

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES

Written By: Philip Kwan
February 2003

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Written by: Philip Kwan, January 2003

Summary

The IronShield Best Practices: Hardening Foundry Routers & Switches document is designed to help network and security administrators understand how to implement Foundry security features. The document gives the reasons why the security features are necessary and how to best implement them to compliment existing security devices, such as firewalls and IDS systems, to create a robust secure network infrastructure. The key is "Defense in Depth" and to apply security at all layers of the enterprise.

IronShield Security is not meant to replace Data Security infrastructures. With careful planning and implementation, IronShield Security features can improve Network Security and enhance Data Security where it's needed.

Contents

Introduction	5
<i>Audience</i>	<i>5</i>
<i>Nomenclature</i>	<i>6</i>
<i>Related Publications</i>	<i>6</i>
Why Harden Routers and Switches	7
<i>Which Routers And Switches To Protect</i>	<i>7</i>
<i>Applying Security Features</i>	<i>8</i>
<i>Security Components For Hardening Foundry Devices</i>	<i>9</i>
<i>Summary of Foundry Security Features</i>	<i>10</i>
Warning Banners	12
<i>Setting Up A Warning Banner</i>	<i>13</i>
<i>banner exec</i>	<i>13</i>
<i>banner incoming</i>	<i>13</i>
<i>banner motd</i>	<i>13</i>
Configuring Access	14
<i>Remote Administration Concerns</i>	<i>15</i>
<i>Determining What To Implement</i>	<i>16</i>
Selection of Strong Passwords	17
Configuration File Security	18
Authentication and Authorization Features	19
<i>Setting Up Privilege EXEC and Telnet Passwords</i>	<i>19</i>
<i>Setting Telnet and Console Timeout Values</i>	<i>19</i>
<i>Setting Up Privilege Level Passwords</i>	<i>20</i>
<i>Setting Up User Accounts and Privilege Levels</i>	<i>21</i>
Restricting Management Protocols	23
<i>Limiting Remote Access To Specific IP Addresses With ACLs</i>	<i>23</i>
<i>Limiting Remote Access To Specific IP Addresses With Built-in Commands</i>	<i>24</i>
<i>Limiting Remote Access To A Specific VLAN</i>	<i>25</i>
Configuring Advanced Access	27
<i>TACACS/TACACS+</i>	<i>27</i>
<i>TACACS/TACACS+ Requirements</i>	<i>28</i>
<i>TACACS/TACACS+ Setup Procedures</i>	<i>28</i>
<i>Configuring Accounting for Telnet/SSH Access</i>	<i>34</i>
<i>Configuring Accounting for CLI Commands</i>	<i>35</i>
<i>Configuring TACACS+ Accounting for System Events</i>	<i>35</i>

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



FOUNDRY
NETWORKS

Configuring an Interface As The Source For All TACACS/TACACS+ Packets.....	35
<i>Other Useful TACACS+ Commands</i>	<i>36</i>
RADIUS AAA Security	36
<i>RADIUS Requirements</i>	<i>37</i>
<i>RADIUS Setup Procedures.....</i>	<i>37</i>
<i>Configuring RADIUS Accounting For Telnet/SSH Access</i>	<i>42</i>
Configuring RADIUS Accounting For Privilege Level CLI Commands.....	43
Configuring RADIUS Accounting For System Events	43
Configuring An Interface As The Source For All RADIUS Packets	44
<i>Other Useful RADIUS Commands</i>	<i>44</i>
Implementing Secure Shell (SSH).....	44
<i>Configuring SSH On Foundry Devices</i>	<i>45</i>
<i>Importing Public Keys.....</i>	<i>47</i>
<i>Other Useful SSH Commands</i>	<i>48</i>
<i>Enabling RSA Challenge-Response Authentication</i>	<i>48</i>
<i>Timing Out SSH Sessions</i>	<i>49</i>
<i>Helpful SSH Commands</i>	<i>49</i>
Securing SNMP	50
<i>SNMP Versions.....</i>	<i>50</i>
<i>Configuring SNMP Version 1 and 2.....</i>	<i>51</i>
<i>Securing SNMP Access.....</i>	<i>52</i>
<i>Securing SNMP To A Physical Interface</i>	<i>52</i>
<i>Blocking SNMP Access From The Internet.....</i>	<i>53</i>
<i>Configuring SNMP Views</i>	<i>53</i>
<i>Configuring SNMP v3 On Foundry Devices</i>	<i>55</i>
<i>Other Useful SNMP v3 Commands</i>	<i>58</i>
Removing Unnecessary Components	59
<i>Disabling Telnet And Suppressing Telnet Rejection Messages.....</i>	<i>59</i>
<i>Disabling Web Management</i>	<i>60</i>
<i>Disabling SNMP.....</i>	<i>60</i>
<i>Disabling Source Routing</i>	<i>61</i>
<i>Disabling L2 Bridging.....</i>	<i>62</i>
<i>Disabling ICMP Services.....</i>	<i>62</i>
ICMP MTU Discovery.....	63
ICMP Redirects.....	64
ICMP Directed Broadcasts	65
ICMP Unreachable	65
ICMP Timestamp and Information Requests.....	66
Stopping Foundry Devices From Responding to Broadcast ICMP Requests	67
Proxy ARP	67
Securing Routing Protocols	68
<i>Static Routes</i>	<i>68</i>
<i>Secure Routing Protocols</i>	<i>69</i>
<i>Securing RIP.....</i>	<i>70</i>
Method 1: RIP Filter and Filter-Groups.....	70
Method 2: RIP Neighbor Filtering	71
<i>Securing OSPF.....</i>	<i>72</i>
<i>OSPF Passwords.....</i>	<i>72</i>
Using Text Passwords	72
Using MD5 Passwords.....	72
<i>OSPF Passive Interfaces.....</i>	<i>73</i>
<i>OSPF Route Filtering.....</i>	<i>74</i>

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



OSPF Distribution Lists	74
OSPF Interface Passive/Ignore	75
<i>Securing BGP</i>	76
Common ACLs for Anti-Spoofing	77
<i>Inbound Anti-Spoofing ACLs</i>	78
<i>Outbound Anti-Spoofing ACLs</i>	79
<i>Other Services To Filter</i>	80
Preventing DoS Attacks	82
<i>Preventing Smurf Attacks</i>	82
<i>Preventing TCP SYN Attacks</i>	83
<i>Preventing LAND Attacks</i>	84
<i>Limiting Broadcast Packets</i>	85
<i>Limiting ARP Requests</i>	85
Preventing Fragmentation Attacks	86
<i>How Fragmentation Works</i>	86
<i>How Hackers Use Fragmentation</i>	86
<i>How Foundry Treats Fragmentation</i>	87
<i>CPU Inspection of Fragmented Packets</i>	87
<i>Controlling the Fragment Rate</i>	88
To Set Fragmentation Rate On Entire Chassis	88
To Set Fragmentation Rate On Interfaces	88
Dropping All Fragments For IronCore Products	89
Synchronizing Time	90
<i>Setting SNTP on Foundry Devices</i>	91
<i>Other Important SNTP Commands</i>	92
Implementing Logging Strategies	92
<i>Setting Up Logging Features</i>	93
Enabling Syslog	94
Console Logging	94
Buffered Logging	94
Changing Local Buffer Entries	95
Setting Up External Syslog Servers	95
Disabling Logging Of A Message Level	96
Classification Using Syslog Facilities	97
<i>TACACS+ and RADIUS Logging</i>	98
<i>Other Logging Recommendations</i>	98
Locking Down Ports	100
Staying On Top of OS Versions	101
Physical Security	102
Recommended Security Links	103
<i>DDoS Information Sites</i>	103
<i>Security Information Sites</i>	103
<i>Total Cost of Ownership</i>	103

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Introduction

Foundry's IronShield Security "Best Practices" papers serves as a guide to assist network and security designers in architecting and applying Foundry security features in their networks. Modern enterprise security is applied in layers – Defense in Depth. Consideration must be given to all the various layers with a full understanding of what threats are possible at each layer before implementing a defense strategy. By applying security in multiple layers, the defense is strengthened and vulnerabilities in one layer will not likely lead to successful attacks of corporate resources. The goal is to make it harder to attack your network by layering security chokepoints.

These practices should accompany your Security and Computer Usage Policies and should not replace them. Applying the steps outlined in this "Best Practices" guide does not guarantee that attacks will not be successful against your security defenses and network resources. The best security design is dynamic. It must be coupled with a strong security policy, proactive network monitoring, diligent network and security staff that is working to stay on top of security alerts and system software upgrades and patches. Enterprise data security is always changing and growing with the advent of new security threats. Thorough, rigorous, and continuous inspection of all security components and processes will help you keep on top of the enterprise network.

IronShield Security documents are specifically written to work with Foundry products and work in conjunction with related Foundry documentation. Reference to other Foundry documentation is made with regards to command syntax and general feature information.

Audience

IronShield Security "Best Practices" papers are designed to help the personnel responsible for designing and configuring the network and security components of an enterprise network. The topics discussed are at an intermediate to advanced level and assumes a good understanding of TCP/IP and related technologies.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Nomenclature

This guide uses the following typographical conventions to show information:

Italic highlights the title of another publication and occasionally emphasizes a word or phrase.

Bold highlights a CLI command.

Bold Italic highlights a term that is being defined.

Underline highlights a link on the Web management interface.

Capitals highlights field names and buttons that appear in the Web management interface.

NOTE: A note emphasizes an important fact or calls your attention to a dependency.

Related Publications

The following Foundry Networks documents supplement the information in this guide.

Foundry Security Guide - provides procedures for securing management access to Foundry devices and for protecting against Denial of Service (DoS) attacks.

Foundry Enterprise Configuration and Management Guide - provides configuration information for enterprise routing protocols including IP, RIP, IP multicast, OSPF, BGP4, VRRP and VRRPE.

Foundry NetIron Service Provider Configuration and Management Guide - provides configuration information for IS-IS and MPLS.

Foundry Switch and Router Command Line Interface Reference - provides a list and syntax information for all the Layer 2 Switch and Layer 3 Switch CLI commands.

Foundry Diagnostic Guide - provides descriptions of diagnostic commands that can help you diagnose and solve issues on Layer 2 Switches and Layer 3 Switches.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Why Harden Routers and Switches

Foundry routers and switches are used in all aspects of enterprise network infrastructure. By hardening Foundry routers and switches, you make it much more difficult for attackers to penetrate the infrastructure components of your company. Routers and switches are often the most overlooked network components with respect to Data Security. Most people think data security is only related to firewalls, intrusion detection systems, VPN's, monitoring systems, and security policies. By hardening your routers and switches you can prevent the following:

- Giving attackers information about your network so they can design a successful attack.
- Accidental or intentional reconfiguration.
- Disabling of networking infrastructure.
- Using networking components to launch further attacks.

Before any successful attack or intrusion can take place, the attacker must gather information about your company and its infrastructure. This is the surveillance or fingerprinting phase of the attack. Routers and DNS servers are primary targets because they can be used to supply valuable information about your network and hosts. Without adequate defenses, monitoring, and auditing, router and switch compromises will go undetected.

