Performance Rules Creation

Part 2: Rules Options and Techniques

SOURCEfire

ENTERPRISE THREAT MANAGEMENT

What madness today?

- Learn by reviewing actual VRT published rules
- Highlight potential issues with certain rule options
- Break down some common rule constructs
  - Buffer overflow detection
  - Protocol decoding
- Abuse a VRT team member with the “replace” functionality

What is a DQOH?

Matthew Olney

(irc nick: dqoh)

VRT Security Analyst for 2 ½ years

Primary Responsibilities:

- Snort rules generation
- QA for SEU and VRT rules feed
- Agent of Karma

Past life:

- Network and Security Engineer
  - Cisco
  - Snort
  - Open source security products

VRT Rule Examples

SOURCEfire

ENTERPRISE THREAT MANAGEMENT

Strap In, the Bus is Leaving

- This topic could be a multi-day course (and, in some cases it is)
- So I'm assuming:
  - You've seen rules and know generally how they are laid out
  - You can reference the Snort Users Manual for general rules question (I'll cite sections as appropriate)
- And I'm providing:
  - Non-obvious information on rules options
  - Usage cases from real snort rules
  - Information on rules options as they occur
- Don't Panic

Example 1: Content based buffer overflow checks

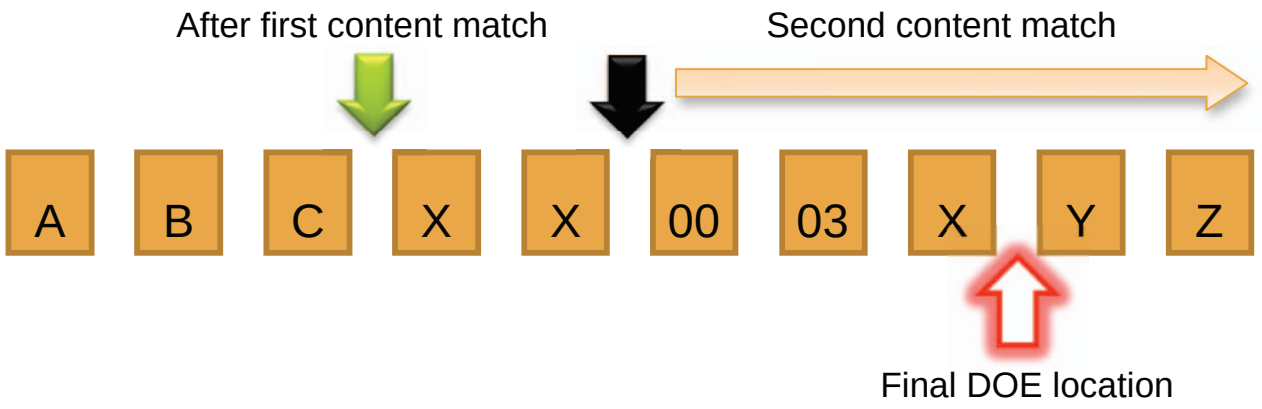
- alert tcp $EXTERNAL_NET any -> $HOME_NET 4000 (msg:"EXPLOIT Alt-N SecurityGateway username buffer overflow attempt"; flow:established, to_server; content:"username="; nocase; isdataat:450,relative; content:!"&"; within:450; content:!"|0A|"; within:450; metadata:policy balanced-ips drop, policy connectivity-ips drop, policy security-ips drop; reference:url,secunia.com/advisories/30497/; classtype:attempted-admin; sid:13916; rev:2;)

Bus Stop: Content Option

- Content can be modified as non-relative:
  - Content:"A"; depth: 3; offset: 2;
  - Move 2 bytes into the payload and look for "A" within the next 3 bytes.
- Content can be modified as relative:
  - Relative matches are made from the DOE pointer (Usually the end of the previous match)
  - Content:"A"; Content: "B"; distance: 4; within: 5;
  - Find "A". Then move 4 bytes from the end of "A" and find "B" within the next 5 bytes.
- Content can be negative (that is alert if this isn't seen):
  - Content:!"A"; depth: 3; offset: 2;
- Content can be made case insensitive using the nocase; modifier.
- Check out sections 3.5.1 – 3.5.7 in the Snort Users Manual for more information.

