protocol	Destination	Length	Info
981	-85.669928	192.168.106.3	HTTP	34.194.122.51	443	POST / HTTP/1.1 (application/x-www-form-urlencoded)
982	-85.589997	34.194.122.51	TCP	192.168.106.3	66	80 → 37006 [ACK] Seq=1 Ack=378 Win=19456 Len=0 TSval
983	-85.587922	34.194.122.51	HTTP	192.168.106.3	326	HTTP/1.1 200 OK (text/plain)
> Frame 981: 443 bytes on wire (3544 bits), 443 bytes captured (3544 bits)						
> Ethernet II, Src: LgElectr_82:a9:a9 (64:bc:0c:82:a9:a9), Dst: 00:ff:2f:92:ed:8b (00:ff:2f:92:ed:8b)						
> Internet Protocol Version 4, Src: 192.168.106.3, Dst: 34.194.122.51						
> Transmission Control Protocol, Src Port: 37006, Dst Port: 80, Seq: 1, Ack: 1, Len: 377						
> Hypertext Transfer Protocol						
> HTML Form URL Encoded: application/x-www-form-urlencoded						
<div>0000 00 ff 2f 92 ed 8b 64 bc 0c 82 a9 a9 08 00 45 00/.d.E 0010 01 ad 4a 55 40 00 40 06 27 55 c0 a8 6a 03 22 c2 ..JU@.@ 'U..j.." 0020 7a 33 90 8e 00 50 1e 8b d7 24 c6 4c 6c f7 80 18 z3...P...\$.Ll.. 0030 01 57 d6 04 00 00 01 01 08 0a 12 35 9d 8a 38 80 .W.....5..8.. 0040 09 d1 50 4f 53 54 20 2f 20 48 54 54 50 2f 31 2e ..POST / HTTP/1.. 0050 31 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 0d 1..User- Agent: . 0060 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 61 .Content -Type: a 0070 70 70 6c 69 63 61 74 69 6f 6e 2f 78 2d 77 77 77 pplicati on/x-www 0080 2d 66 6f 72 6d 2d 75 72 6c 65 6e 63 6f 64 65 64 -form-ur lencoded 0090 0d 0a 48 6f 73 74 3a 20 65 63 32 2d 33 34 2d 31 ..Host: ec2-34-1 00a0 39 34 2d 31 32 32 2d 35 31 2e 63 6f 6d 70 75 74 94-122-5 1.comput 00b0 65 2d 31 2e 61 6d 61 7a 6f 6e 61 77 73 2e 63 6f e-1.amaz onaws.co 00c0 6d 0d 0a 43 6f 6e 74 65 65 63 74 69 6f 6e 3a 20 4b m..Conne ction: K 00d0 65 65 70 2d 41 6c 69 76 65 0d 0a 41 63 63 65 70 eep-Aliv e..Accep 00e0 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 t-Encodi ng: gzip 00f0 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 ..Conten t-Length 0100 3a 20 31 37 38 0d 0a 0d 0a dd a3 da 02 d3 f2 32 : 178... ..2 0110 e1 5a ce 7f da 7c 6d fe da 51 d0 80 69 6c d4 15 .Z... m. .Q..il.. 0120 b2 6b 10 0e d9 79 ac 31 f4 a2 41 2b ec 1b 52 25 .k....y.1 ..A+...R% 0130 77 e4 4d 23 83 a7 68 60 98 32 1a 21 99 d9 8a 10 w/M#...h` .2!.... 0140 dc e7 92 b1 ed 52 d7 ca 40 e6 9e bb 91 87 11 b3R...@..... 0150 9e b3 2c bc 66 30 0b 6f 07 a0 56 3f 3b 6c 12 dc ..,f0-o ..V?;l.. 0160 f0 f8 19 3f c7 a2 37 d4 78 bb a5 07 2e 7b 71 52 ...?..7. x...{qR 0170 61 4b e8 93 df 87 6f 31 3f 5e 42 90 85 fd 75 d9 aK.....o1 ?^B...u+ 0180 4d 45 59 a6 70 2c e1 cc 1f bd 44 e2 48 24 64 ff MEY.p,.. ..D.H\$d.. 0190 fa be fc b5 dc a2 b5 2f c3 f9 ee d2 ac 27 dc 16/.....' 01a0 5a a8 cf f8 b7 36 0b f6 af f6 49 f1 7c e6 ab 0f Z....6... ..I.. ... 01b0 e8 c9 9b a2 20 cb ce 24 64 5f 55\$ d U</div>						

And below are the **origin** and **decoded** buffers found in the debug log:

uri[0](194)=POST

/?~^B2Z~^?|mQ~@iU~k^P^N~y1A+~^[R%wM#~C~h~X2^Z!~Y~J^P~
~R~R~@~^~Q~G^Q~^~f0^Ko^G|
V?;|~R~^Y?~E7~x~^G.{qRaK~S~Go1?^B~P~E~u~MEY~p,~^~D~H\$d~/_/

```
T^VZ6^K I I^O ~[ $d_U HTTP/1.1
uri[1](165)=POST
/?^Gc^Z2^B|m^Z~@il^Tk^P^N^Y1^A"R%w^A^C h`~X2^Z!~Y^FJ^P^R^Mg
~Q~G^Q~^f0^Ko^G|
V?;|R^Y?^A7^T^G.{qRaK^B^So1?^B~P~E_u^YYp,}D H^B^G"/^C^D^
V^O6^K^A^A^B[ ^Kd_U HTTP/1.1
```

In some special cases, you do need to detect the pattern "POST / HTTP/1.1". Obviously, you will not find it in the context URI so you may hope to detect it in the context packet as below:

```
--pattern "POST / HTTP/1.1"; --context packet; --distance 0,context; --no_case;
```

Unfortunately, this won't work. Surprisingly, the same pattern will work after you splitting it as below:

```
--pattern "POST /"; --context packet; --distance 0,context; --no_case; --pattern " HTTP/1.1"; --context packet; --distance 0; --within 9;
```

This is because the IPS engine does the DFA filtering on the URI buffers only so the signature containing the pattern "POST / HTTP/1.1" got filtered out during the pre-match process.

Mantis ID: https://mantis.fortinet.com/bug_view_page.php?bug_id=0505842

14. Extracting from header context and byte_testing from URI context will not work

Because the IPS Engine performs all matching in the URI context before matching in the Header context, the bytes have not been extracted from the Header yet. Therefore, when the signature goes through the IPS Engine, the byte_test is performed before the register is populated.

Workaround:

Perform the extraction in URI and the byte_test in header. See the following example:

This signature will not work:

```
--pattern "X-Original-URL: /"; --context Header; --extract 4,0,$0,relative,big;
--pattern " /"; --context uri; --within 10,context; --byte_test 4,!,$0,0,relative,big;
```

Change to this:

```
--pattern " /"; --context uri; --within 10,context; --extract 4,0,$1,relative,big;
--pattern "X-Original-URL: /"; --context Header; --byte_test 4,!,$1,0,relative,big;
```

Mantis ID: https://mantis.fortinet.com/bug_view_page.php?bug_id=0547972

8.2 Public Tag

Quite often, a Tag signature can be used to cover multiple vulnerabilities, for example in cases

where vulnerability A contains a Tag-Set signature and vulnerability B uses the same Tag-Set signature. There is a problem with this because disabling the Tag-Set signature for A may unexpectedly prevent the signature for B from working. To solve this problem, analysts need to copy the Tag-Set signature from A to B when writing the signature for B. This Tag is classified as a Private Tag.

The Public Tag was introduced to reduce the total number of Tag-Set signatures. When the number of identical Private Tag copies reaches the threshold (currently 4), the Private Tag is upgraded to a Public Tag. All Public Tags are maintained in a separate report, in report ID 15306.

Below is the list of the commonly used public tags with their usage. The complete list of public tag is being updated constantly; therefore, an analyst should always keep themselves updated with the list. For a complete list of public tag signatures, please refer to report ID 15306 on ISTANCE.

Public Tag	Categories	Usage
Tag.Microsoft.Excel.File.Download	File Download	Detect Microsoft compound document through HTTP
Tag.Microsoft.PowerPoint.File.Download	File Download	Detect Microsoft Office PowerPoint document through HTTP
Tag.Microsoft.Word.File.Download	File Download	Detect Microsoft Office Word document through HTTP
Tag.Microsoft.Publisher.File.Download	File Download	Detect Microsoft Office Publisher document through HTTP
Tag.Microsoft.XLS.File.Download	File Download	Detect Microsoft Office Excel document through HTTP
Tag.MS.Visio.File.Download	File Download	Detect Microsoft Office Visio document through HTTP
-Tag.QuickTimeFileFormat.File.Download	File	Detect Quick Time file,

	Download	including mov, mp4, m4a, m4p, through HTTP
DOT.M3U	File Download	Detect m3u playlist file download through HTTP
DOT.PLS	File Download	Detect pls playlist file download through HTTP
PLAYLIST.HTTP.REQUEST	File Download	Detect m3u and pls playlist file download through HTTP
Tag.RTF.File.Download	File Download	Detect rtf file download through HTTP
-Tag.GIF.File.Download	File Download	Detect GIF file download through HTTP
Tag.Adobe.Dir.File.Download	File Download	Detect Dir file download through HTTP
Tag.Zip.File.Access	File Download	Detect Zip file download through HTTP
Tag.SRV SVC.Bind	RPC bind	Detect DCERPC bind to UUID 4b324fc8-1670-01d3-1278-5a47bf6ee188
SMB.DCERPC.WKSSVC.Bind	RPC bind	Detect DCERPC bind to UUID 6bffd098-a112-3610-9833-46c3f87e345a
SMB.DCERPC.Spoolss.Bind	RPC bind	Detect DCERPC bind to UUID 12345678-1234-abcd-ef00-0123456789ab

TAG.CA.BrightStor.ARCserve.Tapeeng.Engine.RPC.Bind	RPC bind	Detect DCERPC bind to UUID 62b93df0-8b02-11ce-876c-00805F842837
TAG.Trend.Micro.ServerProtect.SPNTSVC.Bind	RPC bind	Detect DCERPC bind to UUID 25288888-bd5b-11d1-9d53-0080c83a5c2c
Tag.Oracle.Connect.Data	Session	Detect remote login to Oracle database
MSNP2P.SESSION	Session	Detect Microsoft MSN messenger session
Tag.Novell.iPrint.ActiveX.Access	ActiveX	Detect ActiveX Classid=36723f97-7aa0-11d4-8919-ff2d71d0d32c

Notes:

- **Tag.Microsoft.Excel.File.Download**

The tag **Tag.Microsoft.Excel.File.Download** is used to detect files using Microsoft component document format, including Microsoft office Word, Excel, PowerPoint, Publisher, Visio, Project, Access etc. The tag name is misleading, but it was used and is being kept for historical reasons. Use **Tag.Microsoft.XLS.File.Download** if you want to check Microsoft Excel files only.

- **The following tags are preferable to Tag.Microsoft.Excel.File.Download**

- Tag.Microsoft.PowerPoint.File.Download
- Tag.Microsoft.Word.File.Download
- Tag.Microsoft.Publisher.File.Download
- Tag.Microsoft.XLS.File.Download
- Tag.Microsoft.Visio.File.Download

These Tag-Set signatures check for some special file type patterns in addition to the patterns used for Tag.Microsoft.Excel.File.Download. For example, the pattern "Content-Type: application/msword" is checked for Tag.Microsoft.Word.File.Download. Signatures using these Public Tags usually have lower risk of false positives than signatures using Tag.Microsoft.Excel.File.Download.

- **The following tags are preferable to PLAYLIST.HTTP.REQUEST**

- DOT.M3U

- DOT.PLS

8.3 file_type

The keyword **file_type** was introduced in IPS engine 1.110. It uses **file magic** to determine what type of file the content is. It works like the **file** command in Linux. To check the type of file in a signature, analysts should use **file_type** instead of a public tag if they check for the same thing. For example, use the pattern "--file_type PDF" instead of using "tag test,Tag.Adobe.PDF.File.HTTP.Download". The following table shows comparable Public Tags and file types.

Public Tag	file_type
-Tag.GIF.File.Download	IMAGE *
Tag.Zip.File.Access	COMPRESS *
Tag.Windows.PE.File.Download	EXE
Tag.SWF.Compressed.File.Download	Flash
Tag.Adobe.PDF.File.HTTP.Download	PDF
Tag.Microsoft.Excel.File.Download	MSOFFICE
Tag.Microsoft.PowerPoint.File.Download	MSOFFICE*
Tag.Microsoft.Publisher.File.Download	MSOFFICE*
Tag.Microsoft.Word.File.Download	MSOFFICE*
Tag.Microsoft.XLS.File.Download	MSOFFICE*
Tag.MS.Visio.File.Download	MSOFFICE*

*** The file_type MSOFFICE/IMAGE/COMPRESS is a super set of the corresponding public tags. Analysts are advised to use the appropriate tag to reduce the risk of false positives.**

8.4 SSL Inspection

More and more applications and services support SSL for security. Although FortiOS supported SSL proxy a long time ago, it is not available to IPS and AppCtrl until 5.0. With FortiOS 5.0 HTTPS, IMAPS, POP3S, SMTPS and FTPS traffic can now be decrypted and inspected by IPS and application control. To enable inspection of SSL sessions, you must add an SSL/SSH Inspection profile to a security policy. You can configure SSL/SSH inspection profiles to inspect HTTPS, SMTPS, POP3S, IMAPS and FTPS traffic. You can configure the profile to control which SSL protocols to inspect, the ports to inspect for each protocol and the certificate to use with SSL sessions.

