Application Level Content Scrubbers

Securing an organization's content servers (be it web, file or mail servers) was at one time the primary domain of packet filtering routers. As the Internet became the mainstream medium it is today, the attackers and their attacks became more sophisticated; packet filters were no longer suitable and the perimeter defences evolved towards session awareness – the current benchmark technology being stateful inspection which not only understands sessions but also the basics of an application protocol (e.g. Firewall-1 understands how an FTP session should be setup).

Firewalls clearly excel at keeping clearly undesirable traffic from getting in and allowing acceptable traffic out. Unfortunately, firewalls do not excel in the e-business and content delivery environments that most organizations are interested in protecting. This is because firewalls were originally created for express purpose of blocking external access while still allowing internal users out.

In the e-commerce or content delivery environments, the firewalls can only ensure that a certain variety of traffic will reach the online assets e.g. TCP traffic over port 80 to server xyz; however, a firewall has no control over the actual content the traffic carries. A Cisco PIX can block Java or ActiveX and Firewall-1 has content vectoring, but these measures are only designed to prevent internal users from accidentally bringing malicious content back inside the perimeter; they are not designed or implemented to prevent attacks against servers using malicious content that is syntactically correct from a protocol perspective yet from the perspective of the content providing applications (e.g. a generic web server, Outlook Web Access or a customized online shopping application) semantically dangerous.

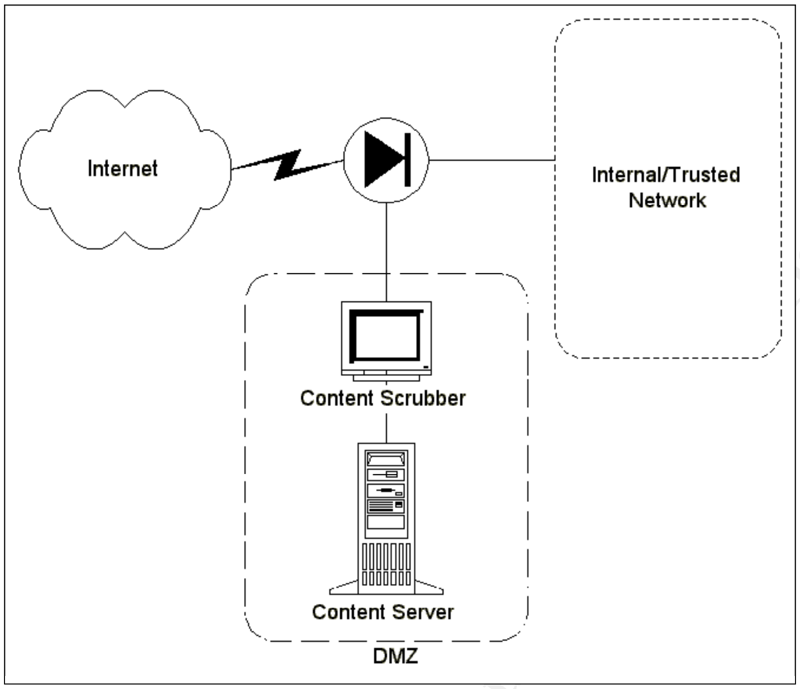
Over the past year, products and solutions have arrived that allow for application level control of inbound content. For the purposes of this paper, I have classified these products and solutions as "content scrubbers". They are not meant as firewall replacements, but rather to augment existing network security architecture by providing a capability previously lacking[1].

This paper presents an overview of some of the available content scrubbers (this is not meant to be a comprehensive product comparison).

General overview

Content scrubbers sit between the content providing servers and the firewall; they are designed to block (or respond to) malicious inbound content requests.

[1] Hurley, Michael. "Network Air Gaps – Drawbridge to the Backend Office" 4 April 2001 http://www.sans.org/infosecFAQ/firewall/gaps.htm (August 22, 2001)



Malicious inbound content requests for the purpose of this paper can be described as followed:

- Originates externally
  The inbound content originated from outside the security perimeter and the session itself was initiated externally.
- Passes firewall rules
  The packet(s) containing the request have a valid source and destination and comply with network and transport layer specifications.
- Syntactically correct
  The request is compliant with the published RFC's for the application (it would therefore pass through an application proxy) and the sessions are established correctly.
- Semantically Dangerous
  The content of the packet(s), the request itself, will cause the responding application to behave in a manner that is not as the developers/vendors intended and impacts application integrity or availability. This may be a buffer overflow that causes the application fail or execute arbitrary code.