Bus Stop: DOE & relative content example

- Relative contents use the DOE pointer
  - Content:”ABC”;
    - Places DOE after C
  - Content: “X”; distance: 2; within: 5;
    - Moves the pointer 2 bytes (First black arrow)
    - Looks at the next 5 bytes for an “X” (Orange arrow)
    - Places the DOE pointer after the X (red arrow)



Example 1: Buffer Overflow Detection (sid: 13916)

(flow:established, to_server; content:"username="; nocase; isdataat:450,relative; content:!"&"; within:450; content:!"|0A|"; within:450;)

- flow: established, to_server;
  - Used to reduce false positives and improve performance.
  - Flow is fairly straight forward, but check out Section 3.6.9 for full details
- content:”username=“;
  - Has the “nocase” modification to allow for any type of match
  - pattern can be anywhere in the payload.
- This pattern match will be the anchor for the rest of our detection

Example 1: Isdataat usage…

(flow:established, to_server; content:"username="; nocase; isdataat:450,relative; content:!"&"; within:450; content:!"|0A|"; within:450;)

- For content:!””; checks involving buffer overflows, we need to make sure the data is there to be checked.
  - Negative content matches look for the absence of data, so even if we run out of data, we still are successful
  - isdataat: verifies that there is data within the specified distance.
  - The check can be relative or non-relative
- Format is: isdataat:<int>[,relative];
  - Check out section 3.5.12 for more information on isdataat:
- isdataat: 450, relative;
  - The anchor (username=) is not at a fixed location, so we must make the size check relative to this match
  - Relative keywords makes the 450 byte check from the DOE.

Example 1, continued

(flow:established, to_server; content:"username="; nocase; isdataat:450,relative; content:!"&"; within:450; content:!"|0A|"; within:450;)

- We have now verified that:
  - The traffic is directed to a server, and the TCP session is established
  - The string “username=“ is in the payload
  - That there is sufficient space for the attack to be delivered
- For this protocol on this server, there are two terminating characters, “&” and line feed (LF) \x0A. We need to check that neither occur within the next 450 bytes:
  - content:!"&"; within:450;
  - content:!"|0A|"; within:450;
- If we, from the anchor point (username=), have 450 bytes available, and we don’t reach any terminating characters, then we will alert

Bus Stop: dsize

- dsize tests the payload size
  - format: dsize: [<>] <number> [<><number];
  - dsize: 300<>400
- You might be tempted to use dsize, rather than isdataat as a size test for buffer overflows. (Spoiler: Don't do this)
  - dsize automatically bails on any packet that is part of a reassembled stream.
  - This leads to a false-negative situation for certain buffer overflow attacks delivered over more than one packet.
  - Dsize does not handle relative checks
- dsize is designed to test abnormally sized packets, and isdataat should be used for all other purposes.

Example 2: PCRE buffer overflow checks

- alert tcp $EXTERNAL_NET any -> $HOME_NET 13782 (msg:"EXPLOIT Symantec NetBackup connect_options buffer overflow attempt"; flow:established,to_server; content:"CONNECT_OPTIONS="; nocase; isdataat:900,relative; pcre:"/CONNECT_OPTIONS\x3D[^\x20\x0A\x0D\x00]{900}/smi"; reference:bugtraq,21565; reference:cve,2006-6822; classtype:attempted-admin; sid:9813; rev:2;)

Example 2: Buffer Overflow Detection (sid: 9813)

(flow:established,to_server; content:"CONNECT_OPTIONS="; nocase; isdataat:900,relative; pcre:"/CONNECT_OPTIONS\x3D[^\x20\x0A\x0D\x00]{900}/smi";)

- content:"CONNECT_OPTIONS="; nocase;
  - nocase argument indicates that the pattern can be matched in any combination of either lower or upper case characters.
- isdatat:900,relative;
  - Again, staring at the DOE, verify that there are 900 bytes available for detection
  - Note, this is not required for PCRE as it is for negative content checks.
    - Content:!"A"; within: 40; checks for the absence of an A in 40 bytes, or end of packet.
    - pcre:"/[^A]{40}/"; looks for 40 consecutive characters that are not "A".
  - This check provides for speed, in the case where there is not sufficient payload left for the buffer overflow attack to happen, we bail on this check, rather than calling PCRE for no reason.
    - Always find ways to bail before running a PCRE
- Now, about that PCRE (remember, don't panic…)