1. Go to Policy > Policy > SSL/SSH Inspection and create or edit an SSL/SSH inspection profile.
2. Under SSL Inspection Option select the CA certificate to use for SSL sessions. You can import a new certificate or use one already imported into the FortiGate unit.
3. Enable the SSL protocols that you want to inspect and set the ports to inspect for each protocol.
4. Configure other settings as required and select Apply to save your changes.

SSL Inspection Options

CA Certificate Fortinet_CA_SSLProxy ▾

Inspect All Ports ☒

Enable	Protocol	Inspection Port(s)
<input type="checkbox"/>	HTTPS	443
<input type="checkbox"/>	SMTPS	465
<input type="checkbox"/>	POP3S	995
<input type="checkbox"/>	IMAPS	993
<input type="checkbox"/>	FTPS	990

8.4.1 Install FortiGate SSL Proxy certificate

When SSL inspection for HTTPS is enabled on a FortiGate, the web browsers will usually prompt a warning message if the Certificate Authority (CA) for the default certificate used by the Fortigate SSL inspection is not known by the browser. The default certificate in this case is **Fortinet_CA_SSLProxy**.

Internet Explorer will display the warning page :

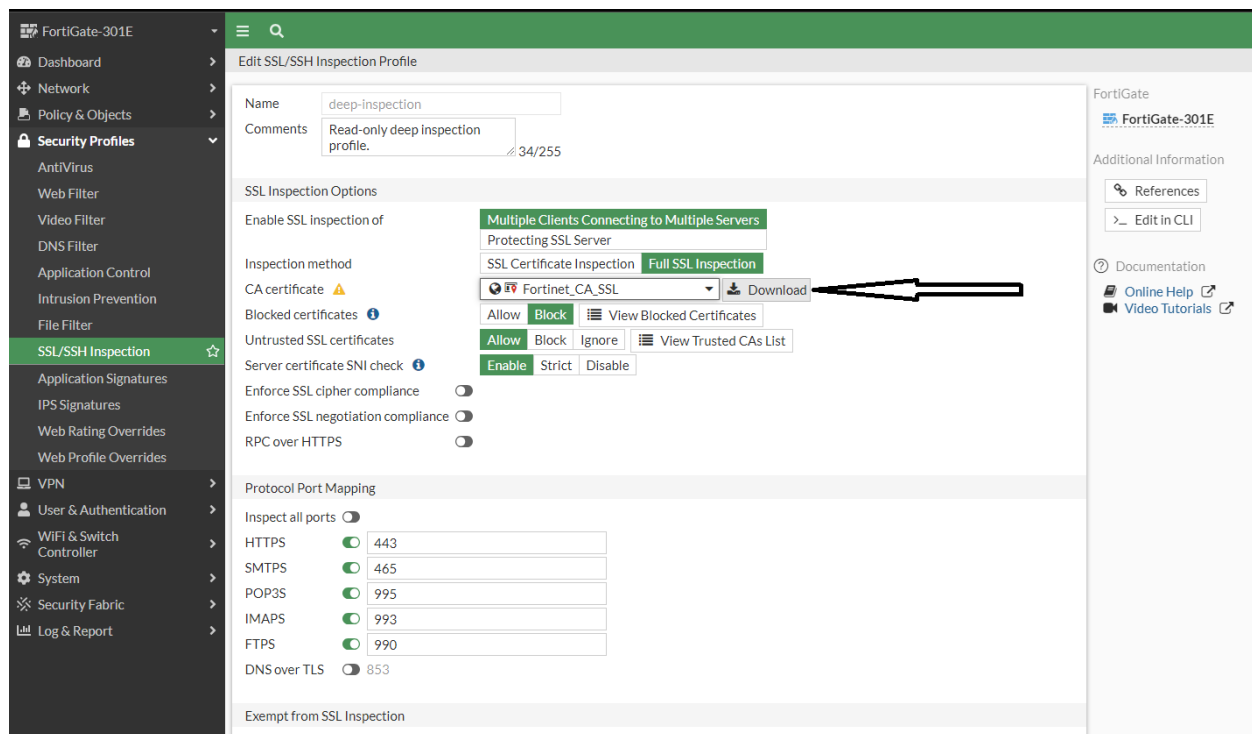


If the user clicks on "**Continue to this website (not recommended)**", the certificate will be temporarily accepted for this connection, but the same message will be prompted at the next connection or when accessing any other HTTPS site.

To avoid warning messages, we need to install the Fortigate root CA in our test environment.

1. Download the FortiGate CA from the Web Based Manager (GUI)

- 1.1. Go to System --> Certificates --> Local Certificates
- 1.2. Select Fortinet_CA_SSLProxy
- 1.3. Click on Download
- 1.4. Save the file Fortinet_CA_SSLProxy.cer



2. Install the root CA in the trusted root certification list of your browser

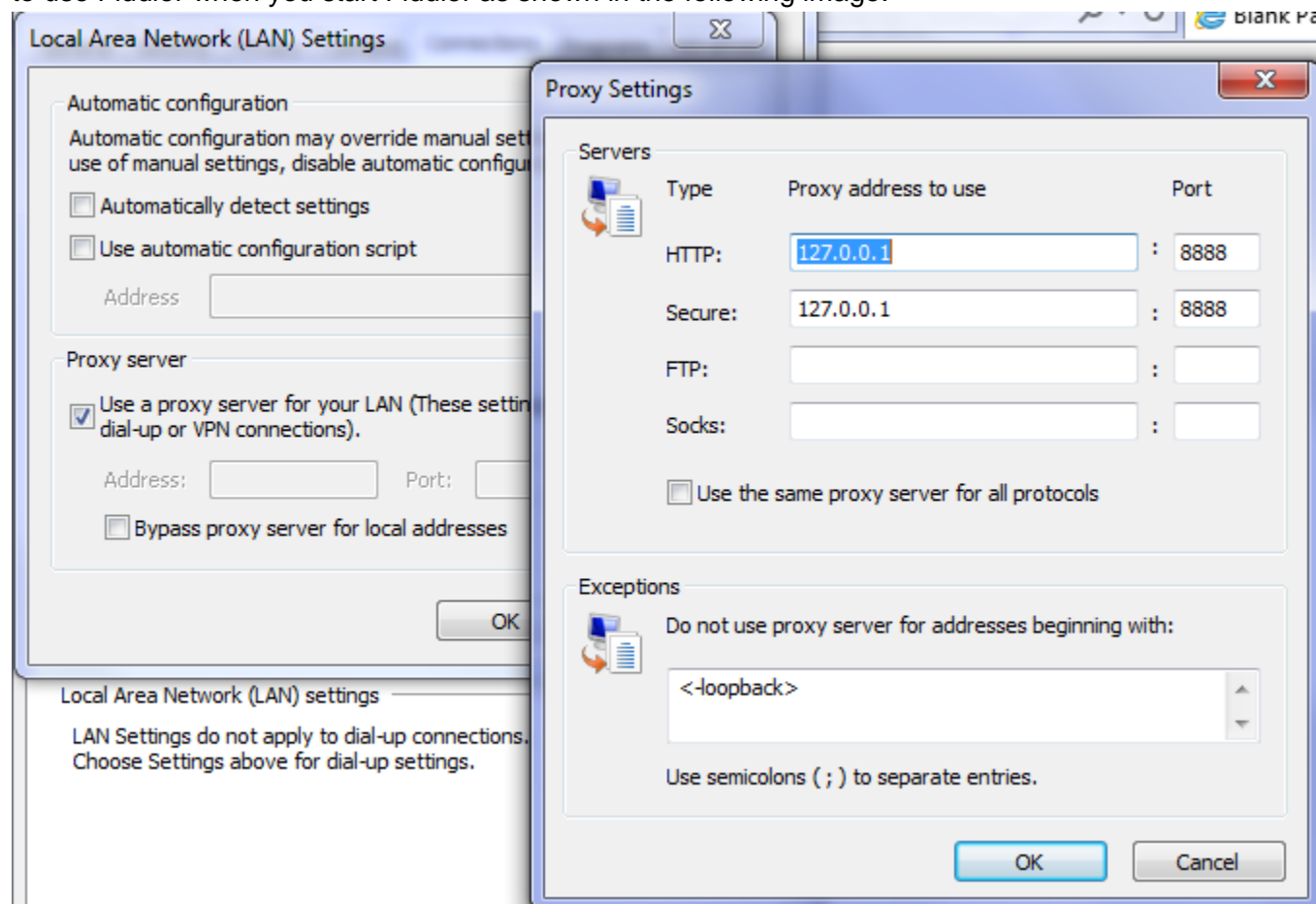
- 2.1. For Chrome, go to Settings --> Privacy and Security --> Security --> Manage Certificates --> Trusted Root Certification Authorities
- 2.2. Click on Import and select the .cer file saved previously; keep all other default options and accept the new Fortigate CA installation
- 2.3. Verify in the Trusted Root Certification Authorities list that the new root certificate is present
- 2.4 Check that the warning message is no longer displayed when accessing an HTTPS Web site

8.4.2 Get Decrypted Content

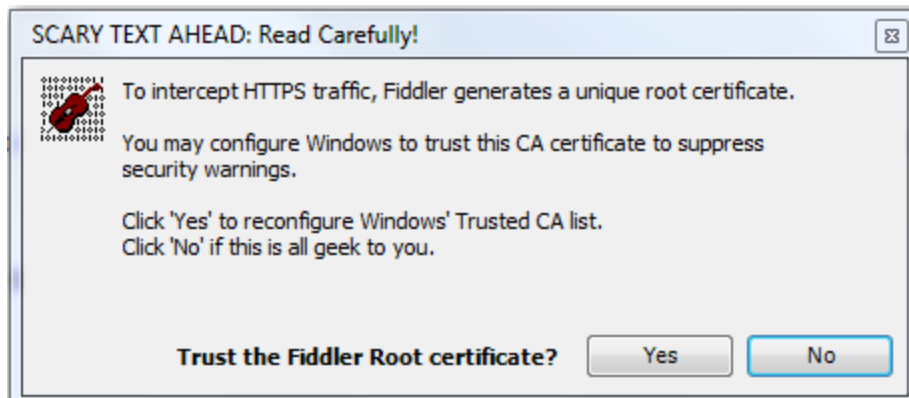
With SSL inspection enabled, SSL proxy will provide IPS engine decrypted data that signature work on. But to write the signature, we first need to get the decrypted content.

1. Using Fiddler

[Fiddler](#) is a [HTTP](#) debugging [proxy server](#) application that can capture HTTP(S) traffic and log it for the user to review. By default, traffic from Microsoft's WinINET HTTP(S) stack is automatically directed to the proxy at runtime, but any browser or application (and most mobile devices) can be configured to route its traffic through Fiddler. The Proxy setting will be changed to use Fiddler when you start Fiddler as shown in the following image.

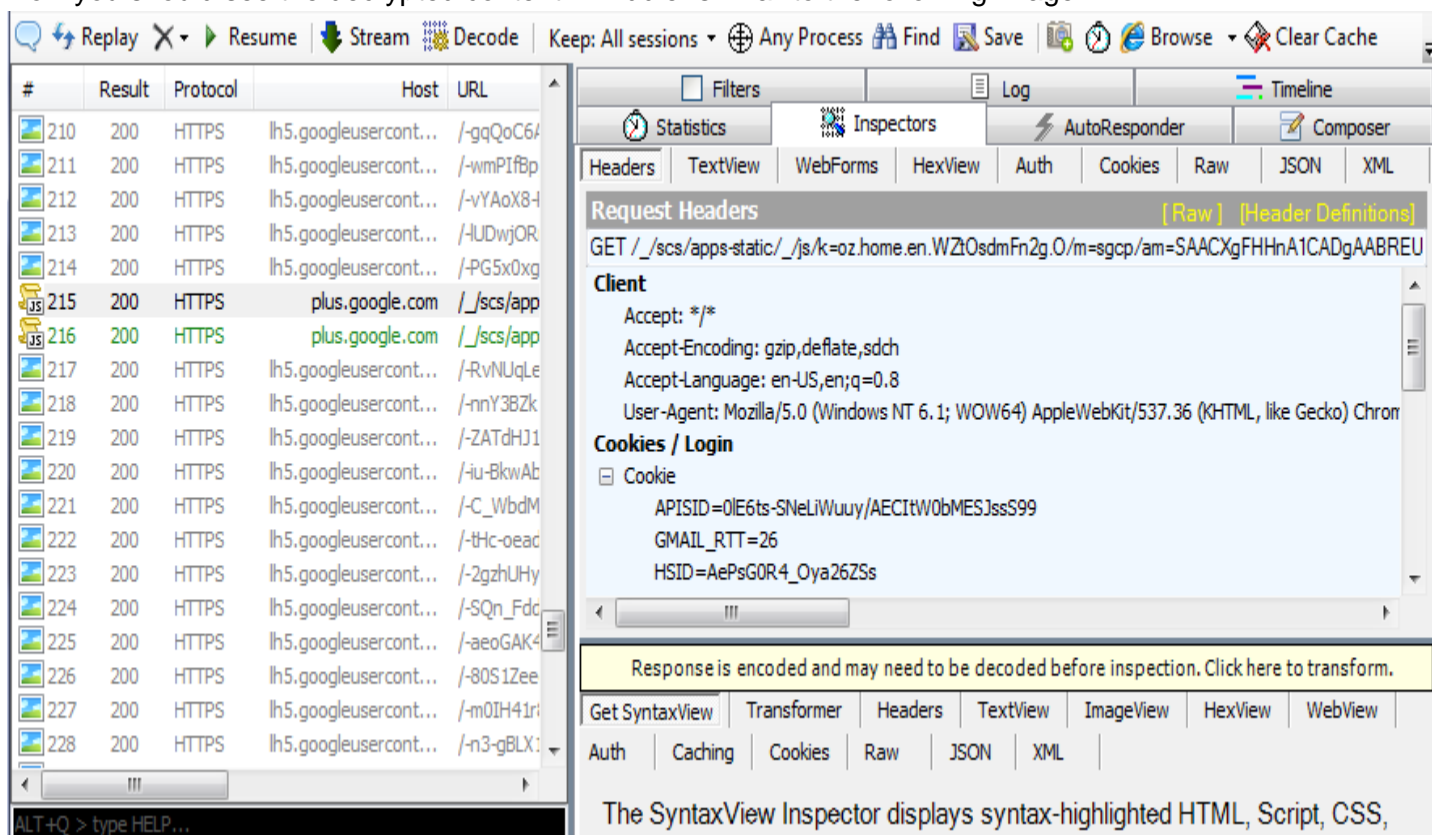


Fiddler2 includes the ability to decrypt, view, and modify HTTPS-secured traffic for debugging purposes. But the decryption feature is disabled by default. You can enable this feature by clicking **Tools > Fiddler Options > HTTPS** and ticking the **Decrypt HTTPS Traffic** box. Same as Fortigate SSL proxy, Fiddler2 relies on a "man-in-the-middle" approach to HTTPS interception. So, you need to install a CA certificate as shown in the following image.