Content Scrubbers

e-Gap

Description

A hardware based solution sold by Whale Communications (www.whale-com.com). The e-Gap is effectively a reverse proxy (although it has much higher throughput)[2] that is aimed at securing e-business transactions. It provides content control capabilities, at an infrastructure level, that allow for verification of all inbound content requests based on business transaction criteria (e.g. the e-Gap can be configured to know that the phone number field on a form should only contain nine numbers when it is submitted; anything else would cause the form to be rejected).

For a more in depth review of Air Gap technology and Whale's e-Gap, please see Michael Hurley's article "Network Air Gaps – Drawbridge to the Backend Office" (http://www.sans.org/infosecFAQ/firewall/gaps.htm).

Content Scrubbing Capabilities

Strengths

- Solution centric[3]
  The e-Gap is focused on delivering content security for specific products e.g. Outlook Web Access[4] and its rulesets are therefore tailored to deliver optimal security.
- E-business integrity
  The content inspection mechanism not only ensures that malicious requests cannot be made, it also verifies content requests based on business transaction criteria.
- Network separation
  The e-Gap is a three component solution (an external server, the real-time switch and the internal server). Since there is no network connection between the external and internal server, a compromise of the external server means that internal network remains secure.
- SSL capable
  The e-Gap can terminate SSL sessions and inspect contents in the clear for compliance with its rules.
- Scalable enterprise class solution
  The e-Gap is designed for high traffic environments and therefore

Disadvantages

- Pricing
  While the price tag is by no means exorbitant for an enterprise class security product, it may put the product out of the reach of smaller organizations

Hogwash

Description

[2] Braunhaut, Jonathan – Whale Communications; conversation – December 12, 2000
[3] Kuo, Vicki – Whale Communications; conversation – July 10, 2001
[4] "e-Gap Webmail System" http://www.whalecommunications.com/fr_0200.htm (August 22, 2001)

Hogwash (http://hogwash.sourceforge.net) is a derivative of Snort (the Open Source IDS) and bills itself as a packet scrubber or a signature based firewall. It is designed to live inline with the network feed and drop malicious packets[5] based on content inspection rules.

Strengths

- *Open Source*
  Open Source software (depending on the licensing variant) makes the code and binaries freely available to all parties. Not only does this mean that licensing costs are eliminated but that the organization has direct access to the source code and can modify it in any manner they see fit. This potentially makes for a security product that can be tailored to specific requirements.
- *Established and Reviewed Code base*
  Since Hogwash is based on Snort, it benefits from its proven/mature code base that is reviewed by security professionals and programmers on an ongoing basis for any deficiencies. However, Hogwash is still new code and will have to prove itself on its own merit. Unfortunately, Open Source allows code review not only by White Hats but also by Black Hats, so that means your attackers may know about in Hogwash that can be exploited.
- Rulesets
  Again, Hogwash benefits from its relation to Snort; Snort rulesets (the signatures used to detect intrusions) are updated on a regular basis (a new file is available every 30 minutes[6]). Since hogwash uses a similar ruleset structure[7] any new rule/attack signature for Snort is useable under Hogwash.
- Lightweight
  Hogwash is lightweight and designed to run on basic hardware[8]. This helps organizations with limited security budgets still implement the defences needed to block intruders.
- Invisible
  Hogwash operates at Layer 2 and has no need for a TCP/IP stack and therefore, should not be detectable or accessible; this enhances overall security since attackers would probably not be aware of the systems existence nor would they be able to access it and compromise it directly.
- Extendable
  Hogwash shares Snort ability to use pre-processor and output plugins (e.g. OPSEC integration, reporting modules, stream reassemblers). This allows an organization to add functionality to Hogwash on the fly.

Disadvantages

[5] Larsen, Jason 'The "I wrote it in 10 minutes" FAQ' http://hogwash.sourceforge.net/HogWash_files/faq.html (August 22, 2001)
[6] http://snort.sourcefire.com/downloads.html (August 22, 2001)
[7] Larsen, Jason "Hogwash Documentation" http://hogwash.sourceforge.net/HogWash_files/using.html (August 22, 2001)
[8] Larsen, Jason 'The "I wrote it in 10 minutes" FAQ' http://hogwash.sourceforge.net/HogWash_files/faq.html (August 22, 2001)

- Cannot decode SSL[9]
  Snort, and therefore Hogwash, was designed to be lightweight. SSL decoding is processor intensive and is not implemented. Therefore, any content based attacks in an SSL sessions will not be stopped. However, this can easily be addressed by terminating the SSL connection at the firewall and forwarding it to the web server in the clear.
- No vendor support
  Obviously, Open Source software does not come with vendor support. This may present a challenge to organizations without the requisite in house expertise and this could result in Hogwash being incorrectly deployed and would therefore not be as effective in enhancing network security.

http_filter

Description

Originally developed by Byron Jones to augment security for Outlook Web Access[10], the initial script was improved upon and released as freeware (http://glob.com.au/http_filter/). Http_filter is an HTTP tunnel with filtering and multiplexing capabilities that resides on the firewall, sitting in front of web servers. Http_filter accepts requests at the firewall, applies a set of rules to them, and allows the requests to be passed through to the back-end Web server only if they pass all filters[11].