Example 2, continued

pcre:"/CONNECT_OPTIONS\x3D[^\x20\x0A\x0D\x00]{900}/smi";)

- This is actually a fairly straight forward example of pcre:
- The format is pcre:"/REGEX/modifiers";
- The \x statement means you are providing a hexadecimal number to check
  - \x3d is the ascii code for "="
  - \x20 is the ascii code for space
  - \x0a is the ascii code for line feed
  - \x0d is the ascii code for carriage return
  - \x00 is the null byte
- The [^\x20\x0A\x0d\x00] is a character class declaration, and the '^' means not. That is, PCRE is instructed to match on characters that are not in this class.
- The {900} means do this match 900 times, so find 900 characters that are not in the character class.

Example 2, final notes

- /smi
  - i = case insensitive search
  - s = include newlines in the dot (.) metacharacter
  - m = metacharacters ^ and $ can match before and after a newline, as well as the beginning and the end of the buffer
  - Check 3.5.13 for more modifiers
- PCRE can do some powerful things
  - And computationally expensive
  - And easy to mess up
  - Make sure you run it only when you have to
  - Test your pcre (pcretest) and optimize.
  - Then do it again.

Example 3: Buffer overflow detection with byte_test

- alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"RPC snmpXdmi overflow attempt TCP"; flow:to_server,established; content:"|00 01 87 99|"; depth:4; offset:16; content:"|00 00 01 01|"; within:4; distance:4; byte_jump:4,4,relative,align; byte_jump:4,4,relative,align; byte_test:4,>,1024,20,relative; content:"|00 00 00 00|"; depth:4; offset:8; metadata:policy balanced-ips drop, policy security-ips drop, service sunrpc; reference:bugtraq,2417; reference:cve,2001-0236; reference:nessus,10659; reference:url,www.cert.org/advisories/CA-2001-05.html; classtype:attempted-admin; sid:569; rev:18;)

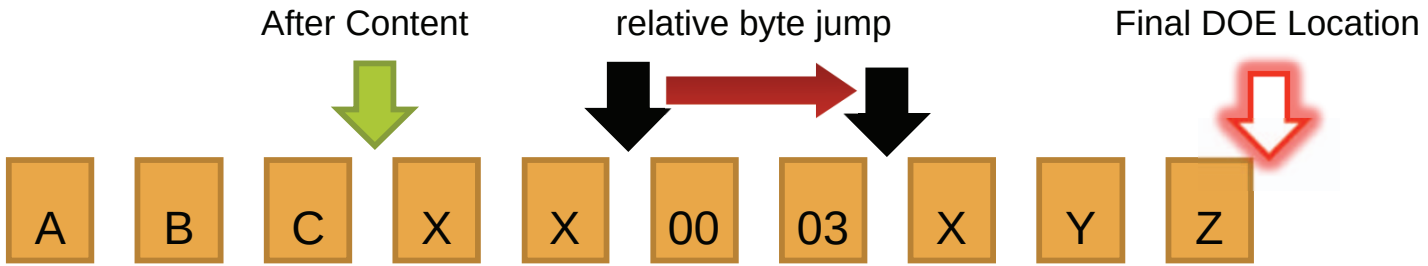
Example 3: Content and the fast pattern matcher

flow:to_server,established; content:"|00 01 87 99|"; depth:4; offset:16; content:"|00 00 01 01|"; within:4; distance:4; byte_jump:4,4,relative,align; byte_jump:4,4,relative,align; byte_test:4,>,1024,20,relative; content:"|00 00 00 00|"; depth:4; offset:8;

- Three four-byte patterns are in this rule.
  - The rule writer chose a specific sequence to speed up detection.
- content:"|00 01 87 99|"; depth:4; offset:16;
  - Absolute 4 byte match, placed at beginning for more unique "longest, first match"
- content:"|00 00 01 01|"; within:4; distance:4;
  - Relative to the previous match. Note that this could also have been depth: 4; offset: 24; instead.
  - Will act as the anchor for our other detection
- content:"|00 00 00 00|"; depth:4; offset:8;
  - Will be the final validation that our packet is an attack.