More details can be found at <http://www.fiddlerbook.com/fiddler/help/httpsdecryption.asp>.

Now you should see the decrypted content in Fiddler similar to the following image.



Fiddler is your first choice for decrypting HTTPS traffic.

2. Using IPS debug information

Instead of using a third-party proxy, you can get decrypted content by leveraging the debug feature of IPS engine. Input following CLI commands:

```
diag ips debug enable all
diag debug enable
```


Note.

SSL inspection will only work in combination with one of the other items listed under Security Profiles for example IPS. Therefore, when enabling SSL inspection, you MUST also enable one of these security profiles in the policy.

IPS engine will print all the debug information including traffics it processed.

```
...
PACKET id:704821 len:399 vf:0 fw:1 view:1 log:(1 traffic:0 pre:1 post:0)
  imp2p:0x1 proxy:0xfe
  192.168.102.110:49486 -> 173.194.74.120:443 protocol:6
  IP length:399b, header:20b, ttl:0, tos:0, id:0
  TCP payload:359b, header:20b
  TCP seq:0, ack:0, win:0, flags:*****
[101/0]ips_run_session_verdict_check: @53329 session is ACTIVE
[101/0]ips_store_packet: store tcp packet, size=359
[101/0]ips_dsct_session_loop: @53329 run: http
[101/0]ips_dsct_http_processor: restore uri
[101/0]ips_dsct_http_processor: ips_pkt_id: 704821
0000 45 54 20 2f 61 70 70 73 2f 63 70 61 6e 65 6c 2f  ET /apps/cpanel/
0010 72 65 73 6f 75 72 63 65 73 2f 69 6d 67 2f 69 6d  resources/img/im
0020 67 2f 73 69 74 65 73 2e 67 69 66 20 48 54 54 50  g/sites.gif HTTP
0030 2f 31 2e 31 0d 0a 41 63 63 65 70 74 3a 20 69 6d  /1.1..Accept: im
0040 61 67 65 2f 70 6e 67 2c 20 69 6d 61 67 65 2f 73  age/png, image/s
0050 76 67 2b 78 6d 6c 2c 20 69 6d 61 67 65 2f 2a 3b  vg+xml, image/*;
0060 71 3d 30 2e 38 2c 20 2a 2f 2a 3b 71 3d 30 2e 35  q=0.8, */*;q=0.5
0070 0d 0a 52 65 66 65 72 65 72 3a 20 68 74 74 70 73  ..Referer: https
0080 3a 2f 2f 77 77 77 2e 67 6f 6f 67 6c 65 2e 63 6f  ://www.google.co
0090 6d 2f 61 2f 63 70 61 6e 65 6c 2f 66 6f 72 74 69  m/a/cpanel/forti
00a0 6e 65 74 2e 63 6f 6d 2f 55 73 65 72 48 75 62 0d  net.com/UserHub.
00b0 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65  .Accept-Language
00c0 3a 20 65 6e 2d 55 53 0d 0a 55 73 65 72 2d 41 67  : en-US..User-Ag
00d0 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30  ent: Mozilla/5.0
00e0 20 28 63 6f 6d 70 61 74 69 62 6c 65 3b 20 4d 53  (compatible; MS
00f0 49 45 20 39 2e 30 3b 20 57 69 6e 64 6f 77 73 20  IE 9.0; Windows
0100 4e 54 20 36 2e 31 3b 20 54 72 69 64 65 6e 74 2f  NT 6.1; Trident/
0110 35 2e 30 29 0d 0a 41 63 63 65 70 74 2d 45 6e 63  5.0)..Accept-Enc
0120 6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 64 65 66  oding: gzip, def
0130 6c 61 74 65 0d 0a 48 6f 73 74 3a 20 73 73 6c 2e  late..Host: ssl.
0140 67 73 74 61 74 69 63 2e 63 6f 6d 0d 0a 43 6f 6e  gstatic.com..Con
0150 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c  nection: Keep-Al
0160 69 76 65 0d 0a 0d 0a                               ive....

[101/0]ips_http_handler_uri: enter
```

```
[101/0]ips_http_handler_uri: confirm HTTP
uri[0](53)=GET /apps/cpanel/resources/img/img/sites.gif HTTP/1.1
...
```

You need to store the whole debug session in a file to avoid losing some information.

```
ssh admin@FGT | tee debug.txt
```

Although it is convenient to use the debug information, FGT sometimes doesn't print all the traffics due to:

- Traffic Overload - IPS is fail-open by default. Some traffic may not reach the IPS engine at all.
- IPS engine bug - IPS engine may not print some traffics
- Heavy output job - FortiOS may not print all the information

3. Using Fortigate's ssl-mirror

In IPS Engine 3.072, the dev team added support for a new feature called ssl-mirror. This setting allows the IPS Engine to dump the decrypted traffic into an interface.

This setting can only be configured on the CLI. E.g.

```
edit 12
    set name "internal3sw"
    set uuid 084f0714-5447-51e6-e088-b0afaa506022
    set srcintf "internal3-sw"
    set dstintf "wan2"
    set srcaddr "all"
    set dstaddr "all"
    set action accept
    set schedule "always"
    set service "ALL"
    set utm-status enable
    set logtraffic all
    set ssl-mirror enable
    set ssl-mirror-intf "internal2" "internal4test"
    set application-list "default-5-6"
    set profile-protocol-options "default"
    set ssl-ssh-profile "certificate-inspection"
    set nat enable
next
```

Ssl-mirror first has to be set to enable. Then the interface to forward the decrypted packets to is specified at ssl-mirror-intf. In the example above, the decrypted packets will

be forwarded to internal2 and internal4test. The decrypted packets can then be obtained through the regular packet capture either through “diagnose sniffer packet “” 3 0” or through the packet capture feature on the GUI on FortiOS 5.2 and above with a disk drive.

Note: This feature is implemented in the IPS Engine. The Fortigate has 2 modes to handle deep-inspection - either by the IPS Engine or the proxy (term used in CLI to refer to the proxy is wad). If proxy-mode is used, IPS or App Control must be enabled to redirect the traffic after decryption to the IPS Engine to dump the packet. The proxy cannot do ssl-mirroring. Therefore, if IPS or App Control is not enabled, in proxy-mode, the proxy will not forward the packet to the IPS Engine, and the decrypted packet cannot be dumped.

For more information, please refer to Mantis ID [246805](#), [409770](#) and [412708](#).

8.4.3 Reminders

1. Unset 4.0 and 4.3 flags
2. Add a note in the ‘recommended action’ section of encyclopedia
Following templates are provided for your reference:

AppCtrl

Note: This signature requires "SSL Inspection" in FortiOS 5.0 to detect the application.

IPS

Note: All traffic is SSL encrypted by default and hence this signature requires "SSL Inspection" in FortiOS 5.0 or higher to detect the attack.

Note: All traffics are SSL encrypted by default and hence this signature requires "SSL Inspection" in FortiOS 5.0 or higher with following configuration to detect the attack:
SSL Inspection Options->HTTPS Inspection Port(s): 7443

3. Mark the signature as SSL deep inspection needed

AppCtrl

Set attribute “require_ssl_di” to “Yes” in the application setting page

IPS

Add a note in the report saying the signature requires deep inspection to detect the attack

8.4.4 SSL session logic

When deep-inspection is done by the IPS Engine, 2 sessions will be created for the actual SSL session. They can be differentiated by the “derived” flag. E.g.

PACKET id:84203 len:64 vf:0 fw:12 view:1 **derived:1** log:(traffic:0 pre:0 post:0)
imp2p:0x100 proxy:0x0 features:0x4 flowutm:1
192.168.110.2:5565 -> 31.13.70.36:443 protocol:6
IP length:64b, header:20b, ttl:127, tos:0, id:21234
TCP payload:24b, header:20b
TCP seq:3439458904, ack:272021107, win:258, flags:***AP***
[166/0]ips_run_session_verdict_check: can't find session
[166/0]ips_create_tcp_session: ACK packet from client
[166/0]ips_create_session: enter
[166/0]ips_create_session: set ignore_app_after_size from 0 to 204800 by dependencies of 0
Root
[166/0]ips_proxy_processor: @1770 no proxy detected
[166/0]ips_l7_dsct_processor: @1770 create the only service: http2
[166/0]ips_dsct_session_loop: @1770 only: http2
[166/0]ipsa_adapter_search_prepare: service: HTTP
[166/0]ipsa_adapter_search_prepare: service: HTTP2
[166/0]proc_results: ipsa results:
[166/0]proc_results: [0]: (1, 20423, 40852, 51386)
[166/0]proc_results: [0]: (2, 19431, 40011, 49366)
[166/0]proc_results: [0]: (3, 19352, 39925, 49162)
[166/0]proc_results: [0]: (4, 22017, 42322, 55062)
[166/0]proc_results: [0]: (5, 22016, 42322, 55061)
[166/0]ips_match_rule: pattern matched 32642,47293: Psiphon
[166/0]ips_match_rule: pattern matched 32642,49264: Psiphon

Derived:0 will refer to the SSL encrypted session.

Derived:1 will refer to the decrypted packet.

On 7.0 or above GA released engine, the detection in the derived:0 session will be propagated to the derived:1 session, and the scan-range from the children will be applied to both the derived:0 and derived:1 sessions. Say we have a Facebook signature that checks only --service SSL and not --service HTTP. The detection will be made in the derived:0 session and then the engine will take the scan-range value from the children and apply it to the derived:0 session. Even though we do not have a --service HTTP signature and there will be no trigger on the derived:1 session, the engine will display a "keep last match" message to keep the last matched signature which triggered on the derived:0 session, and update the scan-range accordingly for the derived:1 session as well.

However, for proxy-mode inspection, the IPS engine sees only the derived:1 (all sessions will be marked as derived:0) session as the SSL decryption is done by the proxy (wad). The scan-range of the decrypted session will not be updated if there is only --service SSL signature triggering on the encrypted session.

Hence, when we write a signature for a SSL service, we should still add a signature for the HTTP service too if there are children signatures where the scan-range needs to be updated for the decrypted session.

For more information, please refer to Mantis ID [396989](#).

8.5 SNI verification

SNI verification (Mantis [0483417](#)) is implemented by IPS engine to disallow forged SNI for Application Control. At the beginning IPS engines only passively verify SNI and do not verify SNI before receiving server certificate packets. To enhance SNI verification and server protocol detection, a lightweight SSL client is implemented for active probing (Mantis [0526500](#), [0619824](#)) to account for the first connection or TLS 1.3 connection where the server certificate packet is encrypted. At the time of writing, SSL protocol active probing will come into effect in the following cases:

1. SSL certificate inspection mode when a Web Filter profile is also enabled or
2. SSL deep-inspection mode is enabled

Once a SSL connection is established (for TLS 1.3, this requires active probing or deep inspection), the SNI and server information will be inserted into the "server_cache", which can be viewed by entering the CLI command "diagnose ips share list server_cache", and can be cleared by "diagnose ips share clear server_cache". This entry is shown as "sni_cache" in FOS 6.2 and below.