Which Routers And Switches To Protect

Every corporate router and switch is at risk. The level of risk will depend on its function and its location and other factors such as physical security and accessibility. Perimeter components are more exposed to the public Internet and often incur a higher risk. Internal routers and switches are usually protected by a firewall making them less susceptible to an external attack – or so we would think. Networks are created through a mesh of routers and switches that trust each other and communicate freely by routing protocols. They create an infrastructure to pass packets quickly between source and destination hosts.

For this reason, we must treat all routers and switches on the same risk level, as one successful attack can expose more parts of the enterprise network and its resources. Routers and switches to secure and monitor include:

- Border routers that connect your company to the Internet
- Switches that are used in the DMZ and screened subnets outside the firewall
- Routers and switches that are connected to internal trusted or secure networks
- Routers and switches that perform packet filtering

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Applying Security Features

Whenever you apply new changes to your production environment, you should take some precautionary measures. before and after making the change, to ensure a successful implementation.

- Create a backup of your device's configuration first.
- If possible, test the changes on a spare device in a lab environment beforehand.
- Test your changes by thoroughly running controlled tests against the changes made.
- Test existing features to make sure they were not affected by the changes.
- Communicate and get authorization to make the changes beforehand.
- Coordinate the necessary support staff to have them ready to test and verify key systems after the change to ensure functionality availability of service after the changes are made.
- Make the changes off peak business hours when the least amount of people will be affected.
- Communicate the status of the changes to the appropriate people and departments to close the loop.
- Document your changes and the status.

Securing each router and switch will vary depending on the hardware and operating system code of the device. As we move forward through each section, the following Foundry Networks documents supplement the information in this guide.

- *Foundry Security Guide* - provides procedures for securing management access to Foundry devices and for protecting against Denial of Service (DoS) attacks.
- *Foundry Enterprise Configuration and Management Guide* - provides configuration information for enterprise routing protocols including IP, RIP, IP multicast, OSPF, BGP4, VRRP and VRRPE.
- *Foundry NetIron Service Provider Configuration and Management Guide* - provides configuration information for IS-IS and MPLS.
- *Foundry Switch and Router Command Line Interface Reference* - provides a list and syntax information for all the Layer 2 Switch and Layer 3 Switch CLI commands.
- *Foundry Diagnostic Guide* - provides descriptions of diagnostic commands that can help you diagnose and solve issues on Layer 2 Switches and Layer 3 Switches.

Depending on your version of operating system code and hardware, not all commands may be available or the syntax may vary from what is used in this paper. Also note that some security features require CPU intervention while others are done in hardware. For these features, care must be taken when initiating the security feature to make sure they do not impede your normal network bandwidth. Testing in a controlled environment should be done before implementing the feature in a production environment.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Security Components For Hardening Foundry Devices

In order to successfully setup a hardened network infrastructure, many different aspects of accessibility and authorization will be discussed along with other best practices. We will go through each of the following security features to demonstrate common hardening practices and illustrate when and where it is best to apply them.

- Warning Banners
- Configuring Access
 - Selecting Strong Passwords
 - Authentication and Authorization Features
 - Management Protocol Restriction
 - Advanced Access
 - TACACS, TACACS+, RADIUS
- Implementing Secure Shell (SSH)
- Configuring SNMP
- Removing Unnecessary Components
 - Removing Telnet
 - Removing SNMP
 - Removing Web management
 - Enabling Route Only
 - Removing Source Routing
 - Removing ICMP Issues
 - Removing Proxy ARP
- Securing Routing Protocols
 - RipV2, OSPF, BGP4
- Configuring Anti-Spoofing ACLs
- Configuring DoS Prevention
- Guarding Against Fragmentation
- Configuring Time Synchronization
- Implementing Logging Strategies
- Locking Down Ports
- Staying On Top of OS Versions
- Reviewing Physical Security

For each feature, this paper will give examples of how to configure the feature and attempt to answer the questions of when and where to implement them. As each corporation is different and varies in corporate culture and security requirements, you will be the best judge of when and where to apply these security features.

This paper will attempt to rate the threat level for each security feature and use the following terms:

High Security Model – Installations that require highly fortified defenses for the networked devices. These devices are exposed to the public Internet and have a high probability of being scanned or attacked. These devices are used to connect valuable corporate resources that must be protected at all times - even if it means losing some accessibility and user friendliness. Physical access to these networked devices may or may not be secure. There are many support staff requiring different types of access to the network devices. Examples include network devices that are border routers, switches used in the DMZ or screened subnets outside the corporate firewall, and routers and switches that are left in commonly accessible areas and not protected by any physical security measures.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Medium Security Model - Installations that are not as highly affected by external threats as components listed in the High Security category. The installation is less likely to be probed or attacked from the public Internet. These network devices are typically behind a corporate firewall and/or are in environments that are deemed more "trusted" by the company. Not all networked devices are installed in ideal locations such as locked networking closets limiting physical access. There are many support staff requiring access to the network devices with different access levels. Examples include routers and switches that are located on the internal LAN, devices that are used to hook your company's LAN to a metro carrier, and network devices that are in semi-accessible areas.

Low Security Model – Installations that are comfortable with their Security Policy and realize that implementation of a strong security posture is not required. The chances of being attacked or probed from either the public Internet or corporate network are extremely small and trust level of employees and IT staff is very high. All or most network devices are physically secure from general access or the need for physical security is not warranted. There are only a few people who need access to the network devices to support them. Examples include small organizations that do not have a big presence on the Internet – no Ecommerce requirements located at the enterprise or they are hosted off site. Entities that foster open network architectures and do not require higher levels of security, such as educational institutions.

Applying the security features discussed in this document does not guarantee protection against security breaches, but it does lessen the chances of a successful break-in or Denial of Service attack, and make it much more difficult for the hacker(s) to use your network routers and switches for an attack.

Summary of Foundry Security Features

(Code Version 07.6.01)

This chart shows the IronShield Security features that come standard on many of Foundry's products and the security models where they can be used.

Security Feature	Low Security	Medium Security	High Security
Warning banners	✓	✓	✓
Selection of strong passwords	✓	✓	✓
Protecting backup configuration files	✓	✓	✓
Privilege EXEC and CONFIG password	✓	✓	✓
Console timeout	✓	✓	✓
Telnet access and password, Telnet timeout	✓		
Privilege Levels Passwords	✓		
User Accounts and User Privilege Level	✓	✓	

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Setting AAA Authentication Method Order	✓	✓	✓
Restricting Remote Access (Telnet, SSH, Web, SNMP) to IP Address, VLAN, ACL	✓	✓	✓
TACACS Authentication	✓		
TACACS+ Authentication, Authorization, Accounting		✓	✓
RADIUS Authentication, Authorization, Accounting		✓	✓
Secure Shell (SSH) Secure Copy (SCP)	✓	✓	✓
SNMP Version 3		✓	✓
SNMP Views		✓	✓
Blocking SNMP Using ACLs	✓	✓	✓
Removing Unnecessary Services (Telnet, Web Management, SNMP, Source Routing, L2 Bridging, ICMP Services, ICMP IP-Directed Broadcasts)	✓	✓	✓
ICMP Broadcast Response	✓	✓	✓
Proxy ARP	✓	✓	✓
Secure Routing Protocols (RIP V2, OSPF, BGP)		✓	✓
Anti-Spoofing ACLs	✓	✓	✓
DoS Attack Prevention (Smurf, SYN Flood, Broadcast Limiting, ARP Request Limiting)	✓	✓	✓
Fragmentation Attack Prevention	✓	✓	✓
Synchronizing System Date and Time	✓	✓	✓
System Logging	✓	✓	✓
Console Logging	✓	✓	✓
Disabling Unused Ports	✓	✓	✓
Port Security With MAC Address Locking		✓	✓
802.1x Port Locking		✓	✓
OS And Patch Versions	✓	✓	✓
Physical Security	✓	✓	✓

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Warning Banners

Warning banners provide no actual physical protection to block or protect the network against an attack. However, they are very necessary for complying with Corporate Security Policies and are imperative for giving your legal department and law enforcement personnel the means to prosecute the hacker(s). Without properly designed banners your case against the attacker(s) may be dismissed or the evidence gathered may not be admissible in court.

A good warning banner should parallel your Corporate Security Policy. At a minimum, it needs to provide legal protection to allow prosecution of intruders, protect your company from liability, warn of monitoring and recording usage, and be general enough as to not provide additional information that can be used to further the attack.

The warning banner should communicate the following:

- State that use is for *authorized users only* and that the system is *restricted to official authorized corporate work only*. Offenders often claim that they had no knowledge of who should be using the system or what work is allowed on the systems. Do not start off by saying "Welcome...". This can be construed as an invitation to use the system.
- State that all usage *may be monitored and/or recorded*. Clearly state that there is *no expectation of privacy* with use of the system.
- State that all suspicious activity, abuse, and/or criminal activity *may be handed over to law enforcement* for further investigation or prosecution.
- State that *use of the system implies consent* to all conditions of the warning banner.
- Do not include company information, system administrator information, and router/switch information in the warning banner. This information may give the attacker additional information to successfully infiltrate your network resources. This includes company name, company address, system administrator name/contact information, location, type and version of router or switch.

Example of Warning Banner

Warning Notification!!!

This system is to be used by authorized users only for the purpose of conducting official company work. Any activities conducted on this system may be monitored and/or recorded and there is no expectation of privacy while using this system. All possible abuse and criminal activity may be handed over to the proper law enforcement officials for investigation and prosecution. Use of this system implies consent to all of the conditions stated within this Warning Notification.

NOTE: Application – Low, Medium, and High Security Models.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Setting Up A Warning Banner

All Foundry routers and switches support the implementation of Warning Banners. The following types of warning banners are supported and all three types should be set to cover all access methods.

banner exec

Configures the Foundry device to display a message when a user enters the Privileged EXEC CLI level.

Syntax: [no] banner exec <delimiting-character>

EXAMPLE:

```
BigIron(config)# banner exec $ (Press Return)
Enter TEXT message, End with the character '$'.
Warning Notification!!!
This system is to be used by authorized users only for company work.
Activities conducted on this system may be monitored and/or recorded with
no expectation of privacy. All possible abuse and criminal activity may be
handed over to the proper law enforcement officials for investigation and
prosecution. Use implies consent to all of the conditions stated within
this Warning Notification. $
```

banner incoming

Configures the Foundry device to display a message on the Console when a user establishes a Telnet session. This message indicates where the user is connecting from and displays a configurable text message.

Syntax: [no] banner incoming <delimiting-character>

EXAMPLE:

```
BigIron(config)# banner incoming $ (Press Return)
Enter TEXT message, End with the character '$'.
Remote TELNET session from above host. $
```

banner motd

Configures the Foundry device to display a message on a remote user's terminal when they establish a Telnet CLI session.

Syntax: [no] banner <delimiting-character> | [motd <delimiting-character>]

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

```
BigIron(config)# banner motd $ (Press Return)
Enter TEXT message, End with the character '$'.
Telnet Warning Notification!!!
This system is to be used by authorized users only for company work.
Activities conducted on this system may be monitored and/or recorded with
no expectation of privacy. All possible abuse and criminal activity may be
handed over to the proper law enforcement officials for investigation and
prosecution. Use implies consent to all of the conditions stated within
this Warning Notification. $
```

NOTE: Check with your local or state's cybercrime department to see what laws are in place to protect against data security crimes and to check for additional information with regards to Warning Banners.

