SecureWorks

Multiple Vulnerabilities in Cisco ASA

Jeff Jarmoc, SecureWorks Inc.

Black Hat USA 2010

Abstract

Firewalls, being among the oldest of security devices, have become somewhat less than glamorous. They are generally accepted as the bare minimum in network security and the silent sentinels guarding networks around the world, largely ignored and often passively monitored and managed. This lack of attention, however, may lead to problems both through misconfigurations and buggy behavior going unnoticed, and through direct compromise. It's unfortunately all too common for firewalls to be considered first as network devices responsible for passing traffic, and only secondly as enablers of enterprise security. This lack of attention to these oft-forgotten devices can lead to increased risk — risk that is easily avoidable through proper monitoring, attention to detail, strong administrative practices, and proper focus on risk analysis and patching.

Cisco Adaptive Security Appliance (ASA) is Cisco's flagship firewall product. ASA replaced the older PIX (Packet Internet eXchange) firewall and has become one of the most widely deployed perimeter filtering devices. ASA often serves as a front line of defense to corporate and enterprise networks. Through features such as its user-friendly web-based Adaptive Security Device Manager GUI, protocol inspection and analysis, VPN and SSL VPN termination, as well as less traditional firewall features like active routing participation, voice and video support, QOS, and content inspection, ASA has grown to provide a broad range of functionality. This expanding feature set also presents a larger attack surface, and presents an increasingly enticing target to attackers. Despite being a security appliance, ASA is unfortunately not immune to programming and design errors that present themselves as security impacting bugs and vulnerabilities.

ACL Bypass Vulnerability

(CVE-2009-1160, Cisco bug ID CSCsq91277)

At the very heart of the Cisco Adaptive Security Appliance (ASA) functionality is traditional stateful firewalling. To be called a stateful inspection firewall, a device must be capable of filtering traffic based on layer three and four information, such as source and destination IP addresses and ports or services. Additionally, session state is taken into account so that, for example, replies to outbound requests can be permitted while newly initiated inbound requests are denied. However, there exists at least one rare set of circumstances that causes even this basic feature to fail catastrophically.

Background & Configuration Summary

In the initial configuration of an ASA, an engineer performs several tasks to configure its network interfaces, provide basic information about the network(s) it protects and define the policies it should enforce. The relevant configuration is listed in Figure 1. Full documentation of these commands is available via Cisco's web site and product documentation.

The general workflow is summarized below, and a small configuration snippet is shown in Figure 1.

- Name each interface (nameif command)
- Configure a Security level for each interface — a numeric weight of each interface's relative level of trust. (security-level command)
- Assign an IP address to each interface (IP address command)
- Optionally, create an Access Control List (ACL) for each interface and direction (in/out) in which traffic is to be inspected (access-list command)
- Optionally, apply the ACLs to their respective interfaces and choose the directionality of traffic to be compared against it (access-group command)

```
interface Ethernet0/0
 nameif outside
 security-level 0
 ip address 192.168.1.222 255.255.255.0
!
interface Ethernet0/1
 nameif inside
 security-level 100
 ip address 10.10.10.1 255.255.255.0
!
interface Ethernet0/2
 nameif dmz
 security-level 50
 ip address 10.10.20.1 255.255.255.0
!
<output truncated>
access-list outside remark ### Obviously, the below is for demonstration
purposes only and is extremely permissive.  ###
access-list outside extended deny tcp any any eq ssh
access-list outside extended permit ip any any
access-list inside extended permit tcp host 10.10.10.0 any eq www
access-list inside extended permit tcp host 10.10.10.0 any eq https
access-list inside extended permit udp any host 10.10.20.53 eq domain
access-list dmz extended permit tcp host 10.10.20.25 any eq smtp
access-list dmz extended permit udp host 10.10.20.53 any eq domain
<output truncated>
access-group outside in interface outside
access-group inside in interface inside
access-group dmz in interface dmz
<output truncated>
```

Figure 1. Sample ASA configuration snippet.

If the access-list and access-group commands are not applied to an interface, it will default to allowing traffic based on security levels. In this configuration, traffic is allowed from an interface to any other interfaces with a lower security level. In our example configuration, traffic arriving at the ASA's inside interface would be allowed to destinations on either the outside or DMZ interfaces. Traffic arriving on the DMZ interface is allowed to destinations reachable via the outside interface, and traffic arriving to the outside interface is not allowed to any inside or DMZ destinations.