Common configuration and scenario on SNI verification and active probing are tested in the following cases with corresponding FOS versions:

1. **Does not support SNI verification nor Active Probing** (IPSE 3.544 and below, IPSE 4.023 and below) (these engine versions don't support --context host,SNI and --context host,CN; and don't support TLS 1.3 decryption either):

Scenario \ Protocol	TLS 1.2 and below	TLS 1.3
Certificate inspection	SNI: sub.example.com	SNI: sub.example.com

	<p>CN: example.org</p> <p><i>Match on client hello:</i></p> <pre>--pattern "example.com"; --context host;</pre> <p><i>Match on server cn:</i></p> <pre>--pattern "example.org"; --context host;</pre>	<p>CN: example.org</p> <p><i>Match on client hello:</i></p> <pre>--pattern "example.com"; --context host;</pre> <p><i>No match:</i></p> <pre>--pattern "example.org"; --context host;</pre>
	<p>SNI: sub.example.com</p> <p>CN: fake.cert</p> <p><i>Match on client hello:</i></p> <pre>--pattern "example.com"; --context host;</pre> <p><i>Match on server cn:</i></p> <pre>--pattern "fake.cert"; --context host;</pre>	<p>SNI: sub.example.com</p> <p>CN: fake.cert</p> <p><i>Match on client hello:</i></p> <pre>--pattern "example.com"; --context host;</pre> <p><i>No match:</i></p> <pre>--pattern "fake.cert"; --context host;</pre>

Certificate inspection	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p><i>Match on client hello:</i></p> <p>--pattern "example.com";</p> <p>--context host;</p> <p><i>Match on server cn:</i></p> <p>--pattern "example.org";</p> <p>--context host;</p>	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p><i>Match on client hello:</i></p> <p>--pattern "example.com";</p> <p>--context host;</p> <p><i>No match:</i></p> <p>--pattern "example.org";</p> <p>--context host;</p>
------------------------	---	---

SNI: sub.example.com

CN: fake.cert

Match on client hello:

--pattern "example.com";

--context host;

Match on server cn:

--pattern "fake.cert";

--context host;

SNI: sub.example.com

CN: fake.cert

Match on client hello:

--pattern "example.com";

--context host;

No match:

--pattern "fake.cert";

--context host;

Deep inspection	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p><i>Match on client hello:</i></p> <p>--pattern "example.com";</p> <p>--context host;</p> <p><i>Match on server cn:</i></p> <p>--pattern "example.org";</p> <p>--context host;</p>	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p><i>Match on client hello:</i></p> <p>--pattern "example.com";</p> <p>--context host;</p> <p><i>No match:</i></p> <p>--pattern "example.org";</p> <p>--context host;</p>
-----------------	---	---

SNI: sub.example.com CN: fake.cert <i>Match on client hello:</i> --pattern "example.com"; --context host; <i>Match on server cn:</i> --pattern "fake.cert"; --context host;	SNI: sub.example.com CN: fake.cert <i>Match on client hello:</i> --pattern "example.com"; --context host; <i>No match:</i> --pattern "fake.cert"; --context host;
--	--

As you can see, without the SNI verification, our signatures(highlighted with the red color) can be easily bypassed.

2. **Support SNI verification but does not support Active Probing** (IPSE 3.551 and above, 4.047 and above, 5.002 and below)

Scenario \ Protocol	TLS 1.2 and below	TLS 1.3
---------------------	-------------------	---------

<p>Certificate inspection / SNI verified</p>	<p>SNI: sub.example.com CN: example.org</p> <p><i>Match on client hello:</i></p> <pre>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI;</pre> <p><i>Match on server cn:</i></p> <pre>--pattern "example.org"; --context host,CN;</pre> <p><i>No match:</i></p> <pre>--pattern "example.org"; --context host;</pre>	<p>SNI: sub.example.com CN: example.org</p> <p><i>Match on client hello:</i></p> <pre>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI;</pre> <p><i>No match:</i></p> <pre>--pattern "example.org"; --context host; --pattern "example.org"; --context host,CN;</pre>
--	---	---

SNI: sub.example.com

CN: fake.cert

No match:

--pattern "example.com";

--context host;

--pattern "example.com";

--context host,SNI;

Match on server cn:

--pattern "fake.cert";

--context host;

--pattern "fake.cert";

--context host,CN;

SNI: sub.example.com

CN: fake.cert

***Match after ssl
handshake:***

--pattern "example.com";

--context host;

No match:

--pattern "example.com";

--context host,SNI;

--pattern "fake.cert";

--context host;

--pattern "fake.cert";

--context host,CN;

Deep inspection / SNI verified	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p><i>Match on client hello:</i></p> <p>--pattern "example.com";</p> <p>--context host;</p> <p>--pattern "example.com";</p> <p>--context host,SNI;</p> <p><i>Match on server cn:</i></p> <p>--pattern "example.org";</p> <p>--context host,CN;</p> <p><i>No match:</i></p> <p>--pattern "example.org";</p> <p>--context host;</p>	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p><i>Match on client hello:</i></p> <p>--pattern "example.com";</p> <p>--context host;</p> <p>--pattern "example.com";</p> <p>--context host,SNI;</p> <p><i>Match on server cn:</i></p> <p>--pattern "example.org";</p> <p>--context host,CN;</p> <p><i>No match:</i></p> <p>--pattern "example.org";</p> <p>--context host;</p>
--------------------------------	---	---

SNI: sub.example.com

CN: fake.cert

No match:

--pattern "example.com";

--context host;

--pattern "example.com";

--context host,SNI;

Match on server cn:

--pattern "fake.cert";

--context host;

--pattern "fake.cert";

--context host,CN;

SNI: sub.example.com

CN: fake.cert

No match:

--pattern "example.com";

--context host;

--pattern "example.com";

--context host,SNI;

Match on server cn:

--pattern "fake.cert";

--context host;

--pattern "fake.cert";

--context host,CN;

<p>Certificate inspection / SNI unverified and first connection</p>	<p>SNI: sub.example.com CN: example.org</p> <p><i>Match after ssl handshake:</i></p> <pre>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host,CN;</pre> <p><i>No match:</i></p> <pre>--pattern "example.org"; --context host;</pre>	<p>SNI: sub.example.com CN: example.org</p> <p><i>Match after ssl handshake:</i></p> <pre>--pattern "example.com"; --context host;</pre> <p><i>No match:</i></p> <pre>--pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host; --pattern "example.org"; --context host,CN;</pre>
---	---	---

SNI: sub.example.com

CN: fake.cert

No match:

--pattern "example.com";

--context host;

--pattern "example.com";

--context host,SNI;

Match on server cn:

--pattern "fake.cert";

--context host;

--pattern "fake.cert";

--context host,CN;

SNI: sub.example.com

CN: fake.cert

***Match after ssl
handshake:***

--pattern "example.com";

--context host;

No match:

--pattern "example.com";

--context host,SNI;

--pattern "fake.cert";

--context host;

--pattern "fake.cert";

--context host,CN;

<p>Deep inspection / SNI unverified and first connection</p>	<p>SNI: sub.example.com CN: example.org</p> <p><i>Match after ssl handshake:</i></p> <p>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host,CN;</p> <p><i>No match:</i></p> <p>--pattern "example.org"; --context host;</p>	<p>SNI: sub.example.com CN: example.org</p> <p><i>Match after ssl handshake:</i></p> <p>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host,CN;</p> <p><i>No match:</i></p> <p>--pattern "example.org"; --context host;</p>
--	---	---

SNI: sub.example.com CN: fake.cert No match: --pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; Match on server cn: --pattern "fake.cert"; --context host; --pattern "fake.cert"; --context host,CN;	SNI: sub.example.com CN: fake.cert No match: --pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; Match on server cn: --pattern "fake.cert"; --context host; --pattern "fake.cert"; --context host,CN;
--	--

As you can see, it is still possible to bypass our signatures when the deep inspection is not enabled and the SNI is not verified yet for the TLS 1.3. Many customers are reluctant to enable the SSL deep inspection due to many concerns. So active probing was introduced to work around this issue.

3. **Support SNI verification and support Active Probing** (IPSE 5.003 and above, 6.010 and above; Active Probing improved in 5.211 and 6.022)

Scenario \ Protocol	TLS 1.2 and below	TLS 1.3
Certificate or deep inspection / SNI verified (or record exists in server_cache)	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p>(server certificate subjectAltName includes *.example.com)</p> <p>Match on client hello:</p> <pre>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host,CN;</pre> <p>No match:</p> <pre>--pattern "example.org"; --context host;</pre>	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p>(server certificate subjectAltName includes *.example.com)</p> <p>Match on client hello:</p> <pre>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host,CN;</pre> <p>No match:</p> <pre>--pattern "example.org"; --context host;</pre>

SNI: sub.example.com

CN: fake.cert

No match:

--pattern "example.com";

--context host;

--pattern "example.com";

--context host,SNI;

Match on client hello:

--pattern "fake.cert";

--context host;

--pattern "fake.cert";

--context host,CN;

SNI: sub.example.com

CN: fake.cert

No match:

--pattern "example.com";

--context host;

--pattern "example.com";

--context host,SNI;

Match on client hello:

--pattern "fake.cert";

--context host;

--pattern "fake.cert";

--context host,CN;

<p>Certificate inspection (no active probe) /</p> <p>SNI unverified and First connection (record not in server_cache)</p>	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p>(server certificate subjectAltName includes *.example.com)</p> <p>Match after ssl handshake:</p> <pre>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host,CN;</pre> <p>No match:</p> <pre>--pattern "example.org"; --context host;</pre>	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p>(server certificate subjectAltName includes *.example.com)</p> <p>Match after ssl handshake:</p> <pre>--pattern "example.com"; --context host;</pre> <p>No match:</p> <pre>--pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host; --pattern "example.org"; --context host,CN;</pre>
---	--	--

<p>SNI: sub.example.com</p> <p>CN: fake.cert</p> <p>No match:</p> <p>--pattern "example.com";</p> <p>--context host;</p> <p>--pattern "example.com";</p> <p>--context host,SNI;</p> <p>Match on server cn:</p> <p>--pattern "fake.cert";</p> <p>--context host;</p> <p>--pattern "fake.cert";</p> <p>--context host,CN;</p>	<p>SNI: sub.example.com</p> <p>CN: fake.cert</p> <p>Match after ssl handshake:</p> <p>--pattern "example.com";</p> <p>--context host;</p> <p>No match:</p> <p>--pattern "example.com";</p> <p>--context host,SNI;</p> <p>--pattern "fake.cert";</p> <p>--context host;</p> <p>--pattern "fake.cert";</p> <p>--context host,CN;</p>
---	--

<p>Deep inspection (active probe working) / SNI unverified and First connection (record not in server_cache)</p>	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p>(server certificate subjectAltName includes *.example.com)</p> <p>Match on client hello:</p> <pre>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host,CN;</pre> <p>No match:</p> <pre>--pattern "example.org"; --context host;</pre>	<p>SNI: sub.example.com</p> <p>CN: example.org</p> <p>(server certificate subjectAltName includes *.example.com)</p> <p>Match on client hello:</p> <pre>--pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; --pattern "example.org"; --context host,CN;</pre> <p>No match:</p> <pre>--pattern "example.org"; --context host;</pre>
--	--	--

SNI: sub.example.com CN: fake.cert No match: --pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; Match on client hello: --pattern "fake.cert"; --context host; --pattern "fake.cert"; --context host,CN;	SNI: sub.example.com CN: fake.cert No match: --pattern "example.com"; --context host; --pattern "example.com"; --context host,SNI; Match on client hello: --pattern "fake.cert"; --context host; --pattern "fake.cert"; --context host,CN;
---	---

8.6 Best Practices

To achieve better performance and accuracy in signature matching, we can improve our signatures in the following ways.

- Specify the service name, port number or port range.**
Signatures that do not specify a port or service are called generic signatures. Ports can be specified by --dst_port and --src_port while service can be specified by --service options.

IPS signatures are organized by their protocols and port/service options. For a given packet, only signatures that match the protocol and port/service of the packet are inspected. If matching signatures cannot be found, the session that the packet belongs to will be ignored by IPS, meaning the rest of packets in the same session will not be sent to IPS. This can be a huge boost for IPS performance, not only do fewer sessions need to be scanned, but context switches between firewall and IPS are also reduced. However, there are generic signatures for a protocol, even if it is only one, no session can be ignored for that protocol.

For services that use dynamic ports, we can specify a port range, like **1024:** which covers any port from 1024 and 65535.

- **Specify the context whenever it is possible.**

The IPS engine can parse HTTP, SMTP, IMAP, POP3 and SIP protocols, and identify the different sections for those protocols in packets. Using **context** to specify a section can reduce the pattern search range, improving detection accuracy and performance. Analysts should always specify **context** in signatures detecting traffics for those protocols.

Please note that the following definitions are not the same.

Here the range is the first 30 bytes of the packet payload.

```
--pattern "ABC"; --within 30,packet;
```

Here the range is the first 30 bytes of the body section.

```
--pattern "ABC"; --context body; --within 30,context;
```

- **Add the range modifiers (within, distance) for each pattern or PCRE.**

If no range modifiers are specified for a pattern or PCRE, the IPS engine will look for a match at every position from the beginning of a packet to the end. To improve performance and reduce false positives, a match range should be set and limited to as small a range as possible for the detection.

A negative value can be used as a distance modifier to search for a pattern before the last match. For example, to search for the script text,

```
"ActiveXObject("IncrediShellExt.IMMenuShellExt");"
```

we can use the following definition:

```
--pattern "IncrediShellExt.IMMenuShellExt"; --context body; --no_case;  
--pattern "ActiveXObject"; --context body; --no_case; --distance -50; --within 20;
```

Here we search for a more unusual string first. Then we search from an earlier location to find the remaining part within a limited range.