Configuring Access

Foundry devices support a wide range of accessibility and management features. Accessibility is controlled through authentication of the user and management abilities are controlled through authorization of privileges. Depending on your security requirements, the types of authentication and authorization security features will vary for each device. Perimeter devices used to attach to the Internet and create DMZ and screened subnets will require much more stringent security features than internal devices and should be treated as *High Security* devices.

Remember that security and usability are trade offs – the higher the security requirement, the less user friendly or accessible the device usually becomes. At a minimum, all Foundry devices should be protected with some form of Basic Access security to protect against unauthorized access and usage. You must review your company's Security Policy and determine what types of authentication and authorization are required on your Foundry devices. All unnecessary access methods should be disabled and all required methods should be protected and monitored adequately.

Foundry devices support the following connectivity methods:

- Console port – requires physical access to the device through a serial connection
- Remote Connection – requires a remote access protocol such as Telnet or SSH to access the Privileged EXEC and CONFIG levels of the CLI
- Web Connection – requires supported Web browser software and http service running on the Foundry device
- SNMP – requires compatible SNMP management interface such as IronView for read-only and read-write access
- TFTP – requires TFTP server to transfer configuration and OS code files

Authentication to Foundry devices may be completed through any of the following methods:

- Privilege EXEC and CONFIG password
- Default privilege level password

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



- Username and password
- TACACS (Terminal Access Controller Access Control System)
- AAA (authentication, authorization, accounting) with TACACS+, RADIUS

Remote Administration Concerns

By using any method to remotely connect to Foundry devices, you automatically expose the device to some risk. Common remote access methods include Telnet, SSH, and Web management and each have their advantages and weaknesses. There are many ways to make these remote access methods more secure and this "best practices" document will address each of them. The concerns associated with remote access include:

Management IP Address Spoofing	The management station's IP address can be spoofed with crafted packets that fake the source IP address of the management station. The intruder hopes to fool any address restriction rules on the device to gain access.
Password Attacks	Hackers attempting to gain access to network devices can use brute force applications to try to break simple password controls. By default, many routers and switches do not limit the number of unsuccessful login attempts and do not log the unsuccessful entries.
Password Sniffing	There are many "sniffer" type of applications that are freely available on the Internet, which may be used to capture packets traveling across the network. A simple feature of these applications is to capture packets based on signatures within the packet – such as the words login or password. This poses significant risks when using unencrypted remote control methods to access network devices.
Session Hijacking	Session hijacking occurs when a hacker uses tools to take over your TCP connection to the device to which you are authenticated. The hacker sends jamming packets to your management workstation and configures their management host to assume your IP address.
Authentication Server Compromise	Authentication methods which use third-party authentication servers such as TACACS, TACACS+, and RADIUS are at risk if the authentication server can be compromised by an attacker and the authentication database manipulated. Passwords can be changed and unauthorized accounts can be added to gain access.
Dial-In Access	Never hook analog modems up to the console ports to supply remote dial-in access to Foundry devices. It is better to dial into a secure Remote Access Server that uses strong authentication and then access the device using a secure management protocol such as SSH. Analog modems do not support strong authentication or encryption.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Determining What To Implement

The following are best practices to implement to harden Foundry routers and switches for access and authority control. Combinations of security features can be implemented together to create a strong defense for accessing network devices. What your company decides to implement will depend largely on your Corporate Security Policy and strategy. Use this guide as a tool to show you the many possibilities of using Foundry's security features.

Low Security Model Requirements - The following security features may be implemented to restrict access to the Foundry device.

- Enable Privilege EXEC and CONFIG level passwords for each device.
- For installations that require granular access privileges, create additional accounts and assign privileges for better logging and granular control.
- Setup Telnet passwords to force authentication of Telnet sessions.
- Force Telnet authentication through local user database for better tracking of user access.
- Limit Telnet, SSH, and Web management sessions to well known management IP addresses using ACLs or built-in commands.
- Specify a minimum password length.

Medium Security Model Requirements - The following security features may be implemented to restrict access to the Foundry device – in addition to methods mentioned in the *Low Security* model.

- Use of local accounts and passwords to authenticate and authorize use. TACACS, TACACS+, or RADIUS for large environments with more than 16 administrators.
- Telnet access should be restricted with ACLs or built-in commands. Telnet can be optionally disabled in favor of SSH to encrypt the management connection.
- If Telnet isn't used, suppress the rejection messages to limit the amount of information that can be groomed for an attack.
- Web management should be restricted with ACLs or built-in commands. Web management can optionally be disabled for heightened security.
- Router configurations should not be downloaded from a TFTP server if possible. If they have to, make sure the TFTP server is well secured and ACLs are used to limit the TFTP servers used.
- Limit the remote management hosts to well-known management IP addresses through ACLs or built-in commands.

High Security Model Requirements - The following security features may be implemented to restrict access to the Foundry device in addition or in place of the methods mentioned in the *Low and Medium Security* models.

- Use of TACACS, TACACS+ or Radius for system authentication with separate authorization privileges to control the level of access each person has to the device.
- Configure secure secondary authentication methods in case primary methods are not available – such as username and passwords for each user.
- Telnet access should be disabled in favor of SSH to encrypt the management connection.
- If Telnet isn't used, suppress the rejection messages to limit the amount of information that can be groomed for an attack.
- Web management should be disabled or limited to specific management addresses.
- Determine if SNMP is needed and disable the service if it isn't required.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



- Router configurations should not be downloaded from a TFTP server at boot if possible. This poses a security risk due to the inherent low security features of TFTP.
- Setup separate port-based VLANs for managing network devices.
- Limit the amount of remote administration to the device whenever possible – the most secure model is to eliminate it all together if possible. Use ingress ACLs to the management IP to limit management protocols or lock SSH and Telnet sessions down to known management IP addresses. Implement management VLANs and lock remote access services down to a particular management VLAN.

Selection of Strong Passwords

Foundry supports a wide range of security features that are used to authenticate access into the device. Because many of these authentication mechanisms rely on passwords, it is imperative that strong passwords be selected. Security is only as strong as the “weakest link” and weak passwords that are easily guessed or cracked by brute force password crackers will bypass all the other security features implemented. Encryption should always be considered as part of password security as most passwords used in management tools are passed in clear text – Telnet, SNMP v1, Web management, etc.

Strong passwords must meet two fundamental requirements:

- They are tough to guess or crack
- They should be easy to remember so they don’t have to be written down

Common words found in the dictionary should not be used as passwords, as password crackers use dictionaries as a source for their brute force attacks. Passwords should not be words from a foreign dictionary either as most password crackers have the ability to use foreign dictionaries. The best way to create and remember passwords that are tough to crack but are still easy to remember is to use “pass phrases”.

Pass phrases are whole sentences or parts of a sentence that can be used to form the password by selecting characters from each word to form the password. Ideally, it should contain numbers as well as case sensitivity to make it harder to crack. An example of creating a password from a pass phrase is:

Foundry has great layer 3 products for you!

By taking the first letter in each word and substituting the number “4” for the word “for”, we can extract the following strong password: Fhgl3p4u!

Other substitutions for common letters and words to use are:

The number “3” for the letter “E”
 The number “2” for the word “to” or “too”
 Use the “@” symbol for the letter “a”
 The symbol “+” for the letter “T”
 The symbol “!” for the number 1
 The symbol “\$” for the letter “S”

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Whenever a pass phrase can't be used to form a password, select "easy to remember" passwords that are at least 6 – 8 characters long and are not part of any dictionaries.

For large enterprises with many devices and related passwords, a formula or algorithm should be used to come up with passwords that make sense – making it easier to remember them. For example, code names can be used for departments or geographical locations and the letters forming the code names can be used to start the passwords. Never increment your passwords by assigning numbers to the end of them. This will weaken your password strategy.

Coming up with strong password schemes for large organizations with hundreds, if not thousands of devices, is a difficult task. If you end up documenting your passwords, make sure the file is fully password protected, encrypted with an application such as PGP, and stored on a hardened server with minimal access. If you can, use only hints to the real password in the password file and not the actual password.

NOTE: Rules for strong passwords should also be applied when setting up SNMP public and read-write community strings. SNMP community strings are passed in clear text in SNMP v1. Use SNMP v3 if possible.

Configuration File Security

Configuration files store the complete setup of your Foundry devices including the passwords. By default, Foundry devices store all passwords in encrypted form but there are commands to stop the encryption of passwords – this may expose critical system passwords stored in backup configuration files. Best practices dictate that you should always have a minimum of one backup copy of all Foundry device configuration files on hand for restore purposes.

Never keep backup configuration files on an insecure system. You should create a secure directory on a secure hardened server with the minimum set of access privileges to store your backup copies. Use encryption programs such as PGP to encrypt the configuration files to give them another layer of protection so they cannot be viewed easily.

Never keep your backup configuration files on your TFTP server. After the configuration files are backed up onto the TFTP server, transfer them to a secure file server and delete them from the TFTP server directory and any "trash bins". Encrypt the configuration files on the secure server.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Authentication and Authorization Features

Authentication and authorization of users is one of the most important security defenses to apply in hardening your network devices. There are a number of ways to implement authentication and authorization security ranging from simple password protection for the Privilege EXEC and CONFIG access, to usernames and passwords, and strong third-party external authentication servers.

Please refer to your device's release notes and command line references for the proper syntax of each command. Depending on the hardware revision level, not all security features may be available.

Setting Up Privilege EXEC and Telnet Passwords

At a bare minimum, the Privileged EXEC and CONFIG level password and the Telnet Password (if Telnet is enabled) should be set on all devices. If you will be using stronger authentication methods such as AAA TACACS+ or RADIUS, you can still set a local super user password to provide a backup authentication method in case any of the other primary authentication methods are disabled or removed from the device. By default, Foundry devices do not have a super-user-password or the Telnet password set.

NOTE: Application – Low, Medium, and High Security Models for EXEC password. Telnet is only recommended in Low Security Models and SSH preferred in Medium to High Security models.

Syntax: [no] enable super-user-password <string>

Syntax: [no] enable telnet password <string>

EXAMPLE:

```
BigIron(config)# enable super-user-password super-user-password_text
BigIron(config)# enable telnet password telnet-password_text
```

Setting Telnet and Console Timeout Values

If you will be using Telnet as a remote access method to your Foundry devices, it is recommended that an idle timeout value be set for Telnet and the restriction of Telnet clients be setup using one of the recommended methods outlined in the section "Restricting Management Protocols"

Syntax: telnet-timeout <0 – 240 minutes>

EXAMPLE:

This example sets the Telnet timeout to ten minutes.

```
BigIron(config)# telnet-timeout 10
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



The serial console connection can also be programmed to timeout after a period of inactivity to prevent a console connection from being left on indefinitely. The default has no console timeout values set.

Syntax: [no] console timeout <0 – 240 minutes >

EXAMPLE:

This example sets the console timeout to ten minutes.