Strengths

- Perl
  http_filter is written in Perl and this not only makes it far more portable to multiple O/S'es but also far easier to modify given that the language is far more accessible.
- Freeware
  Licensing costs are eliminated which is of benefit to cost conscious security administrators.
- Multiplexing
  Allows requests to port 80 to be directed to different web servers depending on the content requested. This may reduce the complexity of a firewall administrators job as she would only need to write a single rule for inbound web requests and then let http_filter handle redirection via separate tunnels.

Disadvantages

- Perl
  Perl is not a compiled language and therefore lacks the speed of Hogwash; this

[9]Roesch, Martin "Re: Snort New Feature Request" August 17, 2001 http://groups.google.com/groups?hl=en&safe=off&th=bcccc1271668b325,2&seekm=9llff0%2419lt%241%40FreeBSD.csie.NCTU.edu.tw#p (August 22, 2001)
[10] Jones, Byron "http_filter" http://glob.com.au/http_filter/ (August 22, 2001)
[11] Security Focus "http_filter v1.2" http://www.securityfocus.com/tools/2123 (August 22, 2001)

makes it unsuitable for high traffic environments.

- Web traffic only
  http_filter is, as its name implies, only designed to work with web traffic. This means that a security administrator that wishes to secure any other service will need to use additional software (unlike Hogwash which supports all traffic types and e-Gap that has a number of application specific shuttles as well as an available software development kit).
- Cannot Decode SSL
- Resides on the firewall
  Since the http_filter application resides on the firewall, if an attacker manages to compromise it, they may also be able to compromise the firewall itself and gain access to protected networks.
- Client IP's altered
  http_filter is not transparent and will modify the originating IP addresses of a request. This can impact IP address restrictions/authentication and accurate logging on the web server. In order to address this, the firewall that http_filter resides on must be reconfigured to handle IP address restrictions/authentication and logging may need to improved/increased.
- No vendor support

Conclusion

Given that attacks have moved up the OSI model to the application level, it is clear that stateful inspection firewalls alone will not suffice. As content scrubbers mature, hopefully we will see the development of application specific content and protocol dictionaries that can be integrated into the next generation of content scrubbers. With these content and protocol dictionaries, it will be far easier to develop comprehensive solutions that assess inbound content requests for dangerous semantic content. Imagine a firewall-like application that would understand that a web server receiving a form with the SQL statement XP_cmdshell in one of the returned fields was likely under attack and the request should be blocked.

References

Hurley, Michael. "Network Air Gaps - Drawbridge to the Backend Office" 4 April 2001 http://www.sans.org/infosecFAQ/firewall/gaps.htm (August 22, 2001)

Braunhaut, Jonathan - Whale Communications; conversation - December 12, 2000

Kuo, Vicki - Whale Communications; conversation - July 10, 2001

"e-Gap Webmail System" http://www.whalecommunications.com/fr_0200.htm (August 22, 2001)

Larsen, Jason 'The "I wrote it in 10 minutes" FAQ' http://hogwash.sourceforge.net/HogWash_files/faq.html (August 22, 2001)

http://snort.sourcefire.com/downloads.html (August 22, 2001)

Larsen, Jason "Hogwash Documentation" http://hogwash.sourceforge.net/HogWash_files/using.html (August 22, 2001)

Larsen, Jason 'The "I wrote it in 10 minutes" FAQ' http://hogwash.sourceforge.net/HogWash_files/faq.html (August 22, 2001)

Roesch, Martin "Re: Snort New Feature Request" August 17, 2001 http://groups.google.com/groups?hl=en&safe=off&th=bcccc1271668b325,2&seekm=9llff0%2419lt%241%40FreeBSD.csie.NCTU.edu.tw#p (August 22, 2001)

Jones, Byron "http_filter" http://glob.com.au/http_filter/ (August 22, 2001)

Security Focus "http_filter v1.2" http://www.securityfocus.com/tools/2123 (August 22, 2001)