The default reference point for **within** and **distance** is the last match point, not the beginning of the packet or context.

- **Increase the pattern length.**

Shorter patterns have a higher probability of matching something unexpected, resulting in a higher risk of false positive. This is especially true for patterns that do not have strict range limits. If it is impossible to put a tight limit on the range, the pattern should not be shorter than 4 bytes.

- **Increase the number of patterns and tag them more tightly.**

This point is linked with the previous one. If a pattern is short, and common in packets, it is more likely to cause false positives. In this case we need to increase the number of patterns and tag them tightly.

- It is forbidden to use PCRE with extensive searches like `".*"`.

The IPS Engine handles PCRE matching quite slowly compared to normal pattern matching. Sometimes one signature can cause 20 percent throughput degradation. This often happens on signatures that examine HTTP server-side packets, because most traffic on the Internet is HTTP and the length of the HTTP server response is normally much longer than the client request.

The current solution is to identify patterns that can be matched by normal pattern matches and add them before any PCRE patterns, so that packets can be filtered by the normal patterns first. These patterns cannot be trivial. Patterns like "href=" or "name=|22|" are far too common in HTTP pages to filter the unwanted packets out effectively.

Not all PCRE options have the same effect on performance. Generally using PCRE to match long and unknown length strings causes greater degradation. An example of this is `".*"`, which we should avoid using.

The following is a **bad PCRE** definition, intended to detect SMTP traffic.

[illegible]

When this matches an email containing many occurrences of "**To:**", the CPU increases to 99%, which is quite close to it collapsing and will lead to the network being almost broken.

- **Multi-search patterns can't be used in the DFA, so it has a similar performance impact as PCRE**

Make sure the signature has at least one pattern that can be used in the DFA. If this is

impossible, split the signature and use the option **pattern** instead.

- **Specific patterns should be matched first, in order not to evaluate the whole signature when it is not going to match.**

For example, the following signature can be improved by changing pattern order and adding range limits:

```
--pattern "Content-Type|3a 20|application/octet-stream"; --context body; --no_case; --pattern "name=|22|upxout.exe2|22|"; --context body; --no_case; --within 40;
--pattern "Content-Transfer-Encoding|3a 20|base64"; --context body; --no_case; -distance 0;
--pattern "TVpLRVJORUwzMi5ETEw"; --context body; --no_case; --distance 0;
```

Here is the improved version:

```
--pattern "name=|22|upxout.exe2|22|"; --context body; --no_case;
--pattern "Content-Type|3a 20|application/octet-stream"; --context body; --no_case; --distance -100; --within 80;
--pattern
"TVpLRVJORUwzMi5ETEwAAExvYWRMaWJyYXJ5QQAAABHZXRQcm"; --context body; --no_case; --distance 0;
--pattern "Content-Transfer-Encoding|3a 20|base64"; --context body; --no_case; -distance -150; --within 120;
```

- **Do not completely depend on common patterns, excluding the common HTTP header fields.**

Our signatures should not depend completely on common patterns, because they will be matched easily on the Internet. They increase the risk of performance degradation and high false positive rates. If we need to check some common protocol fields, for example in the HTTP header, we can send a request to the engine team to see if a decoder can do it more efficiently.

- **HTTP signatures.**

Adding more and more signatures will gradually slow down IPS. It makes sense to remove or disable some signatures. Like for example, a vulnerability that is out-of-date and does not appear in the wild. We do not need signatures for vulnerabilities in software that are not so popular or commonly used if no exploit exists in popular testing tools like Metasploit, BP, Spirent, etc.

Because HTTP signatures make up more than 60% of our signature database, and the volume of HTTP traffic on the Internet is huge, these signatures have a major effect on performance. For this reason, the quality of HTTP signatures must be as high as possible.

- **General buffer overflow vulnerability detection.**

Usually, we use the following signature format to detect a general buffer overflow.

```
--pattern "XXX"; --pattern !"Y"; --within_abs M;
```

Because the first pattern is usually common, and the second match is time-consuming, we should try to limit the number of signatures of this type. If this kind of vulnerability occurs in protocols, we can request the engine team to cover it in the IPS engine.

- **Specify seq and ack on uncommon protocol data exchanges if possible.**

The IPS engine constructs a session sequence number for TCP, UDP and ICMP. This number can be used to help reduce false positives.

- **Set the severity accurately.**

The Severity level is valuable information for our customers. If the severity level of a signature is **Critical**, they can be certain that an attack detected by it is a serious threat to their IT infrastructure.

To deliver correct information to our customers, and not mislead them, we need to set the correct severity level. Most importantly, do not allow a **Low** severity threat to be set mistakenly as **High** or **Critical**. If vulnerable software is not popular, do not set its severity to **Critical**. It is important that the quality of **Critical** signatures be as high as possible.

- **Make use of cset tag for multiple sessions to the same destination IP.**

Some applications have multiple protocols that can be used to transmit data. For e.g. a VPN could have a HTTP, HTTPS, DNS and a proprietary protocol that it will try to connect to its server. Some of the protocols may be hard to write a signature for. If one or two of the protocols are easier to write signatures for and those protocol sessions are sent out at around the same time, we can add signatures for the easy ones with a cset tag and use a tag test on the protocols that are harder to write signatures for.

- **Try to identify packet headers and make educated guesses on proprietary protocols.**

For many proprietary TCP protocols (similar for UDP except no packet length most of the time), here are the possible scenarios:

- 1) Packet is encrypted right away - can be identified if we see random bytes with almost no repeating bytes from 0x00-0xFF.
- 2) Packet is not encrypted at all - can see meaningful strings, a lot of zeroes.
- 3) Some packets in the beginning are not encrypted followed by encrypted packets.

For 2) and 3), there are a couple of standard practices that can be employed to try to get a start on analyzing the protocol.

- a) Identify if there is any obvious magic number - strings or acronyms that might be related to analyzed applications.
- b) Try to draw out a rough picture of the packet header - before the payload.
- c) Look for the payload length bytes - typically 2 or 4 bytes, in big or little-endian format in the packet header.

- **Read RFC or official documentation.**

Many protocols may have datasheets that specify the packet bytes. There is no need to guess, or reverse engineer them.

8.7 Signature Templates

- **HTTP Detection**

The template below uses a part of the URI and the HTTP Host header value as patterns:

```
F-SBID( --name "NameOfTheWebApp"; --protocol tcp; --service HTTP; --flow
from_client; --parsed_type HTTP_POST; --pattern "/uripart"; --context uri; --
no_case; --pattern "domain.name.com"; --no_case; --context host; --within
100,context; )
```

- **HTTPS Detection**

The template below detects the domain name in the HTTPS website certificates:

```
F-SBID( --name "NameOfTheWebApp"; --protocol tcp; --service SSL; --flow
from_server; --pattern "domain.name.com"; --context host; --no_case; )
```

Note: all common names are available in 'host' context. This feature has been supported since engine 1.115.

- **ActiveX Control Method Detection**

The object can be loaded with either the Class ID or Object Name. We should detect it both ways, like in the example below, where:

Class ID = 7EC7B6C5-25BD-4586-A641-D2ACBB6629DD

Object Name = YDPCTL.YDPControl.1

Vulnerable method/property = GetComponentVersion

```
F-SBID( --name "Yahoo.Widgets.YDP.ActiveX.Control.Command.Execution"; --
protocol tcp; --status hidden; --service HTTP; --flow from_server;
--pattern "7EC7B6C5-25BD-4586-A641-D2ACBB6629DD"; --context body; --
no_case;
--pcre
"/<OBJECT\s+[^>]*classid\s*=\s*[\x22\x27]? \s*clsid\s*\x3a\s*\x7B? \s*7EC7B
6C5-25BD-4586-A641-D2ACBB6629DD/i"; --context body; --distance -100; --
within 100;
--tag set,Tag.Yahoo.Widgets.YDP.ActiveX.Control.Command.Execution; --tag
quiet;)
```

```
F-SBID( --name "Yahoo.Widgets.YDP.ActiveX.Control.Command.Execution"; --
protocol tcp; --status hidden; --service HTTP; --flow from_server;
--pattern "YDPCTL.YDPControl.1"; --context body; --no_case;
```

```
--pcre "/(ActiveX|Create)Object/i"; --context body; --distance -100; --within 81;  
--tag set,Tag.Yahoo.Widgets.YDP.ActiveX.Control.Command.Execution; --tag  
quiet;)
```

```
F-SBID( --name "Yahoo.Widgets.YDP.ActiveX.Control.Command.Execution"; --  
protocol tcp; --service HTTP; --flow from_server;  
--pattern "GetComponentVersion("; --context body; --no_case;  
--tag test,Tag.Yahoo.Widgets.YDP.ActiveX.Control.Command.Execution;)
```

- **DNS Query Detection**

The template below detects the domain name in a DNS query:

```
F-SBID( --name "Abc.com.DNS.Query"; --protocol udp; --service DNS; --flow  
from_client; --byte_test 1,<,128,2; --pattern "abc.com"; --context host; --  
no_case; )
```

Note: Host names in DNS queries are stored in a buffer and connected by '\0'. This feature has been available since **engine 1.125** .

- **Cset Tag**

```
F-SBID( --name "SkyVPN"; --protocol tcp; --app_cat 6; --service HTTP; --flow  
from_client; --pattern "User-Agent"; --no_case; --context header; --pattern "skyvpn."; --  
context host; --no_case; --tag cset,SkyVPN.SSL,300,src_ip,dst_ip,all_sessions; --  
scan-range 2k,all; )
```

```
F-SBID( --name "SkyVPN"; --protocol udp; --app_cat 6; --tag test,SkyVPN.SSL; --scan-  
range 2k,all; )
```

```
F-SBID( --name "SkyVPN"; --protocol tcp; --app_cat 6; --tag test,SkyVPN.SSL; --scan-  
range 2k,all; )
```

In the examples above, the HTTP User-Agent signature will set a tag on the src_ip and dst_ip for 300 seconds when detecting the HTTP protocol of SkyVPN. Since it is simple to detect, we are using this protocol to tag the src_ip and dst_ip pair and identify them with the next 2 signatures instead of writing difficult pattern checks for the encrypted sessions.

9 Signature Quality Assurance

9.1 Overview

Signature quality will be measured by these three major factors: **false negatives**, **false positives** and **performance**. To improve the overall quality of our signatures, we have

implemented the following processes:

- Set up a signature review team to do signature review weekly.
- Set up a process for new analysts to undergo training and to go through a qualification examination if possible after the training is completed.
- Signatures should be released as a beta signature first for testing before it can be officially released.
- Every signature will go through a coverage test, and a false positive test against our clean traffic archive, before its release.
- Establish measurable metrics weekly to gauge the signature quality.
- Every signature's triggered event from the FDN network, traffic submissions from live FortiGates and corporate traffic samples, will be monitored.
- Constantly increase and update our clean traffic archive.

9.2 Review Team

The review team comprises senior analysts from the IPS analyst team. The responsibilities of this team are as follows:

- Review new signatures
- Approve upgrade of signature status and action
- Maintain the guidelines for signature development.
- Mark the analyst qualification exams to determine if new analysts are qualified after their training is done.

9.3 False Negatives

Here are the steps we take to ensure that a vulnerability is properly covered by the signatures that are developed for it:

- Analysts should focus mainly on developing signatures based on vulnerability analysis and avoid developing signatures based on an actual exploit or PoC. Following patterns should be avoided if possible:
 - Copyright information. For example, " Copyrights TELUS Security Labs"
 - Patterns like "exploit", "poc.html", "Proof of Concept" or "Example"
 - IP addresses. For example, "http://172.16.1.2/"
 - Patterns like "AAAAAAAAAA", "BBBBBBBBBB", etc
- Every pcap file loaded onto ISTATE should be triggered and verified by QA.
- QA will regularly run the coverage test, to measure our coverage on these penetration tools, including every supported evasion technique:
 - Metasploit
 - Mu Security Appliance
 - Trend Micro

- BreakingPoint
- For every new engine release, the coverage test will be run to ensure that there is no detection losses with the new engine.

The results of our IPS coverage will be reported in our weekly IPS metrics.

9.4 False Positives

Fighting against false positives in IPS will always be a major challenge, particularly due to these factors:

- It is difficult to know when a false positive occurs, as it is hard to catch the traffic that triggers a false positive. Therefore, it can be difficult for an analyst to fix the FP problem if it cannot be reproduced.
- It is often hard to obtain details of a discovered vulnerability.
- The techniques available with the current IPS detection technology are still limited. We often must develop signatures even though we know that the signatures will have false positives. In those cases, the signatures should be released as disabled, by default.