```
BigIron(config)# console timeout 10
```

Setting Up Privilege Level Passwords

Privilege Level Passwords can be assigned to each of the three default access levels: Super User level, Port Configuration level, and Read Only level. Anyone who knows the password for the privilege level is granted access at that level. Because this method of authentication does not take usernames into consideration, log entries are generic in nature when access is granted. Although this authentication method may be feasible in installations where there are a limited number of device administrators, it should be not be used in environments where granular authentication and accountability is required.

NOTE: Although it is possible to assign just one password to each of the default privilege levels (Super User, Port Configuration, Read Only), it is recommended that user accounts be created and privilege levels be assigned to each person accessing the device. This creates better accountability as the names of the user accounts are logged with the date and time of access.

NOTE: Application – Low Security Model or installations with very few device administrators. Medium to High Security Models should use AAA authentication methods.

Default Management Privilege Levels

- Super User level - Allows complete read-and-write access to the system. This is generally for system administrators and is the only management privilege level that allows you to configure passwords.
- Port Configuration level - Allows read-and-write access for specific ports but not for global (system-wide) parameters.
 - The User EXEC and Privileged EXEC levels
 - The port-specific parts of the CONFIG level
 - All interface configuration levels
- Read Only level - Allows access to the Privileged EXEC mode and CONFIG mode of the CLI but only with read access.

NOTE: You must use the CLI to assign a password for management privilege levels. You cannot assign a password using the Web management interface.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Syntax: [no] enable <privilege-level>-password <password-string>

EXAMPLE:

This example creates an access password for each default Management Privilege Level.

```
BigIron(config)# enable super-user-password <text>
BigIron(config)# enable port-config-password <text>
BigIron(config)# enable read-only-password <text>
```

Setting Up User Accounts and Privilege Levels

For small to medium sized companies that have less than 16 system administrators and low to medium security model requirements, Foundry's local authentication functions can be used to create user accounts and access privileges for each system administrator.

NOTE: Application – Low through Medium Security Models. Secondary authentication mechanism in High Security Model

Syntax: [no] username <user-string> privilege <privilege-level> password | nopassword <password-string>

The **privilege** parameter specifies the privilege level for the account. You can specify one of the following:

- **0** - Super User level (full read-write access)
- **4** - Port Configuration level
- **5** - Read Only level

The default privilege level is **0**.

EXAMPLE:

This example will setup four separate user accounts and assign them with the proper access privileges. The device will then force all Telnet, Web, and Privilege EXEC and CONFIG access to authenticate against the device's local user accounts.

Paul	System Administrator with all rights
Jane	System Administrator with all rights (backup to Paul)
Andy	Desktop Support with port configuration access only
Brad	Unix Administrator with read only access rights

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



First, create the accounts with the associated privilege levels.

```
BigIron(config)# username paul privilege 0 password paul's_password
BigIron(config)# username jane privilege 0 password jane's_password
BigIron(config)# username andy privilege 4 password andy's_password
BigIron(config)# username brad privilege 5 password brad's_password
```

Next, implement one or more authentication methods for securing access to the device by configuring authentication-method lists that set the order in which the authentication methods are used. In an authentication-method list, you specify the access method (Telnet, Web, SNMP, and so on) and the order in which the device tries one or more of the following authentication methods:

- Local Telnet login password
- Local password for the Super User privilege level
- Local user accounts configured on the device
- Database on a TACACS or TACACS+ server
- Database on a RADIUS server
- No authentication

Syntax: [no] aaa authentication snmp-server | web-server | enable | login default <method1> [<method2>] [<method3>] [<method4>] [<method5>] [<method6>] [<method7>]

The **snmp-server** | **web-server** | **enable** | **login** parameter specifies the type of access this authentication-method list controls. You can configure one authentication-method list for each type of access.

The <method1> parameter specifies the primary authentication method. The remaining optional <method> parameters specify additional methods to try if an error occurs with the primary method. For a complete listing of <methods> refer to your devices setup documentation.

EXAMPLE:

This example enables Telnet Authentication which forces Telnet connections to use the local user database and forces the Web management and Privileged EXEC and CONFIG levels of the CLI to use the local user database for authentication.

```
BigIron(config)# enable telnet authentication
BigIron(config)# aaa authentication web-server default local
BigIron(config)# aaa authentication enable default local
```

Restricting Management Protocols

With all three security model levels (low, medium, and high), it is recommended that access to the remote management protocols be limited for Foundry devices. These restriction features can be applied to each device individually to create the necessary level of access control for your environment. Management protocols such as Telnet, SSH, Web management, SNMP, and TFTP can be restricted through one or more of the following methods:

- Access Control Lists (ACLs)
- Built-in commands
- Port Based VLAN (management VLAN)

Limiting Remote Access To Specific IP Addresses With ACLs

You can use several different methods to lock remote access management protocols to specific IP addresses: ACLs can be used to limit Telnet, SSH, Web, and SNMP access to specific IP addresses. Foundry built-in CLI commands can be used to limit Telnet, SSH, and SNMP access (but not Web) to IP addresses.

Using ACLs to limit access is more flexible than using the associated CLI command to limit remote access methods. ACLs allow more than 10 devices to be granted whereas the CLI command is limited to 10 remote IP addresses. Telnet, SSH, and Web access can be controlled with either one set of ACLs (if all the management hosts are the same) or by using one set of ACLs for each remote management protocol. SNMP ACLs are applied to the community string that is used to gain access.

NOTE: Application – Low, Medium, and High Security Models

Syntax: telnet access-group <num>

Syntax: ssh access-group <num>

Syntax: web access-group <num>

Syntax: snmp-server community <string> ro | rw <num>

The <num> parameter specifies the number of a standard ACL and must be from 1 - 99.

EXAMPLE:

This example configures five separate Access Lists for use with each remote access method. Using Foundry's default implicit "deny", only the hosts that are permitted to use each remote access method is specified in the access list.

```
BigIron(config)# access-list 10 permit host 10.1.0.25
BigIron(config)# access-list 10 permit 10.2.1.0 0.0.0.255
BigIron(config)# access-list 10 permit 10.2.3.0 0.0.0.255
BigIron(config)# access-list 10 permit 10.2.5.0/24
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



```
BigIron(config)# access-list 11 permit host 10.1.0.25
BigIron(config)# access-list 11 permit host 10.3.1.15

BigIron(config)# access-list 12 permit host 10.3.0.27
BigIron(config)# access-list 12 permit host 10.3.1.15

BigIron(config)# access-list 13 permit host 10.5.0.1
BigIron(config)# access-list 13 permit host 10.5.1.2

BigIron(config)# access-list 14 permit host 10.5.0.3
BigIron(config)# access-list 14 permit host 10.5.1.4

BigIron(config)# ssh access-group 10
BigIron(config)# telnet access-group 11
BigIron(config)# web access-group 12
BigIron(config)# snmp-server community public ro 13
BigIron(config)# snmp-server community private rw 14
BigIron(config)# write memory
```

NOTE: By default, Foundry ACLs uses an **implied deny** statement at the end of each ACL.

Limiting Remote Access To Specific IP Addresses With Built-in Commands

By default, Foundry devices do not control remote management access based on the managing station's IP address. Foundry CLI commands can be used to limit Telnet, SSH, and SNMP (but not Web) to IP addresses.

A maximum 10 IP addresses can be restricted to each remote service by repeating the command up to 10 times for each service.

NOTE: Application – Low, Medium, and High Security Models

Syntax: [no] telnet-client <ip-addr>

Syntax: [no] web-client <ip-addr>

Syntax: [no] snmp-client <ip-addr>

EXAMPLE:

This example uses the built-in commands to limit Telnet on this device to three specific hosts, Web management to two specific hosts, and SNMP to two specific hosts.

```
BigIron(config)# telnet-client 10.1.0.25
BigIron(config)# telnet-client 10.1.0.30
BigIron(config)# telnet-client 10.1.0.35
```


WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



```
BigIron(config)# web-client 10.5.2.12
BigIron(config)# web-client 10.5.2.13

BigIron(config)# snmp-client 10.6.1.30
BigIron(config)# snmp-client 10.6.1.45
```

If all the remote management hosts for all three remote access protocols (Telnet, Web, and SNMP) are the same, you can lock down Telnet, SSH, and SNMP access to the device with one command:

Syntax: [no] all-client <ip-addr>

EXAMPLE:

```
BigIron(config)# all-client 10.1.0.25
BigIron(config)# all-client 10.1.0.30
```

Limiting Remote Access To A Specific VLAN

Management VLANs are a popular way to separate your management traffic from your regular data traffic on your network. It allows the devices to be managed from a secure (or more secure) network and allows stronger restriction ACLs to be applied to further restrict accessibility to the Foundry device. By default, remote management access is allowed on all ports. Once you configure security for a given access method based on VLAN ID, access to the device is restricted to the ports within the specified VLAN.

If other methods of restricting management access are also used, both conditions must be true before access is granted. For example, suppose you configure an ACL to permit Telnet access only to specific client IP addresses, and you also configure VLAN-based access control for Telnet access. In this case, the only Telnet clients that can access the device are clients that have one of the permitted IP addresses as specified by the ACL *and* connected to a port that's a member of an authorized VLAN. Clients with a permitted IP address but are connected to a port that is not in a permitted VLAN still cannot access the device through Telnet.

The following management protocols can be restricted to port-based VLANs on Foundry devices:

- Telnet access
- Web management access
- SNMP access
- TFTP access

NOTE: Application – Medium through High Security Models

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Syntax: [no] telnet server enable vlan <vlan-id>

Syntax: [no] web-management enable vlan <vlan-id>

Syntax: [no] snmp-server enable vlan <vlan-id>

Syntax: [no] tftp client enable vlan <vlan-id>

EXAMPLE:

This example creates two Layer 3 port-based VLANs and restricts the Telnet and Web management clients to a Port Based VLAN with the ID of 10 and restricts access from SNMP and TFTP clients to a port-based VLAN with the ID of 40.

```
BigIron(config)# vlan 10 by port
BigIron(config)# untagged e1 to 2
BigIron(config)# router-interface ve 10
BigIron(config)# span

BigIron(config)# vlan 40 by port
BigIron(config)# untagged e4 to 5
BigIron(config)# router-interface ve 40
BigIron(config)# span

BigIron(config)# telnet server enable vlan 10
BigIron(config)# web-management enable vlan 10
BigIron(config)# snmp-server enable vlan 40
BigIron(config)# tftp client enable vlan 40
```

For this example, Telnet clients that are on the ports supported by VLAN 10 will be granted access. All other Telnet clients from ports outside of VLAN 10 will receive an error message similar to the following:

```
You are not authorized to telnet to this box! Bye.
Connection to host lost.
C:\>
```

NOTE: Please refer to your device's release notes and command references for setting up the proper L2 and L3 Port Based VLANs to support the restriction of management protocols. Setup of L2 VLANs varies slightly due to the management IP address being global for all ports on the L2 device.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Configuring Advanced Access

For installations requiring advanced authentication methods for their network devices, Foundry supports two AAA alternatives using industry standard TACACS+ and RADIUS. AAA access control stands for authentication, authorization, and accounting. By using one of these AAA methods, centralized authentication can be created to support all network devices that support these AAA standards. With both AAA access methods, a third-party access control server (ACS) running TACACS+ or RADIUS is required.