Bus Stop: byte_jump

- Usage:
  - byte_jump: <bytes_to_convert>, <offset> [,relative] [,multiplier <multiplier value>] [,big] [,little][,string][,hex] [,dec] [,oct] [,align] [,from_beginning];
  - Check out 3.5.15 for details.
- By way of example:
  - Content:"ABC";
    - Places DOE after C
  - Byte_Jump: 2, 2, relative;
    - Moves the pointer 2 bytes (First black arrow)
    - Reads two bytes
    - Jumps from the end of the read (Second black arrow)



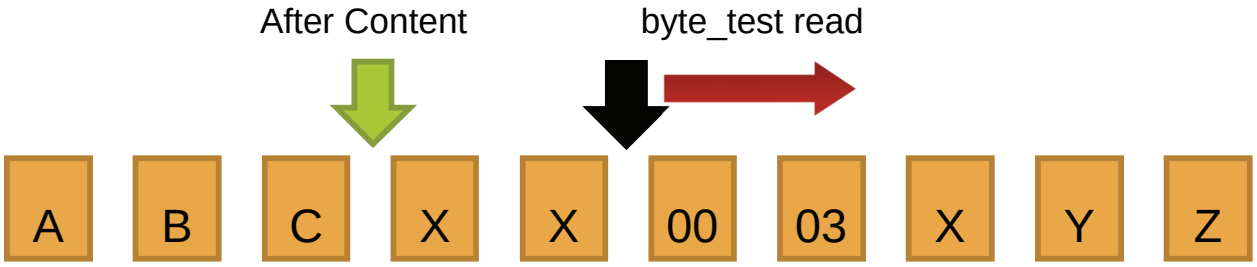
Example 3: Protocol parsing with byte_jump

byte_jump:4,4,relative,align; byte_jump:4,4,relative,align;

- Byte jump is often used to parse length encoded data.
- Here we have two dynamically sized data fields we need to jump over to find the data we need (the same byte_jump structure is used for both data blocks):
  - 4 bytes are read starting 4 bytes from the DOE.
  - This protocol requires that data be stored on 4-byte boundaries. The 'align' keyword tells snort to round jumps as necessary to handle this.
  - The DOE is then moved the calculated number of bytes
  - This process is repeated to jump over the second dynamically sized data field.
- By decoding the protocol we are now 20 bytes from a 4 byte size field that declares how large the target field is.

Bus Stop: byte_test

- Usage:
  - byte_test: <bytes to convert>, [!]<operator>, <value>, <offset> [,relative] [,<endian>] [,<number type>, string];
  - Section 3.5.14 details the byte_test option.
- By way of example:
  - Content:"ABC";
    - Places DOE after C
  - Byte_test: 2, <, 4, 2, relative;
    - Moves the pointer 2 bytes (First black arrow)
    - Reads two bytes
    - If the byte read is less than four, then the check is passed.
    - Note: The DOE does not move.



Example 3: Detecting the overflow

byte_test:4,>,1024,20,relative;

- Using anchored content matches and a sequence of byte_jumps, we now know we are 20 bytes from the target size field.
- Our research shows that if that field is greater than 1024, then the provided data will overflow a buffer in memory.
- We use the above byte test to make the check:
  - Read 4 bytes as a number, starting 20 bytes from the DOE
  - If that number is greater than 1024, an attack most likely is underway

Example 4: Kaminsky DNS Bug detection

- alert udp $EXTERNAL_NET 53 -> $HOME_NET any (msg:"DNS large number of NXDOMAIN replies - possible DNS cache poisoning"; byte_test:1,&,2,3; byte_test:1,&,1,3; byte_test:1,&,128,2; threshold:type threshold, track by_src, count 200, seconds 30; metadata:policy balanced-ips alert, policy security-ips alert, service dns; reference:cve,2008-1447; reference:url,www.kb.cert.org/vuls/id/800113; classtype:misc-attack; sid:13948; rev:2;)