Here are the measures we have adopted to improve on the issues mentioned above, and to minimize the chance of false positives. They are mostly driven by IPS QA:

- Analysts will focus more on vulnerability analysis signature development, instead of just writing some pattern matching rules to capture for a specific POC or exploit.
- Every new signature will be released as a beta rule first, unless we are in an urgent and time-sensitive situation. A beta rule will allow us to see the triggered event numbers reported back from the FDN network, while hiding the information from our customers. Any rule with an unusually high trigger rate will be rejected before its official release. The beta release process will be directly handled by the QA team.
- All the new IPS signatures should be evaluated if their alert level is more than 10 FGT/day during the beta process. Analyst should give the reason in a signature comment if he thinks it is not necessary to evaluate such a signature.
- If a customer reports trigger of an IPS signature and the signature has never been evaluated, the analyst handling the ticket should evaluate the signature.
- Analysts **SHOULD** leave a signature comment on ISTANCE after evaluating the signature. The comment should conclude the evaluation result. This information is very important for QA and other analysts.
- Analysts cannot evaluate AppCtrl signatures. To work around this issue, we created a report for evaluating app ctrl signatures: report 27840 ([App.Control.Signature.Test](#)). Analysts can add their signature to the report for testing before creating or adding it to an app ctrl report. The signature will be released as IPS signature, so analysts have a chance to evaluate it during the beta process.

Note.

- a. Analyst **must set the "Is Pending" flag** to avoid releasing the signature

accidentally.

- b. The signature added in vid 27840 **should not** affect IPS or AppCtrl signatures in other reports. For example, analysts **should not** add a tag-set signature in vid 27840 that sets a tag name tested by signature in other reports. Tag name ending with “.Beta” is suggested for tag-set signature in vid 27840.
 - c. After evaluation, the signature should be abandoned
 - d. It is fine to reuse signatures previously abandoned for testing other applications
- The corporate traffic for the Vancouver office will be sample-captured and reviewed weekly. This is also to measure the false positive ratio, which will be reported in our weekly metrics.
 - The Clean Traffic Archive will be constantly built up. Any signature triggered on the archive will be further analyzed by analysts before the signature is released or after it is released. The size of this Clean Traffic Archive will be reported in our weekly metrics as well.
 - QA's automatic testing system will report false positive triggered events and detection events weekly to the analyst team. Analysts can do a one-click download of the triggered traffic for further analysis.

When receiving FP complaints from customers, it is always recommended to ask for a traffic capture or packet log unless you know what was happening there. It is a chance for us to improve the signature. Two case studies are provided in the following section.

9.4.1 Case study - MS.WinVerifyTrust.Signature.Validation.Remote.Code.Execution

Signature MS.WinVerifyTrust.Signature.Validation.Remote.Code.Execution was added as a generic signature to cover CVE-2012-0151:

“

A code injection vulnerability exists in the Windows Authenticode Signature Verification function. The vulnerability is due to improper checking of the size and location of the Attribute Certificate Table. The vulnerable code does not verify that the Attribute Certificate Table contains only the certificates and nothing else. It is possible for the malicious attackers to increase the size of the Attribute Certificate Table and inject arbitrary content in its tail.

”

Since it is impossible to check whether the Attribute Certificate Table contains only the certificate, our signature instead checks its abnormally large size and look for the malicious ZIP file at the end:

```
F-SBID( --vuln_id 31536; --attack_id 38507; --name
"MS.WinVerifyTrust.Signature.Validation.Remote.Code.Execution"; --group applications3; --
protocol tcp; --default_action pass; --revision 4351; --severity critical; --vuln_type "Other"; --app
Other; --os Windows; --status hidden; --file_type EXE; --pattern "PE|00 00|"; --context file; --
pattern "|0B 01|"; --context file; --distance 20; --within 2; --byte_test 4,>,0x1000,126,relative,little;
--tag set,Tag.EXE.Signature; --tag quiet; --date 20150501; )
F-SBID( --vuln_id 31536; --attack_id 41527; --name
```

"MS.WinVerifyTrust.Signature.Validation.Remote.Code.Execution"; --group applications3; --protocol tcp; --default_action drop; --revision 4405; --severity critical; --app Other; --os Windows; --tag test,Tag.EXE.Signature; --pattern "PK|05 06|"; --context file; --distance 22,context,reverse; --within 4,context; --pattern "|2E|exe"; --context file; --distance -10; --within 6; --date 20150501;)

The patterns checked by aid 38507 are underlined in the below image.

00D0h: 52 69 63 68 67 B1 F3 C9 00 00 00 00 00 00 00 00 Richg±óĒ.....

00E0h: 50 45 00 00 4C 01 05 00 16 1D E1 4D 00 00 00 00 PE..L.....áM....

00F0h: 00 00 00 00 E0 00 03 01 05 01 09 00 00 CE 00 00à.....î..

0100h: 00 3A 02 00 00 00 00 00 FD 9A 00 00 00 10 00 00ýš.....

0110h: 00 E0 00 00 00 00 40 00 00 10 00 00 00 02 00 00 .à.....@.....

0120h: 05 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00@.....

0130h: 00 40 03 00 00 04 00 00 7D 80 02 00 02 00 00 85 .@.....}€.....

0140h: 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00ú.3...

0150h: 00 00 00 00 10 00 00 00 F0 FA 00 00 33 00 00 00ô.È....x@..

0160h: D4 EC 00 00 C8 00 00 00 00 F0 02 00 78 40 00 00 Ô.È....x@..

0170h: 00 00 00 00 00 00 00 00 90 02 02 00 08 3F 01 00?..

0180h: 00 00 00 00 00 00 00 00 90 E2 00 00 1C 00 00 00â.....

0190h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@.....

Template Results - EXE.bt

Name	
DWORD LoaderFlags	0
DWORD NumberOfRvaAndSizes	16
struct IMAGE_DATA_DIRECTORIES DataDirectory	Patterns
struct DATA_DIR Export	
struct DATA_DIR Import	
struct DATA_DIR Resource	
struct DATA_DIR Exception	
struct DATA_DIR Security	Directory entry of the Attribute Certificate Table
DWORD VirtualAddress	20290h
DWORD Size	81672
struct DATA_DIR BaseRelocationTable	
struct DATA_DIR DebugDirectory	
struct DATA_DIR CopyrightOrArchitectureSpecificData	
struct DATA_DIR GlobalPtr	

You may find something wrong here if you read the image carefully.

The alert level of aid 41527 is about 50 fgt/day. Analysts tried to evaluate it but unfortunately the traffic collected was found to be useless due to the missing Attribute Certificate Table (checked in aid 38507) in them.

One day a customer complained that

MS.WinVerifyTrust.Signature.Validation.Remote.Code.Execution blocked an executable file (ticket 1379563) and provided us the download link for analysis. The file is about 333 MBs and a virus scan shows it is clean (malware sizes are almost never that large). So, this is a FP.

Find below the part of this file which matches aid 38507.

D960h:	2A CF 72 6F	22 B7 E3 6F	2A CF 72 6F	52 69 63 68	*ïro"-ão*ïroRich
D970h:	2B CF 72 6F	00 00 00 00	00 00 00 00	00 00 00 00	+ïro.....
D980h:	00 00 00 00	<u>50 45 00 00</u>	4C 01 04 00	36 8D AB 4CPE..L...6.«L
D990h:	00 00 00 00	00 00 00 00	E9 00 03 01	<u>0B 01 09 00</u>â.....
D9A0h:	00 F6 0D 00	00 24 08 00	00 00 00 00	<u>40 5C 09 00</u>	.ð...\$.....@\\..
D9B0h:	00 10 00 00	00 10 0E 00	00 00 40 00	00 10 00 00@.....
D9C0h:	00 02 00 00	05 00 00 00	00 00 00 00	05 00 00 00
D9D0h:	00 00 00 00	00 A0 16 00	00 04 00 00	24 32 EE 04\$2j.
D9E0h:	02 00 00 80	00 00 10 00	00 10 00 00	00 00 10 00	...€.....
D9F0h:	00 10 00 00	00 00 00 00	10 00 00 00	00 00 00 00
DA00h:	00 00 00 00	BC 06 11 00	DC 00 00 00	00 C0 11 004...Û....À..
DA10h:	<u>FC DF 04 00</u>	<u>00 00 00 00</u>	<u>00 00 00 00</u>	<u>60 73 ED 04</u>	üß.....`sí.
DA20h:	10 18 00 00	00 00 00 00	00 00 00 00	10 17 0E 00
DA30h:	1C 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
DA40h:	00 00 00 00	00 00 00 00	00 00 00 00	F0 57 0F 00ðW..
DA50h:	40 00 00 00	00 00 00 00	00 00 00 00	00 10 0E 00	@.....
DA60h:	38 06 00 00	08 05 11 00	40 00 00 00	00 00 00 00	8.....@.....
DA70h:	00 00 00 00	00 00 00 00	00 00 00 00	2E 74 65 78tex
DA80h:	74 00 00 00	2C F5 0D 00	00 10 00 00	00 F6 0D 00	t...,ð.....ð..

Patterns
Match
Aid 38507

The underlined 0x04ED7360 is the virtual address of the Attribute Certificate Table, not its size. Signature aid 38507 was checking the wrong field.

Another finding is that the size of the Attribute Certificate Table in this file is larger than 0x1000. We need to raise the bar a little bit to avoid hitting a large but normal certificate. To reduce the risk FP further, we can check the size of import and export tables to make sure it is PE32 option header.

Find below the improved aid 38507:

```
F-SBID( --vuln_id 31536; --attack_id 38507; --name
"MS.WinVerifyTrust.Signature.Validation.Remote.Code.Execution"; --group applications3; --
protocol tcp; --default_action pass; --revision 6639; --severity critical; --vuln_type "Other"; --app
Other; --os Windows; --status hidden; --file_type EXE; --pattern "PE|00 00|"; --context file; --
pattern "|0B 01|"; --context file; --distance 20; --within 2; --byte_test 4,<,256,90,relative,little; --
byte_test 4,<,0x1000,98,relative,little; --byte_test 4,<,0x1000,106,relative,little; --byte_test
4,>,0x2000,130,relative,little; --tag set,Tag.EXE.Signature; --tag quiet; --date 20150506; )
```

As a conclusion, Signature MS.WinVerifyTrust.Signature.Validation.Remote.Code.Execution had relatively low risk of FP. A signature mistake is found by analyzing the sample provided by the customer. The signature is improved accordingly to reduce the risk of FP.

9.4.2 Case study - MS.IE.MSXML.Object.Handling.Code.Execution

Signature MS.IE.MSXML.Object.Handling.Code.Execution was added as a generic signature to cover CVE-2012-1889:

“A memory corruption vulnerability exists in Microsoft XML Core Services. Specifically, when the definition method of the affected ActiveX controls is called with arbitrary parameter(s), the internal code would reference an uninitialized object and attempt to call a function pointer from its vtable, which could lead to arbitrary code execution.

”

Our signatures check the related GUIDs, ProgIDs and the ‘*definition*’ method call:

```
F-SBID( --vuln_id 32238; --attack_id 39279; --name
"MS.IE.MSXML.Object.Handling.Code.Execution"; --group applications2; --protocol tcp; --
default_action pass; --revision 4356; --severity critical; --app IE; --os Windows; --status hidden; -
-service HTTP; --flow from_server; --pattern "Microsoft.XMLDOM"; --context body; --no_case; --
pcre "/(Activex|Create)Object/i"; --context body; --distance -40; --within 25; --tag quiet; --tag
set,Tag.MSXML.ActiveX.Control.Access.New; --date 20140423; )
F-SBID( --vuln_id 32238; --attack_id 41507; --name
"MS.IE.MSXML.Object.Handling.Code.Execution"; --group applications2; --protocol tcp; --
default_action pass; --revision 4401; --severity critical; --app IE; --os Windows; --status disable; -
-service HTTP; --flow from_server; --pattern ".definition"; --context body; --no_case; --pcre
"^\\.definition(\\s[a-zA-Z0-9]+|\\()/i"; --context body; --distance -11; --within 16; --tag
test,Tag.MSXML.ActiveX.Control.Access.New; --date 20140423; )
Signatures checking GUID are omitted for brevity.
```

Aid 41507 was developed as a generic signature and was disabled by default due to its high alert level (2000+ fgt/day with disabled by default).

We have been receiving FP complaints since this signature was released. Unfortunately, in most cases customers failed to provide us with a useful traffic capture, thus it ended up unresolved.

Obviously, this signature hit something popular on the Internet. If we find out what it is and avoid it, the FP risk can be dramatically reduced.

During evaluation we got 23 useful traffics. Most of them seem related to JQuery and LESS libraries.

JQuery library:

```
$author.tokenInput(gdn.url('/user/tagsearch'), {
    hintText: gdn.definition("TagHint", "Start to type...")
...//omitted for brevity

$('.AdvancedSearch .AdvancedWrap').hide();

});
})(jQuery, window, document);
```

LESS library:

```

primary: function() {
    for (var a, b = this.mixin, c = J, d = []; C;) {
        if (a = this.extendRule() || b.definition() || this.rule() || this.ruleset() || b.call() ||
this.comment() || this.rulesetCall() || this.directive()) d.push(a);
        else if (!c(/^[^\n]+/) && !c(/^[^;]+/)) break;
    }
}

```

The 'definition' method calls are highlighted. No GUIDs or ProgIDs patterns can be found in these traffics. With these fragments we can find some files on the Internet. For example, <http://dev.wikia.com/wiki/Less/less.js>

You will find these files only contain the pattern ". definition" but still no GUIDs/ProgIDs. So, the tag Tag.MSXML.ActiveX.Control.Access.New must be set when scanning other files in the same session.