The benefits of using AAA access are:

- Scalability of authentication for all networked devices
- Centralized username and password control
- Centralized logging of activities
- Centralized authorization of privileges

NOTE: Application – Medium through High Security Models.

TACACS/TACACS+

TACACS and TACACS+ can be used to authenticate the following types of remote access to Foundry devices:

- Telnet access
- SSH access
- Web management access
- Access to the Privileged EXEC level and CONFIG levels of the CLI

TACACS uses the UDP protocol to communicate between the device and the TACACS server and was originally developed by BBN for MILNET. TACACS is not considered to be an AAA access control method and is considered to be less attractive than TACACS+ due to the advanced features and reliability of TACACS+. Foundry provides TACACS for backward compatibility for enterprises still using TACACS in their environment. It is highly advisable to upgrade all TACACS devices to one of the more secure AAA authentication methods for better security and accountability, as TACACS is considered end-of-maintenance by other vendors such as Cisco.

TACACS+ was developed by Cisco as an enhancement to TACACS and includes the following benefits:

- Uses the TCP protocol for more reliable communications.
- Separates the functions of authentication, authorization, and accounting allowing separate servers to control specific functions.
- Encrypts the conversations between the device and the TACACS+ server.
- Allows for arbitrary length and content authentication exchanges.
- Extensible to allow for customization and feature development.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



TACACS+ provides flexible authentication and allows granular access control for each user authenticated. Because TACACS+ separates the authentication, authorization, and accounting features, you can implement any one or all of the AAA features.

If you are using TACACS+, Foundry recommends that you also configure **authorization**, in which the Foundry device consults a TACACS+ server to determine which management privilege level (and which associated set of commands) an authenticated user is allowed to use. You can also optionally configure **accounting**, which causes the Foundry device to log information on the TACACS+ server when specified events occur on the device.

NOTE: For a complete overview on TACACS/TACACS+ please refer to the *Foundry Security Guide*. It contains a thorough implementation guide with all of the supported options.

TACACS/TACACS+ Requirements

If you are going to use TACACS or TACACS+ for authenticating your Foundry devices, the following points must be noted:

- You must have at least one TACACS/TACACS+ server on your network.
- Foundry devices will support up to eight TACACS/TACACS+ servers and will authenticate against them in the order they were added to the devices' configuration.
- Only one primary authentication method is supported for each of the access methods (Telnet, SSH, Web management, Privilege EXEC and CONFIG).
- Backup authentication methods can be used in case the primary method is unavailable.
- You can configure the Foundry device to authenticate using a TACACS or TACACS+ server, but not both.

TACACS/TACACS+ Setup Procedures

The first three steps are identical to both TACACS and TACACS+. Steps 4 and 5 are only for TACACS+ installations and allow you to optionally setup TACACS+ Authorization and Accounting.

Step 1: Configuring Access Control Server's

The first step is to configure the Access Control Servers (ACS), TACACS/TACACS+ server(s), which will be used to authenticate users from the Foundry devices. Up to eight ACS's can be specified and the Foundry device will contact each in the order that they were configured.

Syntax: `tacacs-server <ip-addr>|<hostname> [auth-port <number>]`

EXAMPLE:

This example configures three TACACS servers for authenticating this Foundry device using the default port number (49). The Foundry device will attempt to authenticate each server in the order they were configured into the device.

```
BigIron(config)# tacacs-server host 207.94.6.161
BigIron(config)# tacacs-server host 207.94.6.191
BigIron(config)# tacacs-server host 207.94.6.122
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



For TACACS+, you can also specify a specific TACACS+ server for each of the AAA functions – authentication, authorization and accounting. Use the following command to setup different AAA functions on different TACACS+ servers.

Syntax: tacacs-server host <ip-addr> | <server-name> [authentication-only | authorization-only | accounting-only | default] [key <string>]

EXAMPLE:

This example configures the Foundry device to use a different TACACS+ server for each AAA function and specifies the shared key for each server. The default port of (49) will be used.

```
BigIron(config)# tacacs-server host 10.2.3.4 authentication-only key abc
BigIron(config)# tacacs-server host 10.2.3.5 authorization-only key def
BigIron(config)# tacacs-server host 10.2.3.6 accounting-only key ghi
```

Step 2: Setting Optional TACACS/TACACS+ Parameters

The second step is to specify any additional settings for communications to the ACS server. The optional parameters are:

Secret Key	The TACACS+ secret key used to encrypt the packets transmitted between the Foundry device and the TACACS+ server. This is a text string that matches the TACACS+ server's secret key.
The Retransmit Interval	The number of times the Foundry device will attempt to authenticate against the ACS. The default is 3 times and the acceptable range is 1 – 5 times.
Dead Time	The number of seconds the Foundry device will wait until it assumes the ACS is unavailable and moves onto the next ACS programmed in the device. The default is 3 seconds and the acceptable range is 1 – 5 seconds.
Timeout	The number of seconds the Foundry device will wait before retransmitting or assuming the ACS is not available. The default is 3 seconds and the acceptable range is 1 – 15 seconds.

Syntax: tacacs-server key [0 | 1] <string>

Syntax: tacacs-server retransmit <number>

Syntax: tacacs-server dead-time <number>

Syntax: tacacs-server timeout <number>

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



The most important optional parameter applies to TACACS+ implementations only – the Secret Key must be used to configure the Foundry device to authenticate against the TACACS+ server. Both secrets must match exactly for encryption and authentication to be successful. TACACS implementations do not require the key to be set.

EXAMPLE:

This example sets the Secret Key to be used with the TACACS+ server and encrypts the key so it will not be displayed in the configuration. The "1" enables encryption of the key and "0" disables encryption. By default, secret keys are encrypted.

```
BigIron(config)# tacacs-server key 1 abc
BigIron(config)# write terminal
...
tacacs-server host 1.2.3.5 auth-port 49
tacacs key 1 $!2d
```

Step 3: Configuring Authentication-Method Lists for TACACS/TACACS+

The third step is to configure the Foundry device with the access methods to assign to the TACACS or TACACS+ servers. Foundry devices support one primary authentication method and up to six backup authentication methods, in case the primary isn't available.

You can assign the following access methods to use TACACS/TACACS+ server as an authentication method:

- Telnet access
- SSH access
- Privilege EXEC and CONFIG CLI access
- Web Management access

One authentication method must be used for each access method that is assigned to the ACS server.

Syntax: [no] aaa authentication enable | login | web-server default <method1> [<method2>] [<method3>] [<method4>] [<method5>] [<method6>] [<method7>]

The following are the valid <methods> that can be used to authenticate the access methods. Only one of these can be used as the primary access method and up to six backup methods can be set.

line	Authenticate using the password you configured for Telnet access. The Telnet password is configured using the enable telnet password... command.
enable	Authenticate using the password you configured for the Super User privilege level. This password is configured using the enable super-user-password... command.
local	Authenticate using a local user name and password you configured on the device. Local user names and passwords are configured using the username... command.
tacacs	Authenticate using the database on a TACACS server. You also must identify the server to

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



	the device using the tacacs-server command.
tacacs+	Authenticate using the database on a TACACS+ server. You also must identify the server to the device using the tacacs-server command.
radius	Authenticate using the database on a RADIUS server. You also must identify the server to the device using the radius-server command.
none	Do not use any authentication method. The device automatically permits access.

Telnet & SSH EXAMPLE:

This example sets the Telnet access method to use the TACACS/TACACS+ server as the primary authentication method, the local user accounts as the second authentication method, and the enable password as the third authentication method. Both Telnet and SSH uses the "login default" parameter.

```
BigIron(config)# enable telnet authentication
BigIron(config)# aaa authentication login default tacacs local enable
```

Privilege EXEC and CONFIG EXAMPLE:

This example configures the device to use the TACACS/TACACS+ ACS's as the primary authentication method with the local usernames as the secondary method and the Telnet line password as the third method.

```
BigIron(config)# aaa authentication enable default tacacs local line
```

Web Management EXAMPLE:

This example setups up the primary authentication method as the TACACS/TACACS+ server with the local user accounts as the secondary backup authentication method.

```
BigIron(config)# aaa authentication web-server default tacacs local
```

Step 4: Setting UP TACACS+ Authorization Levels

As mentioned earlier, TACACS+ allows the separation of the AAA features and permits each access level to be authenticated before access is granted. If TACACS+ is used in your environment, it is recommended that authorization levels be created and enforced for additional security.

TACACS+ server use a response called the A-V (Attribute-Value) pair to send the privilege level of the user back to the Foundry device. The device extracts the A-V pair configured for the Exec service and uses it to determine the user's privilege level. To configure authorization using TACACS+, first enable the authorization service for TACACS+ on the Foundry device and then setup each user's access privileges on the TACACS+ server.

Enabling Authorization For EXEC Privilege Access

On the CLI of the Foundry device, enable the authorization feature for the EXEC service. This will instruct the device to check the TACACS+ server for the privilege level of the user logging into the device.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Syntax: aaa authorization exec default tacacs+ | none

Using the "none" parameter or omitting the "aaa authorization" statement performs no authorization check.

EXAMPLE:

This example enables authorization checking for the device and uses the TACACS+ server's A-V parameter to determine the privilege level of the user.

```
BigIron(config)# aaa authorization exec default tacacs+
```

Configuring User EXEC Privilege Levels

Each user using TACACS+ for authentication should have their privilege levels set on the TACACS+ server. To set a user's privilege level, you can configure the "foundry-privlvl" A-V pair for the Exec service on the TACACS+ server. The following default privilege levels are available.

- 0 - Super User level (full read-write access)
- 4 - Port Configuration level
- 5 - Read Only level

EXAMPLE:

This example sets the EXEC privilege level to "0" for a user account named "bob".

```
user=bob {
  default service = permit
  member admin
  # Global password
  global = cleartext "cat"
  service = exec {
    foundry-privlvl = 0
  }
}
```

To be compatible with Cisco privilege levels, the following mappings are made for non-Foundry A-V pair values.

Value for non-"foundry-privlvl" A-V Pair	Foundry Privilege Level
15	0 (super-user)
From 14 - 1	4 (port-config)
Any other number or 0	5 (read-only)

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

This example will use the Foundry Privilege Level of "0-SuperUser" for a non-Foundry Privilege Level of 15.

```
user=bob {
  default service = permit
  member admin
  # Global password
  global = cleartext "cat"
  service = exec {
    privlvl = 15
  }
}
```

Enabling Authorization For Command Access

For granular authorization of each command used by the user, enable the Foundry device's Command Authorization feature. Command Authorization is can configured for each of the user privilege levels.

Syntax: aaa authorization commands <privilege-level> default tacacs+ | radius | none

The <privilege-level> parameter can be one of the following:

- **0** - Authorization is performed for commands available at the Super User level (all commands)
- **4** - Authorization is performed for commands available at the Port Configuration level (port-config and read-only commands)
- **5** - Authorization is performed for commands available at the Read Only level (read-only commands)

NOTE: TACACS+ Command Authorization can be performed only for commands entered from Telnet sessions, SSH sessions, or from the console. No authorization is performed for commands entered at the Web management interface or IronView.

EXAMPLE:

This example instructs the Foundry device to use the TACACS+ server to authenticate each command used by privilege level "0-Super User" for Telnet or SSH sessions.