However, this behavior changes once an access-list is configured and bound to an interface with the access-group command. With an access-group configured, traffic matching its interface/directionality pair will be inspected by the ACL and processed according to the first matching rule. If no rule matches, then the traffic is denied. This is known as an 'implicit deny' behavior. That is, if no ACL entry explicitly allows traffic, it will be discarded. Note that ACLs can

still take a 'deny' or 'reject' action, which will take action on matching traffic accordingly. Reply traffic is implicitly allowed by the firewalls state table prior to ACL inspection, and thus no ACL entry is needed.

Description of the Vulnerability

Under some circumstances, the behavior described in the previous section can change. Certain versions and configurations of the ASA will continue to process packets against configured access-groups, but its default action will change. When the problem is present, packets not matching any access-control entries in the access-list bound via the access-group command will be processed according to security-interface behavior, rather than matching a default deny rule. This outcome effectively bypasses the implicit deny behavior, rendering ACLs relying on it moot.

The behavior appears to be caused by the initial configuration of the device. When access-list and access-group commands are entered in the wrong order, the ASA typically generates an error and refuses to accept the configuration. However, it seems this was not the case with older versions of ASA. The commands would be accepted and function as intended until the device is upgraded to a version vulnerable to this bypass issue. After the ASA is upgraded to a vulnerable version, the implicit deny is replaced by the security-level behavior.

There are a few ways to detect if a given device is impacted. Viewing the configuration shows everything as normally configured, and the commands are displayed in the proper order with no errors. Only by analyzing syslog messages or tracing traffic using ASA's packet tracer is the abnormal behavior apparent. Note that the ASA must be configured at level 6 (informational) or level 7 (debug) for these logs to be generated. Oftentimes, logs are collected at higher severity levels, if at all.

```
Feb 13 2009 14:50:21 demoasa : %ASA-6-302013: Built outbound TCP connection
451649364 for outside:a.b.c.d/80 (a.b.c.d/80) to inside:10.1.1.100/1469
(192.168.1.222/24278)
Feb 13 2009 14:50:21 demoasa : %ASA-6-305011: Built dynamic TCP translation
from inside:10.1.1.100/1470 to outside:192.168.1.222/7792
Feb 13 2009 14:50:21 demoasa : %ASA-6-302013: Built outbound TCP connection
451649365 for outside:a.b.c.d/80 (a.b.c.d/80) to inside:10.1.1.100/1470
(192.168.1.222/7792)
Feb 13 2009 14:50:21 demoasa : %ASA-6-305011: Built dynamic TCP translation
from inside:10.1.1.100/1471 to outside:192.168.1.222/52312
Feb 13 2009 14:50:21 demoasa : %ASA-6-302013: Built outbound TCP connection
451649401 for outside:a.b.c.d/80 (a.b.c.d/80) to inside:10.1.1.100/1471
(192.168.1.222/52312)
Feb 13 2009 14:50:22 demoasa : %ASA-6-305011: Built dynamic TCP translation
from inside:10.1.1.100/1472 to outside:192.168.1.222/37014
Feb 13 2009 14:50:22 demoasa : %ASA-6-302013: Built outbound TCP connection
451649519 for outside:a.b.c.d/80 (a.b.c.d/80) to inside:10.1.1.100/1472
(192.168.1.222/37014)
```

Figure 2. Syslogs showing traffic passing.

```
packet-tracer input inside tcp 10.1.1.100 1486 a.b.c.d 80

<output truncated>
...
Phase: 2
Type: ACCESS-LIST
Subtype:
Result: ALLOW
Config:
Implicit Rule
Additional Information:
 Forward Flow based lookup yields rule:
 in  id=0x1a09d350, priority=1, domain=permit, deny=false
      hits=1144595557, user_data=0x0, cs_id=0x0, l3_type=0x8
      src mac=0000.0000.0000, mask=0000.0000.0000
      dst mac=0000.0000.0000, mask=0000.0000.0000

<output truncated>
```

Figure 3. Packet-tracer showing traffic passing

ACL Bypass advisory:
http://www.cisco.com/warp/public/707/cisco-sa-20090408-asa.shtml