We can fix these FPs by checking both GUIDs/ProgIDs and "definition" method call in the same signature. This will introduce some FN but comparing it to reducing the FP associated with the signature, it is still worth it.

```

F-SBID( --vuln_id 32238; --attack_id 50585; --name
"MS.IE.MSXML.Object.Handling.Code.Execution"; --group applications2; --protocol tcp; --
default_action pass; --revision 0; --severity critical; --app IE; --os Windows; --service HTTP; --
flow from_server; --pattern "Microsoft.XMLDOM"; --context body; --no_case; --pcre
"/(Activex|Create)Object/i"; --context body; --distance -40; --within 25; --pattern ".definition"; --
context body; --no_case; --pcre "/^\.definition(\s\w+|\\()/i"; --context body; --distance -11; --within
15; --date 20140423; )

```

As a conclusion, signature MS.IE.MSXML.Object.Handling.Code.Execution had high risk of FP. During evaluation we found it hit some popular JS libraries. We fixed these FPs by checking both GUIDs/ProgIDs and "definition" method calls in the same signature.

9.5 Process for Signature Quality Control

To improve the control of signature quality, and to avoid releasing bad signatures to our customers, we have the following processes for signature creation and modification.

9.5.1 New Regular Signature Creation

All new regular signatures (IPS and AppCtrl in category Botnet and Proxy) must go through the **beta process** before their official release. When a new regular signature is created, **ISTAME** automatically sets it to "**beta**". The signature will be assigned to the author for review if its alert reaches a predefined threshold (refer 9.1 for details). Analysts should evaluate IPS signature that triggers more than **5 FGTs/day**, or the review team may reject it. All the IPS signatures triggering more than **10 FGTs/day** must be evaluated.

After the beta process, or possibly longer period of alert monitoring and evaluation, a decision

will be made as to whether its action needs to be changed. This is based on the alert level, its detection conditions and the evaluation result if available. If the signature potentially has a high rate of False Positives, the analyst should set its action to **"disable"**, even though that is only available when it becomes a public signature.

9.5.2 Public Signature Modification

If there is a need to modify a public signature (signatures that have been released), an analyst should instead leave it unchanged and create a new (beta) signature to replace it. This can be done in the signature edit page (Figure 8.5.2).

The screenshot shows the FortiWeb signature edit interface. At the top, there are tabs for Vulnerabilities, Signatures, Rule Files, Descriptions, PCAPs, Documents, Lists, Icons, FortiWeb, and Account. Below the tabs is a status bar with options: To Review, This Signature Replaces (Multi)..., Is pending, Is urgent, Is disable, Is hidden, Is forced, and Is evaluate. A dropdown menu shows 'Firmware list' with options: 3.0, 4.0, 4.3, 5.0, 5.0 Extended, and 5.0 Legacy. The 'Name' field contains 'MiniO Inherit Access Key Privilege Escalation' with a '(BETA)' tag. The 'Body' field contains a signature definition: `--service http; --parsed_type http_post; --flow from_client; --pattern "/minio/admin/v3/update-service-account"; --context uri; --within 100,context; --no_case; --pattern "accessKey="; --context uri; --distance 0; --no_case;`. The 'Comment' field contains a detailed description of the vulnerability and a link to the exploit-db entry. A 'Submit' button is at the bottom left. At the bottom of the page, the signature definition is repeated in a highlighted box.

Figure 8.5.2

Analysts may want to replace signatures for only some of the platforms in some cases. They can do that by only setting the flag of platforms needed. For example, after the engine team fixed a signature attribute of `--file_type VIDEO` in 2.1.35 engine which recognizes most of mp4, mov type of files that were only recognized by `--file_type UNKNOWN` in the past. Since the fix is only done in 2.1 engine, 1.0 engine that is used in most of 4.0 platforms will not be affected by this enhancement. In this case, analysts need to clear the 4.0 flag for the new signature to make sure the old one for 4.0 platform is not abandoned.

9.5.3 Hidden (tag-set) Signatures

Their default **status** and **action** must be **"hidden"** and **"pass"**, and the signature body must contain **"--tag quiet;"**.

Hidden signatures do not go through the beta process as their triggers are not visible to the customer. On the other hand, the body of tag-set signature is part of tag-test signature, so any modification of tag-set changes the body of tag-test signature. And this modification does not go through beta process and has caused serious FP issues in the past. Following processes are

introduced to control modification of tag-set (both private tag and public tag) signature:

1. Analysts can modify a tag-set signature if the corresponding tag-test signature is still in beta
2. Analysts should avoid modifying a tag-set signature after the corresponding tag-test signature becomes public. Analysts should write a new set of tag-set/tag-test signatures to replace the old one and different tag names should be used. Exceptions can be made in cases where analysts are confident that it won't cause FP issues, e.g. VPN signatures, etc.

Using signature MS.Windows.TCP.FIN.WAIT.DoS as an example:

Old signatures:

```
F-SBID( --vuln_id 34674; --attack_id 37237; --name "MS.Windows.TCP.FIN.WAIT.DoS"; --tag quiet; --tag set,MS.Windows.TCP.FIN.WAIT.DoS; --keyword ###;)
```

```
F-SBID( --vuln_id 34674; --attack_id 37238; --name "MS.Windows.TCP.FIN.WAIT.DoS"; --tag test,MS.Windows.TCP.FIN.WAIT.DoS; --keyword ###; )
```

To modify attack_id 37237 after attack_id 37238 is publicly released, analyst should write a new set of tag-set/tag-test signatures:

```
F-SBID( --vuln_id 34674; --attack_id 37239; --name "MS.Windows.TCP.FIN.WAIT.DoS"; --tag quiet; --tag set,MS.Windows.TCP.FIN.WAIT.DoS.A; --keyword $$$;)
```

```
F-SBID( --vuln_id 34674; --attack_id 37240; --name "MS.Windows.TCP.FIN.WAIT.DoS"; --tag test,MS.Windows.TCP.FIN.WAIT.DoS.A; --keyword ###;)
```

As you can see, the body of tag-set signature (attack_id 37239) is modified and a new tag name is used. The only change of the tag-test signature (attack_id 37240) is the tag name. Analyst should set attack_id 37240 to replace attack_id 37238 which will be automatically abandoned when attack_id 37240 is publicly released. Attack_id 37237 will be abandoned by QA script later.

A hidden signature cannot be changed to a regular signature and vice versa, even if the signature was never released.

9.5.4 skip-after Signatures

IPS ISB setting was removed in FortiOS 5.2 and a new option skip-after is introduced in engine 3.38 for analysts to change the ISB setting. Following process is introduced to control the risk of possible performance impact:

1. If the alert level is unknown analysts should test the skip-after signature (remove options skip-after, quiet and tag if applicable) under vid **27840** (App.Control.Signature.Test) for at least 2 days.
2. The alert level needs to be below 1000 fgt/day **AND** 1000 triggers/fgt. If not, analysts can

- either improve the signature or select a skip-after value below 1M.
- Analysts need to leave a comment like below when adding the skip-after signature:
"Tested as aid 12345. Alert level is about 100 fgt/day and 100 triggers/fgt during the test."
 IPS QA will check the comment to make sure the signature has been tested before releasing it.

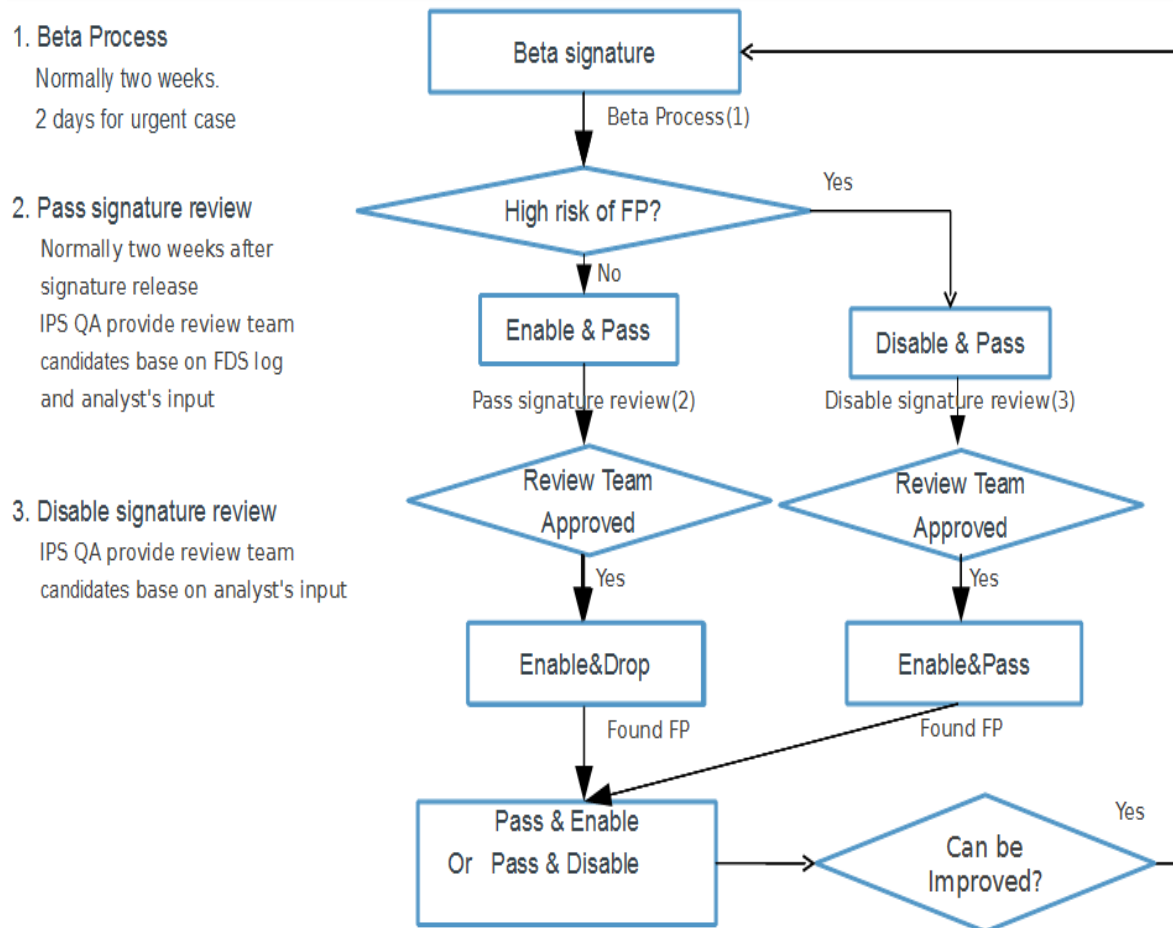
Note.

- It is recommended to use a skip-after value below 10M. Scanning the whole session (skip-after 0) should be avoided.
- AppCtrl skip-after signature is **NOT** applicable to this process. But for the top popular applications (above 10000 fgt/day and 10000 triggers/fgt) it is recommended to use a skip-after below 10M.
- Engine will print something like below when skip-after signature triggers:

[6932/0]ips_opt_skip_after_handler: changing ignore_after_size from 204800 to 1048576

9.5.5 Signature Status and Action Change

Figure below shows the main flow of signature status and action change.



More details can be found in the next chapter.

9.5.6 Pre-release notification of AppCtrl package

Many AppCtrl profiles are based on the category. Changing the category of an existing application or adding a new application effectively modifies these profiles, which may cause a disruption in our customer's network. We received quite a few complaints caused by this kind of update in the past, so we introduced the pre-release notification into the AppCtrl package release process in 2018.

The following changes require a pre-release notification:

1. Changing of category of an existing application
2. Adding a new application (new vid) into the category "Network.Service", "Email", "Storage.Backup", "Collaboration" and "Business".
3. Adding a new application which is expected to be very popular (>10k FGT/day).

Only the team lead has the privilege to release a pre-release notification which comprises the following steps:

1. Prepare the pre-release notification file
2. Manually push out the description of the new application
3. Upload the pre-release notification file onto the Fortiguard website

9.5.7 Signature Pullout & Re-enable

We currently have three types of IPS DBs:

Regular DB: This is the smallest IPS DB which contains the most active and useful IPS signatures. It is designed for customers who prefer performance over security. IPS QA regularly pulls out inactive signatures from this DB and re-enables active ones to make sure it contains only the most useful signatures.

Full Extended DB: This is the largest IPS DB which contains literally all the IPS signatures. It is designed for customers who prefer security over performance.

Slim Extended DB: This is the slim version of the full extended DB which is designed for customers who prefer security over performance, but their FGTs, for example, CP8 models, are no longer able to handle all the IPS signatures.

Figure below shows the main flow of the signature pullout and re-enable in the regular and slim extended DB.