```
BigIron(config)# aaa authorization commands 0 default tacacs+
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Command Authorization & Accounting For Console Access

To use the TACACS+ server to authenticate each CLI command used on the console, enable the Console Command Authentication & Accounting feature with the following command:

Syntax: enable aaa console

EXAMPLE:

This example enables Command Authentication and Accounting for all commands used on the CLI console interface.

```
BigIron(config)# enable aaa console
```

Step 5: Configuring TACACS+ Accounting (optional)

An optional feature to setup on TACACS+ servers is accounting. If you decide to enable the Accounting feature, system events such as user logins, system reboots, and configuration changes are recorded on the TACACS+ accounting server. Accounting can be enabled for the following access methods:

- Telnet Authentication access
- SSH Authentication access
- CLI commands

If your company uses SYSLOG servers to record system events from Foundry devices, this step is optional or it can be used as a backup logging method.

Configuring Accounting for Telnet/SSH Access

Enabling this feature causes an Accounting Start packet to be sent to the TACACS+ server each time a user is authenticated to the device using Telnet or SSH access. When the user logs out, an Accounting Stop packet is sent to the TACACS+ server.

Syntax: aaa accounting exec default start-stop radius | tacacs+ | none

EXAMPLE:

This example configures the Foundry device to use TACACS+ as the accounting server for all start-stop accounting functions.

```
BigIron(config)# aaa accounting exec default start-stop tacacs+
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Configuring Accounting for CLI Commands

All CLI commands for a specific privilege level can be logged on the TACACS+ accounting server when they are executed. An Accounting Start packet will be sent to the TACACS+ server when the command is used and an Accounting Stop packet will be sent to the TACACS+ server when the results of the command are returned to the user. To enable this feature, use the following command:

Syntax: aaa accounting commands <privilege-level> default start-stop radius | tacacs+ | none

The <privilege-level> parameter can be one of the following:

- **0** - Records commands available at the Super User level (all commands)
- **4** - Records commands available at the Port Configuration level (port-config and read-only commands)
- **5** - Records commands available at the Read Only level (read-only commands)

EXAMPLE:

This example enables the accounting feature for all CLI commands in the privilege level "0-Super User". In effect, all commands will be logged since level 0-Super User includes all commands in the CLI.

```
BigIron(config)# aaa accounting commands 0 default start-stop tacacs+
```

Configuring TACACS+ Accounting for System Events

System events such as configuration changes or device reboots can be logged on the TACACS+ accounting server. An Accounting Start packet will be sent to the TACACS+ server when a system event occurs and an Accounting Stop packet will be sent when the system event is completed.

Syntax: aaa accounting system default start-stop radius | tacacs+ | none

EXAMPLE:

This example configures the Foundry device to send all system events to a TACACS+ Accounting server.

```
BigIron(config)# aaa accounting system default start-stop tacacs+
```

Configuring an Interface As The Source For All TACACS/TACACS+ Packets

On L3 devices that have many management addresses that can be used for remote access, it may be advantageous if a single source IP Address was configured for the entire device. If turned on, this feature will use the lowest-numbered IP address configured on the device as the source IP address for all TACACS/TACACS+ packets. By using a single IP address, the following security benefits can be achieved:

- You can configure the TACACS/TACACS+ server to accept packets from specific IP addresses to increase security and simplify configuration.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



- You can use a loopback interface as the single source IP so the TACACS/TACACS+ server can still receive packets if the link is down. This works only if there is a secondary link to the device.

Syntax: ip tacacs source-interface ethernet <portnum> | pos <portnum> | loopback <num> | ve <num>

The single IP address can be assigned to an Ethernet port, POS port, Loopback number, or virtual interface number.

EXAMPLE:

This example specifies the lowest-numbered IP address configured on a virtual interface as the device's source for all TACACS/TACACS+ packets. It first defines a virtual interface called "ve 1" and gives it an IP address of 10.0.0.3 with a 24 bit mask. Then it assigns the virtual interface as the source for all TACACS/TACACS+ packets from the L3 device.

```
BigIron(config)# int ve 1
BigIron(config-vif-1)# ip address 10.0.0.3/24
BigIron(config-vif-1)# exit
BigIron(config)# ip tacacs source-interface ve 1
```

Other Useful TACACS+ Commands

Some useful TACACS+ commands that can be used to view TACACS+ information include:

show aaa	Displays information about all TACACS+ and RADIUS servers on the device
show web	Displays the privilege level of Web management interface users

RADIUS AAA Security

RADIUS authentication, authorization, and accounting is very similar to TACACS+ in that it uses a third-party external Access Control Server (ACS) and security database to authenticate users, authorize their access privileges, and log accounting events. RADIUS was developed by Livingston Enterprises and is documented as RFC 2865. It is much more interoperable between different vendors than Cisco's TACACS+.

The differences between RADIUS and TACACS+ are:

- TACACS+ uses TCP and RADIUS uses UDP for communications between the device and the ACS server
- TACACS+ encrypts the entire packet and RADIUS encrypts only the password inside the access-request packets
- RADIUS combines the Authentication and Authorization functions but allows the separation of the Accounting function. TACACS+ allows the separation of all three AAA services.
- TACACS+ has multi-protocol support built in

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Despite some of the weaknesses of RADIUS, there are still many instances of the technology. With its standards-based protocol and multi-vendor support, it is more compatible across multiple platforms than TACACS+.

Foundry's RADIUS support allows a Foundry device to be configured to authenticate and authorize management sessions using any of the following remote access methods:

- Telnet access
- SSH access
- Web management access
- Privilege EXEC and CONFIG level CLI access

With RADIUS, the Accounting feature is optional. If you implement Authentication, you must implement Authorization as they are tied together. If you are using an external SYSLOG server to capture and centralize your device logs, the accounting feature of RADIUS is optional. RADIUS authentication and authorization is recommended for installations that have a medium to high security requirement.

NOTE: For a complete overview on RADIUS please refer to the Foundry Security Guide. It contains a thorough implementation guide with all of the supported options.

RADIUS Requirements

If you are going to use RADIUS for authenticating your Foundry devices, the following points must be noted:

- You must have at least one RADIUS Access Control Server on your network.
- Foundry devices will support up to eight RADIUS servers and will authenticate against them in the order they were added to the devices' configuration.
- Only one primary authentication method is supported for each of the access methods (Telnet, SSH, Web management, Privilege EXEC and CONFIG).
- Backup authentication methods can be used in case the primary method is unavailable.

RADIUS Setup Procedures

Setting up a RADIUS environment involves six steps.

Step 1: Configuring Foundry-Specific Attributes On The RADIUS Server

The first step is to configure the RADIUS server with the Foundry specific attributes that will be used by the Foundry device when authentication occurs. Upon successful authentication of the username and password supplied by the user, the RADIUS server will send back the following information in an Access-Accept packet:

- The privilege level of the user
- A list of commands
- Whether the user is allowed to use the commands in the list

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



This information is configured on the RADIUS server's individual or group profiles for each user that will access the Foundry device. Information you will need to complete this task is listed below.

Foundry's Vendor ID:		1991	
Vendor-Type:		1	
Attribute Name	Attribute ID	Data Type	Description
foundry-privilege-level	1	integer	<p>Specifies the privilege level for the user. This attribute can be set to one of the following:</p> <p>0 Super User level - Allows complete read-and-write access to the system. This is generally for system administrators and is the only management privilege level that allows you to configure passwords.</p> <p>4 Port Configuration level - Allows read-and-write access for specific ports but not for global (system-wide) parameters.</p> <p>5 Read Only level - Allows access to the Privileged EXEC mode and CONFIG mode of the CLI but only with read access.</p>
foundry-command-string	2	string	<p>Specifies a list of CLI commands that are permitted or denied to the user when RADIUS authorization is configured.</p> <p>The commands are delimited by semi-colons (;). You can specify an asterisk (*) as a wildcard at the end of a command string.</p> <p>For example, the following command list specifies all show and debug ip commands, as well as the write terminal command:</p> <p>show *; debug ip *; write term*</p>
foundry-command-exception-flag	3	integer	<p>Specifies whether the commands indicated by the foundry-command-string attribute are permitted or denied to the user. This attribute can be set to one of the following:</p> <p>0 Permit execution of the commands indicated by foundry-command-string, deny all other commands.</p> <p>1 Deny execution of the commands indicated by foundry-command-string, permit all other commands.</p>

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Step 2: Identifying the RADIUS Server to the Foundry Device

The second step is to configure the Foundry device with the RADIUS server information and specify any options that may be required to complete the authentication connection with the ACS. By default, the port number that is used for RADIUS authentication is UDP port 1645 and RADIUS accounting is UDP port 1646.

Syntax: radius-server host <ip-addr> | <server-name> [auth-port <number> acct-port <number>]

EXAMPLE:

The following example configures the Foundry device to use RADIUS server 209.157.22.99 using the default authentication and accounting port numbers.

```
BigIron(config)# radius-server host 209.157.22.99
```

Like TACACS+, RADIUS allows the separation of each of the AAA functions among different ACS servers. You can have authentication directed to a specific RADIUS server and the accounting function directed to a separate RADIUS server. Authorization is combined with the authentication server.

Syntax: radius-server host <ip-addr> | <server-name> [authentication-only | accounting-only | default] [key 0 | 1 <string>]

The **default** parameter causes the server to be used for all AAA functions.

EXAMPLE:

This example configures the Foundry device to use RADIUS server 10.2.3.4 as the authentication and authorization server and RADIUS server 10.2.3.5 as the accounting server. The key option is used to provide the RADIUS key that is used to authenticate the communications between the RADIUS server and the Foundry device.

```
BigIron(config)# radius-server host 10.2.3.4 authentication-only key abc
BigIron(config)# radius-server host 10.2.3.5 accounting-only key ghi
```

Step 3: Setting RADIUS Parameters

The third step allows you to configure the optional and required RADIUS parameters to communicate with the RADIUS server.

Radius Key

This is a required parameter and the string used here must match the key specified on the RADIUS server to complete the authentication channel. The key can be from 1 – 32 characters.

Retransmit Interval

Retransmit Interval is used to specify the number of times the Foundry device will resend an authentication request if the RADIUS server is not responding. The default is 3 times and valid range is 1 – 5 times.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Timeout Timeout is used to specify how many seconds to wait between Retransmit intervals. The default is 3 seconds and the valid range is 1 – 15 seconds.

Syntax: radius-server key [0 | 1] <string>

Syntax: radius-server retransmit <number>

Syntax: radius-server timeout <number>

The most important option is to setup the RADIUS key. By default, the RADIUS key will be encrypted within the configuration file to hide the value of the key when a “show config” is performed. If you want to unhide the key, specify the “0” value after the key.

EXAMPLE:

This example sets the RADIUS server key to “abc” and turns on the encryption feature to hide the key value from being displayed.