Conclusion

This issue affects only a small number of ASAs that were configured in an unusual manner on an (unknown to the authors) older version of ASA OS. There is no known way to trigger this behavior at will or remotely. However, the impact is still significant, unexpected, and difficult to detect, so this is a good lesson in why thorough device monitoring and patching are important. The identified behavior both drastically changes the security posture of a protected environment and can easily go unnoticed. The nature of the vulnerability supports the following points:

- Firewall logs should be thoroughly monitored, including logs showing sessions that were allowed. By following trends and averages in these logs, it's apparent when unusual activity occurs, such as allowing all outbound traffic,. This monitoring may require debug or informational level logging, and almost certainly requires some form of automated correlation.
- Be prepared to rapidly patch a production firewall upon identification of a serious malfunction. Network architecture should account for this possibility, and allow for upgrading without causing downtime for the protected network(s). High availability clusters and/or load balancers may be useful tools.
- Consider explicitly dropping traffic at the end of each ACL. While this means administrators must manually position lines appropriately within the ACL when creating new ACEs, this also forces more thought and foresight to go into rulebase modifications.

ASDM Command injection

Background & Configuration Summary

Adaptive Security Device Manager (ASDM) is a Java GUI for managing ASA. It presents a more intuitive interface than the command line, and is frequently used by administrators who find the command line daunting or simply prefer a GUI to ASA's command line interface. It's a modern replacement for the legacy PIX Device Manager (PDM) tool that provided similar functionality on PIX appliances. ASDM functions by communicating with the ASA over an HTTPS channel to send commands and receive responses. Essentially, it can be thought of as a web administration application with a Java front end.

Using an SSL interception proxy such as BurpSuite, Fiddler, or WebScarab, we can understand the communications between ASDM client and ASA server.