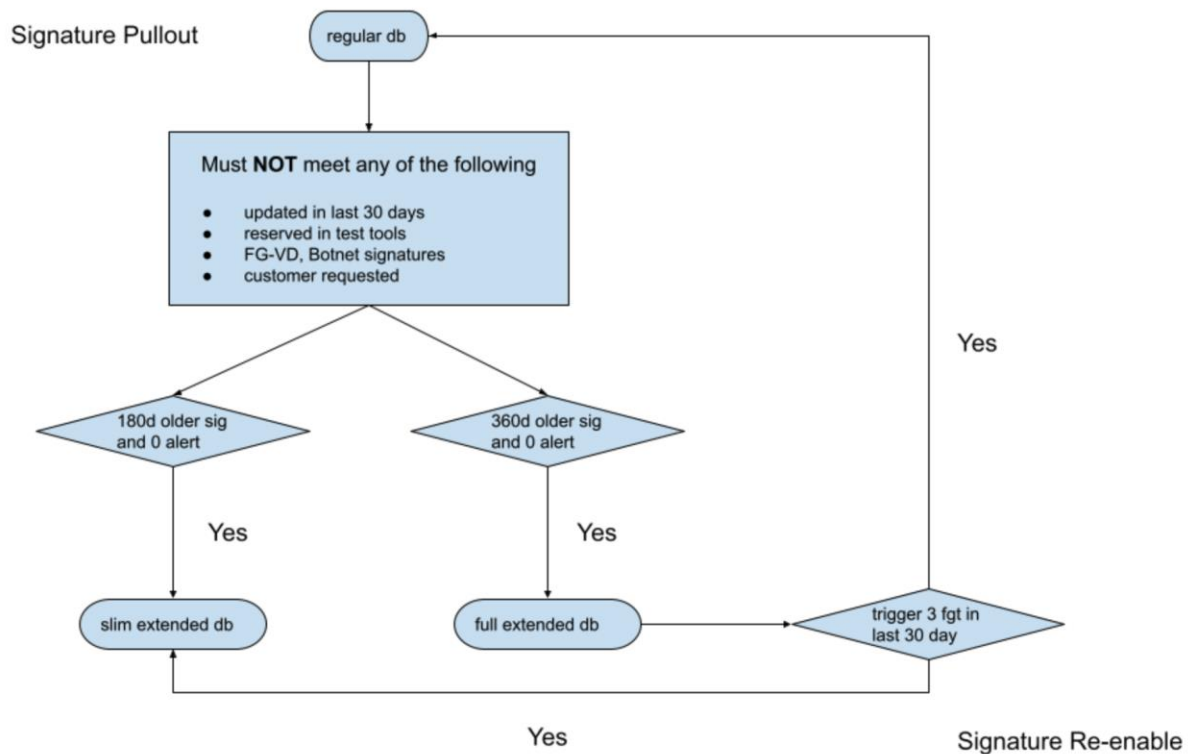


Figure below shows the signature comments automatically added by QA's script when it happens.

MoinMoin.Twikidraw.Action.Traversal.File.Upload

```
--service HTTP; --flow from_client; --pattern "action=twikidraw"; --no_
context uri; --no_case; --distance -300; --within 600; --pattern "../..
```

```
Enable 5.0, 5.0 Legacy due to activity in last 30days
Remove from slim extended db, mantis 756371.
Auto Pull-out inactive signature for 4.3
```

10. Systems for Quality Assurance

10.1 Pcap Management System (PMS)

[PMS](#) is developed by IPS QA for quality assurance purposes. Following see the most frequently used modules by analyst:

- Signature Quality Tracker

This module has two main functions: Control change of signature status/action and handle False Positive (FP). More details will be discussed later.

- Test Summary

This module provides access to weekly test results of supported tools including Trend Micro, BP, Mu, and Metasploit. Analysts can easily find uncovered exploits here.

- Search Pcap

You can find all the traffic covered by a particular signature here.

10.1.1 Review high alert beta signature

When the alert level of a beta signature reaches a predefined threshold (10 or more Fortigates or generates 20 or more alerts within a week at time of writing), it will be categorized as a high alert beta signature. For such signatures, QA will send an email to the analyst that the signature belongs to, requesting the analyst to review it.

The analyst should then decide on the following and update the signature's status on PMS accordingly.

- Is the signature's quality good? (Low risk of FP)
- Can it be further improved?

If the signature quality is good and the alert rate is within an acceptable range. Then the analyst can set the signature status to "Signature Can be Released". See image below for examples.

If the signature quality isn't good, and the signature can be improved, then it should be improved, and status should be set to "fixed".

If the analyst is unsure of the signature quality or as for how it can be further improved, he/she can set the signature's evaluate tag to obtain traffic submission for verification. The signature status on PMS should be set to "Signature Set to Evaluate" until the analyst is certain of what to do for the signature.

If the signature is too difficult to improve and it has a high alert rate, it should be disabled. The analyst can set the signature's status on PMS to "Signature Can Be Released" but add a note that the signature has been disabled.

Note: Not all high alert signatures can be disabled. If the signature's alert rate is very high, it should not be disabled unless approved by the review team. If it cannot be disabled, then the

analyst should simply abandon the signature.

Count on FGT	101
Count in Triggering	2959
Activity	29.29702
Trigger on PMS	0

Other Signatures on this Vulnerability

Attack ID	Sig Status	Sig Action	Revision	Fgt Count	Trigger Count	Activity
27381	enable	drop	3129	87	334	3.839080

Analysts Reviews

Note: 1. **Set to 'Signature Can be Released'**: Signature can be released ;
2. **Set to 'Signature Fixed'**: Signature has been abandoned or improved ;
3. **Set to 'Signature Assigned'**: Signature has been assigned to a specific user ;
4. **Set to 'Signature Set to Evaluate'**: Signature has been set to evaluated.

Set review status:	<div>Signature Assigned</div>	to	Owner:	<div>palanceng</div>
Add a comment:	<div>Signature Assigned Signature Can be Released Signature Fixed Signature Set to Evaluate</div>			
<div>Submit</div>				

Analysts - Comments

Comment Time	Status Desc	User ID	User Comment
2012-10-30 16:12:44	Signature Assigned	auto_qa	some alerts, forward to the owner to review

Review Team Approval

Set review status:	<div>Not Reviewed</div>
--------------------	-------------------------

10.1.2 Handle False Positive

The QA team has a collection of traffic which is collected from our corporate network. These clean traffics are used in testing IPS signatures along with normal traffic from attack tools such as Trend Micro. When a signature triggers such traffic, QA will request an analyst, most of the time it is to whom the signature belongs to, to review the traffic and signature.

In such cases, the analyst should first determine the following:

- Is the traffic clean?
- Is it triggering a disabled decoder signature?

Although the traffic that the QA uses is supposed to be clean, it is not always the case due to various reasons such testing, research, etc. from within the network. Therefore, if the traffic is found to be an attack, then the analyst can mark the FP issue as "Wrong Traffic" and leave a comment on the submit page informing the QA about it. See images below for details.

If the signature is triggering a disabled decoder signature, then the FP issue can be marked as “disabled_decoder”. This is due to decoder signature being an engine side issue.

If the traffic is indeed clean traffic, then the analyst should then determine if the triggered signature can be improved.

If the signature cannot be improved, either due to it being too difficult, or to improve the signature requires support from the engine, then the issue can be marked as “too difficult” and “engine_support” respectively with the explanation given under comments.

If the signature can be improved, then the analyst can simply improve the signature and mark the issue on PMS as “sig_fixed”.

Attack ID	Sig Status	Sig Action	Revision	Fgt Count	Trigger Count	Reviewed Status	Reviewed by	User Action
34027	beta	pass	3198	767	91604	checkout	palanceng	<input type="checkbox"/> too_difficult <input type="checkbox"/> wrong_traffic <input type="checkbox"/> sig_fixed <input type="checkbox"/> engine_support <input type="checkbox"/> disabled_decoder

Name	Tftpd32.DNS.Server.Buffer.Overflow
Attack ID	34027
Vuln ID	32145
Signature Status	beta
Default Action	pass
Revision	3198
Count on FGT	767
Count in Triggering	91604
Activity	119,4315

Add a comment for pcap id **48543** with signature aid 34027 and rev 3198

Currently, this pcap status was set to sig fixed.

Comment	
<input type="button" value="Submit"/>	

Comment History

Comment Time	Status Desc	User ID	User Comment
2012-06-11 10:40:36	Fixed Signature	palanceng	Already improved sig to fix FP issues

10.1.3 Change the default action to drop

To effectively protect customers, we should set the default action of our signature to drop if possible. But to reduce the risk of blocking customer normal traffic accidentally, an internal process is set up to control signature action change.

Process of changing default action of signature to drop:

1. Analyst send IPS QA the signature list that he suggests setting to drop
2. QA put these signatures in review queue and notify review team
3. Review team make the decisions
4. IPS QA enforce review result

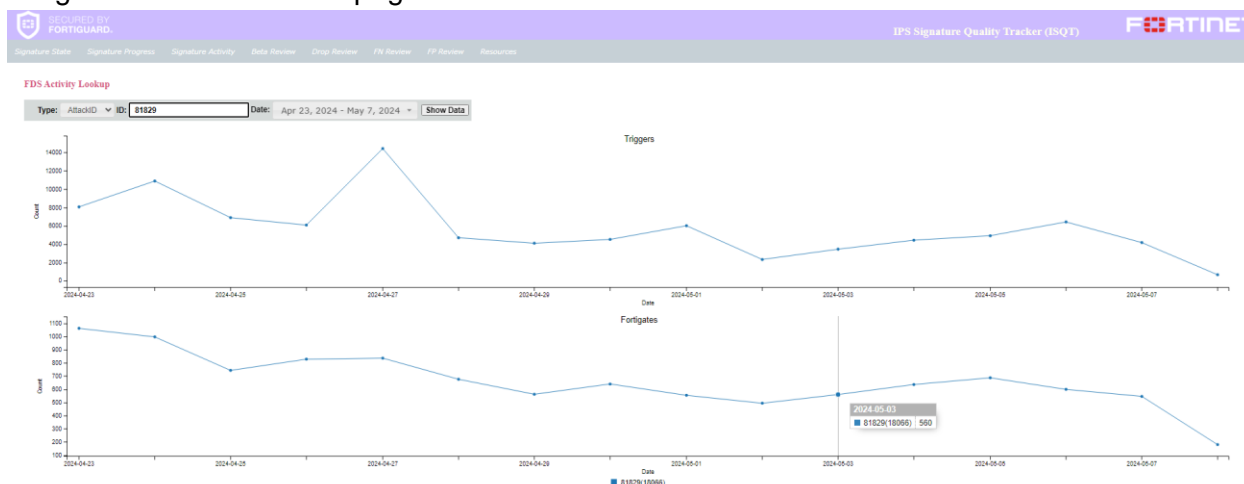
To keep increasing our default block rate, analysts are responsible for submitting action change requests for his signature. Since signatures in a report have the same default action, analysts should take other signatures into consideration when adding a signature into an existing report. For example, an analyst should avoid adding a signature into a report with drop action if he knows the new signature has high risk of FP.

10.2 FRS

The [FDS Lookup](#) system is used by analysts, QA and the review team for quality assurance purposes. It provides two features to IPS analysts: FDS statistics and evaluation.

It allows an analyst to view the statistics for a particular signature. It shows detailed information such as the number of Fortigate triggers, engine version that the triggered Fortigates are using, signature's revision, date of the triggers, and total alert reported for that engine version.

This information allows the analyst to determine if the signature is generating high alerts, has unusual trigger count, or if the signature should be disabled, evaluated. The following is an image of the FDS Statistic page.



10.2.1 Evaluate a signature

The [evaluation](#) system provides analysts access to evaluation traffic. Following is an image of the evaluation feature of the FRS system.

Evaluated Pcaps for Last 3 Days

Signatures Triggered Today:

47162	50657	53244	53262	62434	67736	74419	74998	75000	75962	76484	76573	76743	76808	83240	83241	84909	86886	90077	90137	91510
92536	93071	93730	94221	94222	94340	94495	95030	95032	95048	95054	95060	95074	95075	95170	95194	95197	95211	95226	95279	95406
95444	95474	95512	95533	95561	95594	95660	95673	ALL												

Show

100

entries

Search:

MD5	Vuln ID	Sig ID	Rev	Engine	Entry Time	Serial No.
56512b1911e73ba0822d04b8ad8cb0ca [pcap bin]	41127	53244	6733	7002	2024-05-08 13:40:01	F680003916801247
9a75ddaef69541017c511675883fc8380 [pcap bin]	41127	53244	6733	7004	2024-05-08 12:51:01	FGT81ETK19003197
76207a02c8e41bdcea58cc189f6e59b [pcap bin]	41127	53244	6733	7002	2024-05-08 12:40:02	F680003916801247
ef5c832a52389840b6215977430e56c3 [pcap bin]	41127	53244	6733	7002	2024-05-08 11:20:01	F680003916801247
3d69b4593f897ec11d6406f17193160b [pcap bin]	41127	53244	6733	7004	2024-05-08 10:54:01	FGT40FTK2209AECQ
05de0413242db63b1efff162d0fadb0eb9 [pcap bin]	41127	53244	6733	7002	2024-05-08 10:30:02	F680003916801247
6970bc35351bf8717448983b4a085c395 [pcap bin]	41127	53244	6733	7002	2024-05-08 10:10:01	F680003916801247
4c3ac5c3f3f2f45ac68b90fe7465464a [pcap bin]	41127	53244	6733	7002	2024-05-08 09:40:02	F680003916801247
de8539535e48c76d61b377d363380311 [pcap bin]	41127	53244	6733	7002	2024-05-08	F680003916801247

The default page will show a list of signatures triggered. From here, the analyst can opt to filter the search by vuln_id, attack_id, revision, etc.

Analyst should record the evaluation result on ISTANCE as signature comment. This information is very useful for IPS QA, other analysts and reviewers. If the analyst finds a FP but decides not to improve the signature, a reason should be given, and the FP traffic should be uploaded to ISTANCE as shown in following image.

Attachment purpose: clean traffic ▼

File name: Choose File No file chosen

Comment:

FP traffic collected during evaluate aid 12345|

Test source list:



Submit