```
BigIron(config)# radius-server key 1 abc
BigIron(config)# write terminal
...
radius-server host 1.2.3.5
radius key 1 $!2d
```

Step 4: Configuring Authentication-Method Lists For Radius

The fourth step is to configure the Foundry device with the remote access methods that will be authenticated against the RADIUS servers. Foundry devices support one primary authentication method and up to six backup authentication methods, in case the primary isn’t available.

You can assign the following remote access methods to use a RADIUS server as an authentication method:

- Telnet access
- SSH access
- Privilege EXEC and CONFIG CLI access
- Web Management access

One authentication method must be used for each access method that is to be assigned to the ACS server.

Syntax: [no] aaa authentication enable | login | web-server default <method1> [<method2>] [<method3>] [<method4>] [<method5>] [<method6>] [<method7>]

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



The following are the valid <methods> that can be used to authenticate the remote access types. Only one of these can be used as the primary access method and up to six backup methods can be set.

line	Authenticate using the password you configured for Telnet access. The Telnet password is configured using the enable telnet password... command.
enable	Authenticate using the password you configured for the Super User privilege level. This password is configured using the enable super-user-password... command.
local	Authenticate using a local user name and password you configured on the device. Local user names and passwords are configured using the username... command.
tacacs	Authenticate using the database on a TACACS server. You also must identify the server to the device using the tacacs-server command.
tacacs+	Authenticate using the database on a TACACS+ server. You also must identify the server to the device using the tacacs-server command.
radius	Authenticate using the database on a RADIUS server. You also must identify the server to the device using the radius-server command.
none	Do not use any authentication method. The device automatically permits access.

Telnet & SSH EXAMPLE:

This example sets the Telnet access method to use the RADIUS server as the primary authentication method, the local user accounts as the second authentication method, and the enable password as the third authentication method. Both Telnet and SSH uses the "login default" parameter.

```
BigIron(config)# enable telnet authentication
BigIron(config)# aaa authentication login default radius local enable
```

Privilege EXEC and CONFIG EXAMPLE:

This example configures the device to use the RADIUS server as the primary authentication method with the local usernames as the secondary method and the Telnet line password as the third method.

```
BigIron(config)# aaa authentication enable default radius local line
```

Web Management EXAMPLE:

This example setups up the primary authentication method as the RADIUS server with the local user accounts as the secondary backup authentication method.

```
BigIron(config)# aaa authentication web-server default radius local
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Step 5: Configuring RADIUS Authorization

Radius authorization is configured to allow the CLI user privilege commands to be checked against the RADIUS server. This is primarily done through the default privilege levels of the Foundry device:

Syntax: aaa authorization commands <privilege-level> default radius | tacacs+ | none

The <privilege-level> parameter can be one of the following:

- **0** - Authorization is performed for commands available at the Super User level (all commands)
- **4** - Authorization is performed for commands available at the Port Configuration level (port-config and read-only commands)
- **5** - Authorization is performed for commands available at the Read Only level (read-only commands)

NOTE: RADIUS authorization can be performed only for commands entered from Telnet sessions, SSH sessions, or from the console. No authorization is performed for commands entered at the Web management interface or IronView.

EXAMPLE:

This example configures RADIUS authorization for all user privilege "0-Super User" commands. Essentially, all commands will be verified by the RADIUS server as the 0-Super User level contains all CLI commands.

```
BigIron(config)# aaa authorization commands 0 default radius
```

Step 6: Configuring RADIUS Accounting

This step allows you to optionally configure the RADIUS server accounting feature to log all system events. RADIUS accounting allows the centralization of device logging and can be used to help track anomalies or security incidents. RADIUS accounting can be turned on for the following access methods:

- Telnet access
- SSH access
- Privilege level CLI access
- System events

If your company uses SYSLOG servers to record events and system messages, RADIUS accounting is optional or can be used as a backup logging method.

Configuring RADIUS Accounting For Telnet/SSH Access

To configure the Foundry device to send system events to a RADIUS server when a Telnet or SSH session is established, use the following command. An Accounting Start packet is sent when the session is established and an Accounting Stop packet is sent when the user logs out.

Syntax: aaa accounting exec default start-stop radius | tacacs+ | none

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

```
BigIron(config)# aaa accounting exec default start-stop radius
```

Configuring RADIUS Accounting For Privilege Level CLI Commands

You can configure RADIUS accounting for CLI commands by specifying a privilege level whose commands are to be logged on the RADIUS accounting server. An Accounting Start packet is sent to the RADIUS server when a user enters a command and an Accounting Stop packet is sent to the server when the service provided by the command is completed.

Syntax: aaa accounting commands <privilege-level> default start-stop radius | tacacs | none

The <privilege-level> parameter can be one of the following:

- **0** - Records commands available at the Super User level (all commands)
- **4** - Records commands available at the Port Configuration level (port-config and read-only commands)
- **5** - Records commands available at the Read Only level (read-only commands)

EXAMPLE:

This example configures the Foundry device to perform RADIUS accounting for the commands available at the Super User privilege level (that is; all commands on the device)

```
BigIron(config)# aaa accounting commands 0 default start-stop radius
```

Configuring RADIUS Accounting For System Events

All system events such as device reboots and configuration changes can be sent to the RADIUS accounting server.

Syntax: aaa accounting system default start-stop radius | tacacs+ | none

EXAMPLE:

This example configures the Foundry device to send Accounting Start packets to the RADIUS server when a system event occurs. An Accounting Stop packet is sent when the system event is completed.

```
BigIron(config)# aaa accounting system default start-stop radius
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Configuring An Interface As The Source For All RADIUS Packets

You can configure a single source IP address for RADIUS packets if you need the following:

- Your RADIUS server is configured to accept traffic from only specific IP addresses.
- If you specify a loopback interface as the single source for RADIUS packets, RADIUS servers can receive the packets when network links are down. This allows the RADIUS server to continue to receive traffic from the client if the primary link between the two goes down and an alternate path is available.

An Ethernet port, POS port, Loopback number, or Virtual Interface number can be used to specify the source for all RADIUS packets from the device. The device will use the lowest-numbered IP address configured on the port or interface as the source IP address for the RADIUS packets.

Syntax: ip radius source-interface ethernet <portnum> | pos <portnum> | loopback <num> | ve <num>

EXAMPLE:

This example sets the lowest-numbered IP address configured on the virtual interface VE 1 as the source of all RADIUS packets for this device.

```
BigIron(config)# int ve 1
BigIron(config-vif-1)# ip address 10.0.0.3/24
BigIron(config-vif-1)# exit
BigIron(config)# ip radius source-interface ve 1
```

Other Useful RADIUS Commands

Some useful RADIUS commands that can be used to view RADIUS information include:

show aaa	Displays information about all TACACS+ and RADIUS servers on the device
show web	Displays the privilege level of Web management interface users

Implementing Secure Shell (SSH)

If you are using remote CLI access on your Foundry devices, it is highly recommended that Secure Shell (SSH) be used over Telnet. Telnet is prone to sniffing as all of its transmissions, including the password exchanges, are performed in clear text. It is only a matter of time before Telnet passwords are compromised in high security environments.

Foundry devices support SSH version 1 clients with a wide variety of encryption schemes. Although SSH v1 is still susceptible to threats such as session hijacking, it is still much more secure than Telnet sessions. SSH supports Arcfour, IDEA, Blowfish, DES (56-bit) and Triple DES (168-bit) data encryption methods. Nine levels of data compression are available. You can configure your SSH client to use any one of these data compression levels when connecting to a Foundry device.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



There are two types of SSH authentication supported on Foundry devices:

- ***RSA challenge-response authentication***, where a collection of public keys are stored on the device. Only clients with a private key that corresponds to one of the stored public keys can gain access to the device using SSH.
- ***Password authentication***, where users attempting to gain access to the device using an SSH client are authenticated with passwords stored on the device or on a TACACS/TACACS+ or RADIUS server

Both kinds of user authentication are enabled by default. You can configure the device to use one or both of them.

Foundry devices also support Secure Copy (SCP) to securely transfer files between a Foundry device and SCP-enabled remote hosts. If you have older Foundry equipment that doesn't support the SSH feature, you can still use SSH to strengthen your enterprise. To support these older devices, setup a secure SSH server on your trusted network in a secure subnet. SSH into this server and then Telnet from this secure server to the legacy device. Use ACLs or the built-in Telnet restriction command to lock down the Telnet client to the SSH server's IP address.

NOTE: Application – Low, Medium, and High Security Models. SSH is always preferred over Telnet for remote CLI access.

Configuring SSH On Foundry Devices

Configuring Secure Shell on a Foundry device consists of the following steps:

1. Setting the Foundry device's host name and domain name
2. Generating a host RSA public and private key pair for the device
3. Configuring RSA challenge-response authentication
4. Setting optional parameters

Foundry's SSH implementation supports many options for storing and loading of public keys. For a complete syntax of all related commands and differences of authentication mechanisms, please refer to your device's release note, command reference, and the *Foundry Security Guide*.

EXAMPLE:

This example will setup a Foundry device with all of the supported features of Foundry's SSH implementation. The host name will be "dmz_router1" and the domain name will be "abccorp.com".

Step 1: Setting The Host And Domain Name

Each Foundry device must have a host name and domain setup for SSH to function.

Syntax: hostname <name>

Syntax: ip dns domain-name <name>

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



```
BigIron(config)# hostname dmz_router1
BigIron(config)# ip dns domain-name abccorp.com
```

Step 2: Generating a Host RSA Key Pair

A public and private *host RSA key pair* is generated for the Foundry device when SSH is installed. The Foundry device uses this host RSA key pair, along with a dynamically generated *server RSA key pair*, to negotiate a session key and encryption method with the SSH client connecting to it. The host RSA key pair is stored in the Foundry device's system-config file but only the public key is readable.

Syntax: crypto key generate | zeroize rsa

The "generate" parameter is used to create the key pair and enables SSH on the Foundry device while the "zeroize" parameter is used to delete the key pair and disable SSH on the device.

```
dmz_router1(config)# crypto key generate rsa
dmz_router1(config)# write memory
```

NOTE: SSH host keys may take up to several minutes to create. If you are using SSH to connect to a Foundry device from a UNIX system, you may need to add the Foundry device's public key to a "known hosts" file.

For high security installations, the RSA host key pair can be hidden from the Running Config with the following command:

Syntax: ssh no-show-host-keys

Re-enable the displaying of the RSA host key pair with the following command:

Syntax: ssh show-host-keys

Step 3: Configuring RSA Challenge-Response Authentication

The 3rd step is to create the RSA public and private key pairs between the clients and the Foundry device. This step is only required if you will be using RSA Challenge-Response Authentication to validate SSH clients. If you have decided to use local usernames and passwords for authentication, this step is not necessary. You must remember to enable the desired authentication method as outlined in Step 4: Setting Optional Parameters.

Foundry devices store the client's public key physically on the device and use these keys to authenticate the client when a connection is requested. The two steps required to successfully setup the RSA challenge-response are:

- Importing the authorized public keys from the client(s) into the Foundry Device
- Enabling RSA challenge response authentication on the Foundry device

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Importing Public Keys

Create a text file containing the public keys from your SSH clients that support RSA authentication. Most SSH utilities supporting RSA authentication will have utilities that generate public/private key pairs – you'll need to gather the public key files from each client that will be accessing the Foundry device using RSA authentication.

The following is an example of a text file containing public keys from two SSH clients.