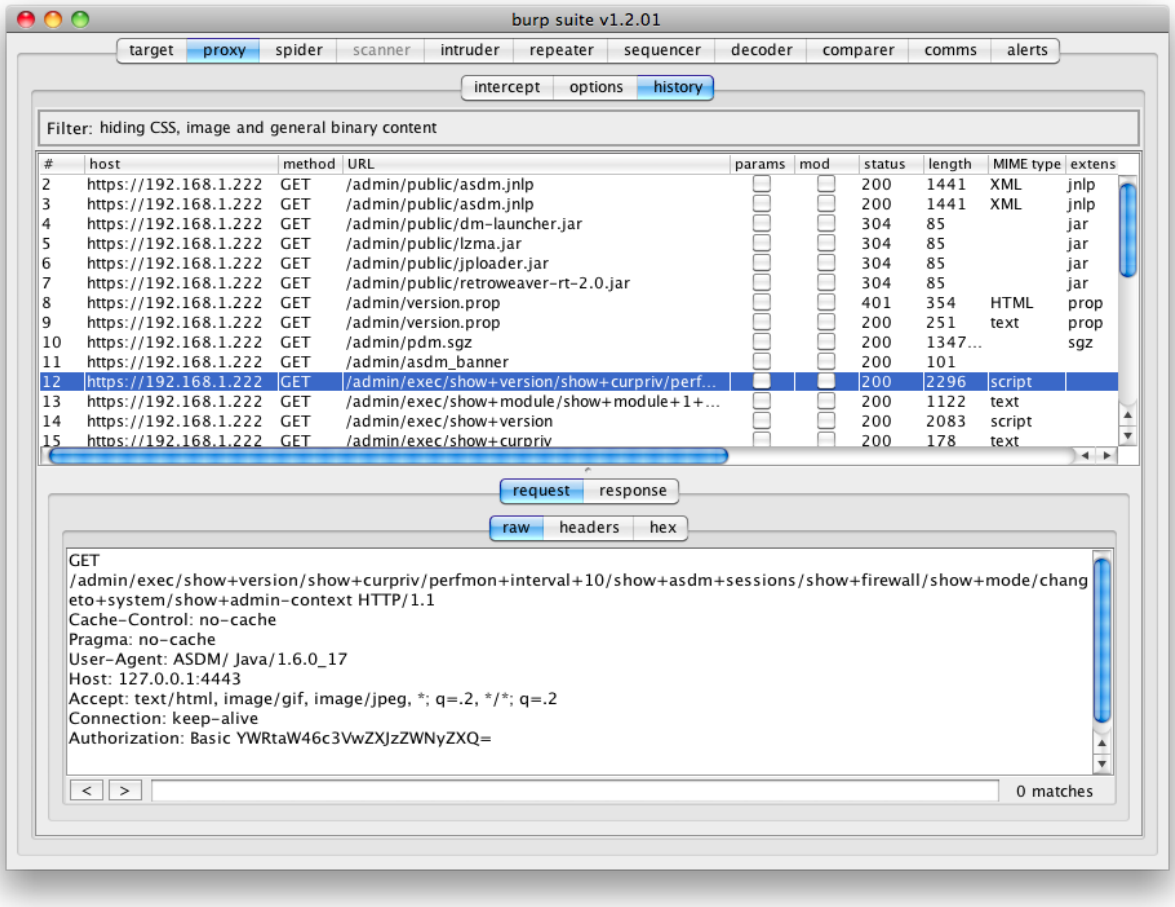

Figure 1. ASDM communications intercepted by Burp.

By analyzing this communication, we see that commands are sent over GET requests from the ASDM client. We see multiple paths being used, and can query these same paths from a web browser:

Path	Purpose	Security
/admin/	Root of ASA management interface	Anonymous
/admin/public/	Stores .jar, .jnlp, and other supporting files	Anonymous
/admin/exec/	Root of commands to be executed. Commands are passed as HTTP encoded paths	Auth required
/admin/config/	Returns the current running-config	Auth required
/admin/capture/	Stores any captures configured, appending /pcap/ returns them in .pcap format.	Auth required

Some examples of commonly used URLs:

To get the version of a device, connect to https://a.b.c.d/admin/exec/sh+ver/

To download a pcap of a capture name 'test', connect to https://a.b.c.d/admin/capture/test/pcap/

To view the current time and an access list called 'inside' connect to https://a.b.c.d/admin/exec/sh+clock/sh+access-list+inside/

Looking at the intercepted session in Figure 1, the first several connections show the ASDM client .jnlp being downloaded (since we launched from a browser instead of through an installed copy of ASDM launcher.) Once this Java client is downloaded and launched, several requests check the device's version and download supporting information.

These requests continue without any credentials until request 12, where we see the first request including a command sent to /admin/exec/ Notice that this request includes HTTP Basic Auth credentials. These credentials are easily Base64 decoded to reveal the username and password. Our example shows "YWRtaW46c3VwZXJzZWNyZXQ=" which decodes to a concatenated username and password, separated by a colon: "admin:supersecret".

Because ASDM uses a weak authentication model and submits commands through HTTP GET, there is much reliance on SSL encryption to provide both confidentiality and integrity. If SSL can be overcome, there are several ways to take advantage of these weaknesses.

Credential interception
By intercepting and re-encrypting data, all commands and data can be read. This action requires sending a different certificate to the client. In many environments ASAs use self-signed certificates, so it's possible that the user may not be suspicious of certificate related errors.

Cross-Site Request Forgery
Because ASDM uses only a GET request for commands sent to the ASA, Cross-Site Request forgery (CSRF) can be trivially accomplished. This action requires that the victim visit a path on the ASA, which requires credentials through their browser. However, there are some cases where Cisco recommends performing this procedure, namely, to back up the firewall's configuration with IPSEC pre-shared keys visible, or to transfer pcap files from it. Attacks may take advantage of other methods, such as fetching browser history through CSS, to target their attempts.

Most notably, the article PIX/ASA 7.x: Pre-shared Key Recovery presents four ways to recover an IPSEC VPN's pre-shared key. These keys are not displayed to authorized administrators through an encrypted (SSH or HTTPS) session. However, all the 'solutions' are flawed. The first involves leveraging the 'more' command to display the config, which does not mask the pre-shared key. Cisco later determined this behavior to be a bug (CSCeh98117), which has been fixed in 8.3 despite still being a recommended 'solution' in this article. Two of the other options require transferring the configuration over cleartext TFTP or FTP. This option is inherently less secure than displaying the credentials over SSH, which is apparently deemed too risky by Cisco. The remaining solution involves accessing the configuration over HTTPS from a browser. This browser will then cache the user's credentials and leave them susceptible to CSRF exploits. All four proposed solutions sacrifice varying degrees of security in an attempt to mask configuration information from an authorized administrator attempting to view it on an encrypted session.