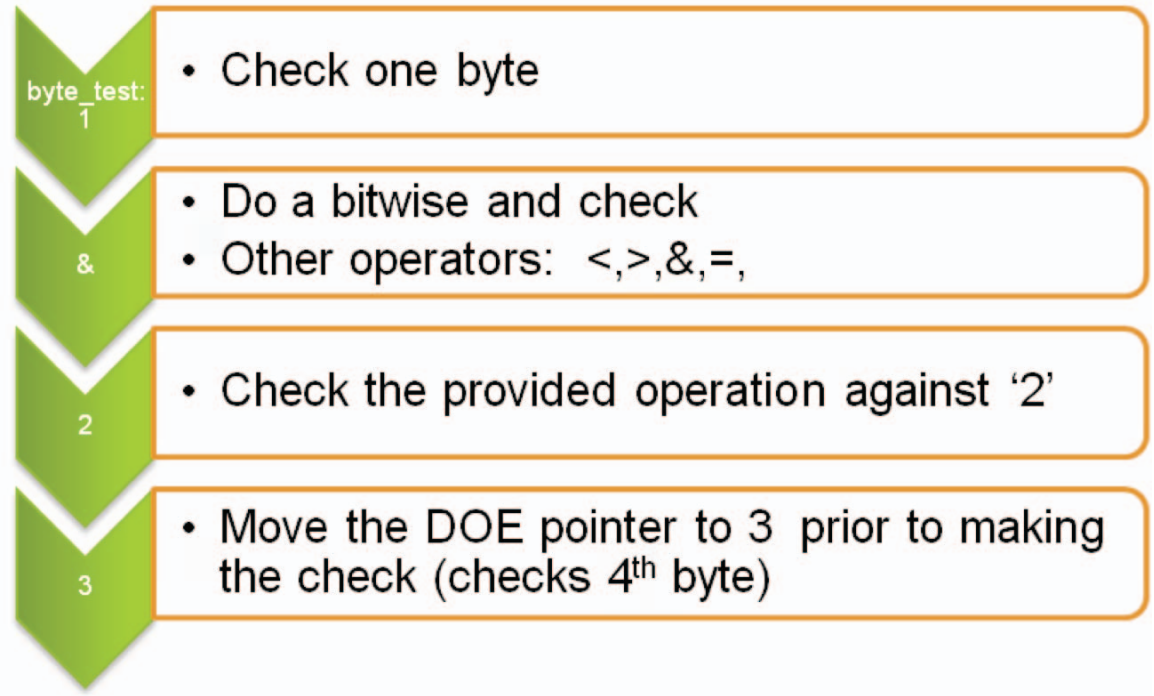
Example 4: Detection notes for the Kaminsky Bug

- A fairly in-depth technical review of both the bug itself and the detection methodology is available on the vrt white papers page:
  - http://www.snort.org/vrt/docs/white_papers/
- In short, the detection looks for ‘backscatter’
  - Backscatter is the legitimate traffic that is formed in response to attack traffic
  - The attack floods a server with random requests, and the legitimate servers will send a flood of NXDOMAIN (i.e. I have no idea what you’re talking about…) responses.

Example 4: Kaminsky Bug (sid: 13948)

byte_test:1,&,2,3; byte_test:1,&,1,3; byte_test:1,&,128,2; threshold:type threshold, track by_src, count 200, seconds 30;

- byte_test:1,&,2,3;
  - byte_test: <bytes to convert>, [!]<operator>, <value>, <offset> [,relative] [,<endian>] [,<number type>, string];



Example 4: Why 3 separate byte_tests?