SSL renegotiation command injection
The SSL Renegotiation vulnerability (CVE-2009-3555) was first discovered by Marsh Ray and Steve Dispenza of Phone Factor in November, 2009. Much has been written about this vulnerability, and a detailed description could easily become a paper of its own. However, a brief description of the issue is necessary to explain its applicability to ASA and ASDM.

The vulnerability allows a man-in-the-middle to inject arbitrary plaintext into an SSL session. In a very simplified view, the attacker does this by sending plaintext that is buffered by the recipient, and then asking both sides to renegotiate their cryptographic association. When they renegotiate, the buffered plaintext is prepended to the ciphertext and injected into the session. This outcome compromises the integrity of the session but not its confidentiality: the attacker is neither able to read the request, nor view the response from the server.

This vulnerability affected every major SSL/TLS implementation, not only Cisco's. Cisco was quick to release an advisory, but was unfortunately light on the details of the impact. The advisory covers multiple platforms and so only states that '…the impact of an attack depends on the application protocol running over TLS.' It refers to separate bugs for each affected product, including ASA (CSCtd00697) and ASDM (CSCtd01491), neither of which include any statement as to impact. As we'll demonstrate below, a man in the middle can completely compromise an ASA

by injecting commands into an authorized administrator's session. This compromise fundamentally and significantly raises the impact of this vulnerability.

Cisco has patched the issue by entirely disabling support for SSL renegotiation in recent releases. 8.2(2) is the first release which is both not vulnerable and not an interim release. Detailed release information is available in the bug reports linked above.

As an example, consider the following request from an ASDM client to an ASA:

```
GET /admin/exec/show+version/show+curpriv/perfmon+interval+10/ HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
User-Agent: ASDM/ Java/1.6.0_17
Host: 127.0.0.1:4443
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Authorization: Basic YWRtaW46c3VwZXJzZWNyZXq=
```

An attacker injects the text:

```
GET /admin/exec/name+1.1.1.1+pwn3d/ HTTP/1.1
X-ignore:
```

When this text is injected into the session and prepended to the original request, the following transaction occurs, which will be received by the target ASA:

```
GET /admin/exec/name+1.1.1.1+pwn3d/ HTTP/1.1
X-ignore: GET /admin/exec/show+version/show+curpriv/perfmon+interval+10/
HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
User-Agent: ASDM/ Java/1.6.0_17
Host: 127.0.0.1:4443
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Authorization: Basic YWRtaW46c3VwZXJzZWNyZXq=
```

This is essentially what our proof of concept exploit does. While we won't be releasing our code, there is public code available as a PoC against the SSL Renegotiation issue. This is available at exploitdb.com as well as on the web site of Red Team Pentesting Gmbh. This example code requires some modifications to function with ASDM. Most notably, it must be modified to skip the first several requests, which as we've seen above are unauthenticated. Since we can't read the contents of the traffic, were merely skip the first several packets from a given source.

Conclusion

This example should teach us that administrative access is sensitive and should be authorized and used cautiously. Administrative access should be restricted to only the required sources, and if possible confined to a dedicated network segment. This practice can help minimize the risks of an attacker gaining administrative access.

Additionally, review security advisories with a cautious eye and consider impacts that may not be clearly disclosed.

Patching against serious vulnerabilities also remains good practice. Networks should be designed in such a way that allows for zero downtime upgrades, increasing the flexibility of our responsiveness.

Closing

The vulnerabilities detailed in this paper can be taken as examples of the threats facing modern firewalls. Considering these issues and the possibility of others, in firewall engineering and network design can help produce a robust, secure operating environment. Specific action items may include:

- Review vendor advisories and perform impact assessment on a regular basis. Infrastructure should be patched with at least the same vigor as endpoint.
- Review release notes for new versions, and associated bug fixes, to determine if there may be security issues that aren't presented as such.
- Consider firewalls and other critical infrastructure in scope for penetration testing and vulnerability assessment.
- Design networks in a redundant fault tolerant fashion. In addition to providing increased uptime in the case of failure, this allows for more rapid patching and upgrades as needed, without impacting the production environment.
- Thoroughly monitor device behavior to verify proper operation and policy enforcement.
- Wherever possible, segregate and restrict administrative traffic to reduce the possibility of compromise against these sensitive interfaces.
- Disable unnecessary features to reduce available attack surface.