- Byte_test: 1, &, 3, 3;
  - On any byte_test, a non-zero response is a success, so all three cases below will pass the byte test, even though only the case with both flags set is correct

	Both flags set	First flag set	Second flag set
Packet Value	00000011	00000001	00000010
Byte_test Value	00000011	00000011	00000011
Result	00000011	00000001	00000010

- The proper approach is to check both values:
  - byte_test: 1, &, 2, 3;
  - byte_test: 1, &, 1, 3;
- The final byte_test ensures the packet is a response:
  - byte_test:1,&,128,2;

Example 4: Thresholding

threshold:type threshold, track by_src, count 200, seconds 30;

- Thresholding is commonly used while tuning the IDS.
- Thresholding is discussed in section 3.8 of the Snort Users Manual
  - First packet starts the window
  - Each additional packet checks for expiration and increments the counter
- The thresholding is actually a critical part of the detection methodology:
  - The actual attack would generate a huge volume of legitimate NXDOMAIN responses.
- In this case, we are looking for 200 NXDOMAIN responses within a 30 second window.

Example 4: Thresholding

threshold:type threshold, track by_src, count 200, seconds 30;

- In this case, we are looking for 200 NXDOMAIN responses within a 30 second window.
- The thresholding is actually a critical part of the detection methodology:
  - The actual attack would generate a huge volume of legitimate NXDOMAIN responses.
- Thresholding is discussed in section 3.8 of the Snort Users Manual
  - First packet starts the window
  - Each additional packet checks for expiration and increments the counter

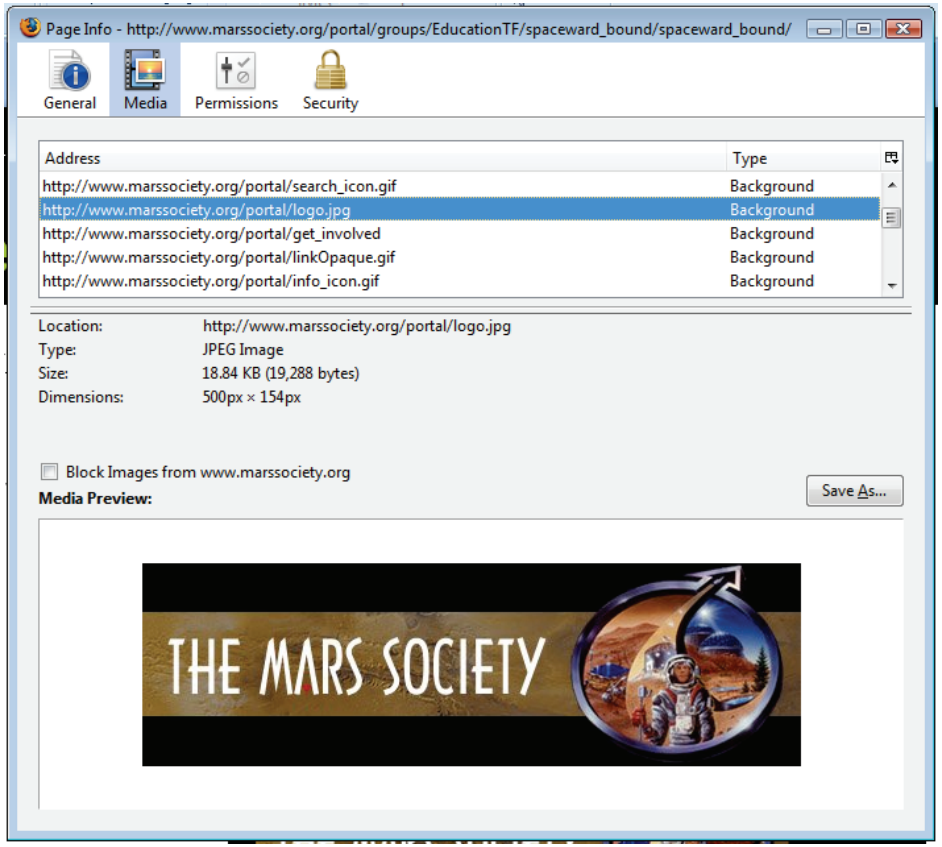
Example 5: Let's mess with Alex

This is Alex!

- Alex is one of the rule writers here in the VRT
- Alex spends some of his free time as a webmaster for the Mars Society
- We love Alex
- We love to mess with people.

Example 5: The goal…

Targeting only Alex's computer, let's replace the banner on the Mars Society website with one of our choosing.



Example 5: In a javascript file...

- First we need to figure out how the image is being delivered.
- So by looking through some packet captures, we came across this gem...
- We need to replace:
  - http://www.marssociety.org/portal/logo.jpg
  - With a file of our choosing...

```
#portal-logo {
background:
    url(http://www.marssociety.org/
    portal/logo.jpg) no-repeat;
border: 0;
margin: 0.75em 0em 0.75em
    1.5em;
padding: 0;
}
```

Example 5: Content/Replace

alert tcp $EXTERNAL_NET $HTTP_PORTS -> $ALEXPC any (msg:"WE THINK ITS FUNNY"; flow: established, to _client; content:"http://www.marssociety.org/portal/logo.jpg"; replace:"http://vrt-app-01/imagemakeevenlongerr.jpg"; classtype: successful-dos; sid: 100001;)

- This rule replaces any instance of the logo.jpg string inbound to Alex's box with the link to our file (hosted on a local box).
  - Filename was modified to satisfy the equality of length requirement.
  - This replace causes Alex's browser to request the modified banner and then insert it into the presented webpage.
- Notes on the replace keyword:
  - Detailed in section 1.5.3 of the Snort Users Manual
  - The only thing to remember is the replace content must match the length of the content it is replacing.

Example 5: Well, we thought it was funny…

Before

After

Example 5: Disclaimer

- All modifications were done to traffic just before it reached Alex's computer
- No exploits or attacks were used, only the functionality of the Snort engine
- We had advanced approval

Questions?

- If you have questions in general:
  - snort-sigs mailing list
  - snort-users mailing list
  - #snort on freenode irc
  - research@sourcefire.com
- If you have questions or comments on this presentation:
  - molney@sourcefire.com

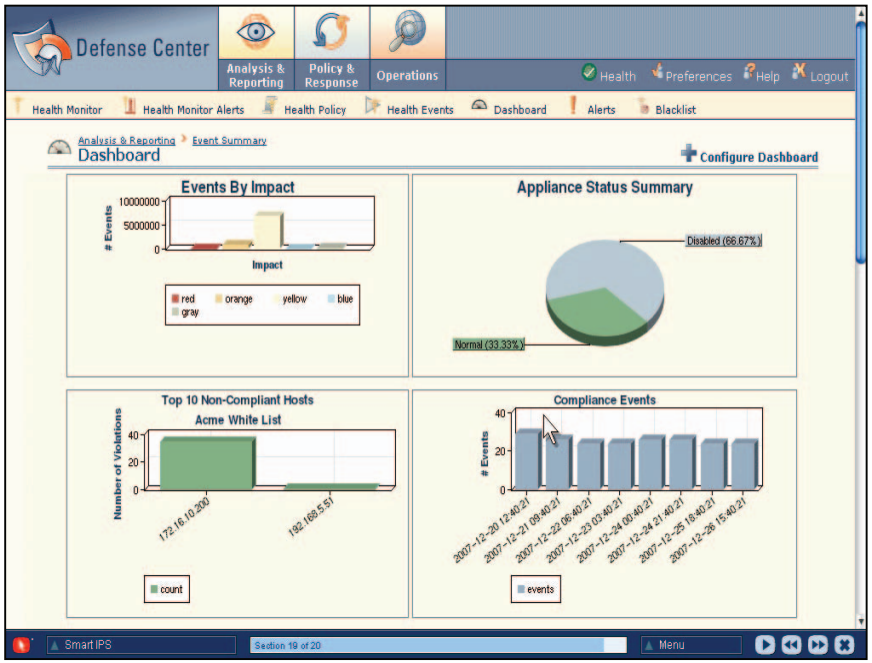
Sourcefire Commercial Products

Sourcefire 3D™ System

- Sourcefire 3D Sensors
  - Sourcefire IPS™
  - Sourcefire RNA™
  - Sourcefire RUA™
  - Sourcefire NetFlow Analysis
- Sourcefire Defense Center™
- Sourcefire Intrusion Agent for Snort

For More Information…

Sourcefire 3D System Flash Demo



"Extending Your Investment in Snort" Technology Brief

Available Now on Sourcefire.com

Questions?

Please submit questions via the Q&A interface in the lower-right corner of your screen.