```
1024 65537 162566050678380006149460550286514061230306797782065166110686648548574
94957339232259963157379681924847634614532742178652767231995746941441604714682680
00644536790333304202912490569077182886541839656556769025432881477252978135927821
67540629478392662275128774861815448523997023618173312328476660721888873946758201
  user@csp_client
1024 35 152676199889856769693556155614587291553826312328095300428421494164360924
76207475545234679268443233762295312979418833525975695775705101805212541008074877
26586119857422702897004112168852145074087969840642408451742714558592361693705908
74837875599405503479603024287131312793895007927438074972787423695977635251943 ro
ot@unix_machine
```

Once you have constructed this file, you can upload it to the Foundry device from either a TFTP server or from a file on the Management IV module's PCMCIA flash. Foundry devices also support Secure Copy to securely transfer files between hosts on a network. Once the file is copied to one of the supported upload devices (TFTP server or PCMCIA flash), the file is automatically loaded into the active configuration each time the device is booted.

Use the following commands to load the public key files.

To load a public key file called pkeys.txt from a management IV module's PCMCIA flash card:

Syntax: ip ssh pub-key-file slot1 | slot2 <filename>

EXAMPLE:

```
dmz_router1(config)# ip ssh pub-key-file slot1 pkeys.txt
```

To load a public key file called pkeys.txt from a specified TFTP server:

Syntax: ip ssh pub-key-file tftp <tftp-server-ip-addr> <filename>

EXAMPLE: dmz_router1(config)# ip ssh pub-key-file tftp 192.168.1.234
pkeys.txt

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Other Useful SSH Commands

Display the current public keys
 Remove the active public keys
 Reload the public keys

Syntax: show ip client-pub-key
Syntax: clear public-key
Syntax: ip ssh pub-key-file reload

If you have a small SSH implementation with less than 10 SSH client public keys in your RSA public key file, you can store the keys in the startup-config file with the following command:

Syntax: ip ssh pub-key-file flash-memory

EXAMPLE:

```
dmz_router1(config)# ip ssh pub-key-file flash-memory
dmz_router1(config)# write memory
```

The keys can be removed from the startup-config with the "clear public-key" command.

Enabling RSA Challenge-Response Authentication

Once the public keys have been imported and configured to load with each restart of the Foundry device, RSA authentication can be enabled on the device. By default, Foundry devices will automatically enable RSA authentication when RSA public keys are loaded. You can enable RSA Authentication with the following command:

Syntax: ip ssh rsa-authentication yes | no

To enable RSA challenge-response authentication:

```
dmz_router1(config)# ip ssh rsa-authentication yes
```

To disable RSA challenge-response authentication:

```
dmz_router1(config)# ip ssh rsa-authentication no
```

Step 4: Setting Optional Parameters

Foundry's implementation of SSH allows you to configure a wide range of optional parameters.

- The number of SSH authentication retries (default value: 3 retries)
- The server RSA key size (default value: 768 bits)
- The user authentication method the Foundry device uses for SSH connections
- Whether the Foundry device allows users to log in without supplying a password (default value: NO)
- The port number for SSH connections (default value: port 22)
- The SSH login timeout value (default value: 120 seconds)
- A specific interface to be used as the source for all SSH traffic from the device (default value: IP address with lowest value)
- The maximum idle time for SSH sessions (default value: no idle timeout set)

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



NOTE: For a complete reference and syntax of all supported SSH optional parameters, please refer to your device's release notes, command reference, and Foundry's Security Guide.

The most important SSH option to setup is the "User Authentication Method" that a Foundry device uses for its SSH connections. If you completed Step 3 and imported RSA Client public keys, you will want to use the RSA Authentication method and disable Password Authentication. If you will be using Password Authentication and not RSA Authentication, disable the RSA Authentication method and enable Password Authentication.

Syntax: ip ssh rsa-authentication no | yes

Syntax: ip ssh password-authentication no | yes

EXAMPLE:

For our example, say that your company has decided to use the existing local usernames and passwords to authenticate to the Foundry device. You will need to disable the RSA challenge-response authentication and enable password authentication.

```
dmz_router1(config)# ip ssh rsa-authentication no
dmz_router1(config)# ip ssh password-authentication yes

dmz_router1(config)# aaa authentication login default local
dmz_router1(config)# write memory
```

The "aaa authentication login default local" forces all logins to the local username and password database.

Timing Out SSH Sessions

If your security policy requires the automatic timeout of unused sessions, use the SSH timeout command, to specify the number of minutes of inactivity to time the SSH session out. This is recommended for all SSH installations to prevent remote sessions from being left unattended for long durations.

Syntax: ip ssh idle-time <minutes>

Helpful SSH Commands

Once SSH is fully configured and running on the Foundry device, the following commands will help you manage the SSH connections.

To view active SSH connections

Syntax: show ip ssh

To display who is using SSH connections

Syntax: show who

To terminate an active SSH connection

Syntax: kill ssh <connection-id>

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Securing SNMP

The Simple Network Management Protocol (SNMP) is a very useful protocol that allows you to manage Foundry devices remotely. SNMP has two modes of operation, read-only and read-write, and allows you to collect information about the network and configure devices remotely. Unfortunately, earlier implementations of SNMP are not very secure – using only plain text community strings to control read-only and read-write accesses to the network devices. What makes it worse, is that most devices use well-known community strings such as public (for read-only) and private (for read-write) as default values.

Because of the weak text based authentication, SNMP has become a primary tool used by intruders to collect information about a network – in effect mapping out the entire network. The types of information that can be retrieved from a network device using SNMP includes:

- MAC address bindings
- IP address bindings
- Type of hardware and version of operating system
- Router interface information
- Router tables
- ARP tables
- Device up time

With this information, a hacker can easily search published vulnerability or bug databases and find weaknesses to exploit or map out the enterprise network to find other vulnerable hosts to attack.

SNMP Versions

There are three SNMP versions that are currently in use.

SNMP Version 1	SNMP v1 is the most widely used - due to the time it has been in production since the early 1990's. SNMP v1 has the most weaknesses with the most obvious being simple text community strings used for authentication purposes.
SNMP Version 2c	SNMP v2 was short lived due to the IETF's failure to agree on the security and administrative features of the new protocol. The most popular version of SNMP v2 is SNMP v2c which contained some enhancements to the original SNMP v1 protocol but left out the desperately needed security features. SNMP v2c still used simple text community strings for authentication.
SNMP Version 3	SNMP v3 builds on 2c by adding several new security features. It added MD5 or SHA hashes for authentication, encryption of the packet, and a mechanism to guarantee message integrity. Some of the older network management tools may not support SNMP v3 so you must check the compatibility and interoperability before using SNMP v3 on your Foundry devices.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Security should be applied to both ends of the SNMP management session. The rest of this section gives you the necessary commands to setup and secure SNMP on Foundry devices, but you should take special care to secure the management station that will be accessing the Foundry devices with SNMP. Since all of the community strings, usernames, passwords, and encryption keys are stored on the management system, securing this station is critical to your network infrastructure security. All the ACLs, VLAN restrictions, and IP address limitations setup on the Foundry devices will not stop an intruder on the management station - since this management station is allowed to configure the Foundry devices.

Configuring SNMP Version 1 and 2

Due to SNMP v1's popularity, it is supported by most if not all network management platforms. For low-risk networks, SNMP v1 can still be used effectively as long as it is secured properly. For networks that are in the medium and high-risk categories, SNMP v3 is highly recommended.

NOTE: Application – Low Security Models. Use SNMP v3 if possible.

If SNMP v1 or v2 must be used, take the following measures to decrease your exposure:

- Use strong community strings that are hard to guess
- Change the default read-only password from "public" to a secure community string
- Use different community strings for read-only and read-write functionality
- If possible, use different community strings for different devices or group of devices
- Enable SNMP read-write functionality only if absolutely necessary
- Limit SNMP access to specific IP Addresses or VLANs using ACLs or built-in commands
- Block outbound SNMP packets at the border router to prevent SNMP leakage
- Block inbound SNMP packets at the border router to prevent SNMP get requests

Foundry devices come with the SNMP read-only community set to "public" by default. With software releases after version 05.0.00, there is no default read-write community set. This prevents management applications from opening a read-write SNMP session to the device – you must enable the read-write option by setting a community string. There is no maximum number of SNMP strings that you can setup for each mode of operation. The limit of SNMP strings allowed is governed by the amount of memory on the Foundry device.

To enable SNMP v1 or v2 and change the default community strings for the read-only and read-write modes of operation, use the following command:

Syntax: [no] snmp-server community 0|1 <string> ro|rw

By default, Foundry devices encrypt the SNMP string to prevent them from being viewed in the CLI with the "show config" command. The "0" value disables the encryption of the SNMP string in the running configuration and the startup configuration and the "1" value enables encryption.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

This example turns on SNMP management for both read-only and read-write access and sets the community strings for each SNMP access method. The read-only community string is set to "my-read-only-string" and the read-write community string is set to "my-read-write-string". The default encryption setting is "1" to encrypt the SNMP strings.

```
BigIron(config)# snmp-server community my-read-only-string ro
BigIron(config)# snmp-server community my-read-write-string rw
BigIron(config)# write memory
```

NOTE: If you delete the startup-config file, the device automatically re-adds the default "public" read-only community string the next time you load the software.

Securing SNMP Access

SNMP access can be limited in one of three ways on Foundry devices. This limits the ability to use SNMP to specific management stations and decreases the threat of unauthorized access by SNMP management applications. You can use the following methods to limit SNMP access.

- | | |
|---------------------|---|
| • ACLs lists | Syntax: snmp-server community <string> ro rw <acl num> |
| • Built-in commands | Syntax: [no] snmp-client <ip-addr> |
| • VLANs | Syntax: [no] snmp-server enable vlan <vlan-id> |

NOTE: For a complete description of how to limit SNMP to specific IP addresses or VLANs using ACLs and built-in commands see the previous section titled, "Restricting Management Protocols".

Securing SNMP To A Physical Interface

In some high security environments where SNMP must be used to manage devices, Foundry allows you to configure the device to only accept SNMP access from a specific port. This can be useful for network devices used in areas such as the DMZ or Screened Subnets. Using this feature in conjunction with SNMP restrictions by IP address can increase the security of your devices in a Layer 2 environment.

Syntax: [no] snmp-server enable ethernet <portnum>

EXAMPLE:

This example configures the Foundry device to only accept SNMP from Ethernet port E7/11.

```
BigIron(config)# snmp-server enable ethernet 7/11
```

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Blocking SNMP Access From The Internet

Since SNMP is not a very secure remote management protocol, it is best to block its use whenever possible. You should never allow SNMP traffic to enter or leave your corporate network and enter the Public Internet. If you must perform remote management of your network devices across the Internet, there are much better ways of doing this. Using VPN connections, strong authentication RAS servers, or using SSH to access a secure terminal server are alternate secure methods of managing the devices remotely.

On the router that connects your DMZ to the Public Internet, implement the necessary inbound ACLs to block SNMP traffic to and from the Internet. To block SNMP requests from the Internet, apply the ingress ACLs to the outside port facing the Internet. To prevent SNMP packets from leaving your corporate DMZ, apply the ingress ACLs to the Ethernet port facing your DMZ.

NOTE: Inbound ACLs are more efficient than outbound ACLs. Whenever possible, try to design your ACLs so they can be applied in the inbound direction.

EXAMPLE:

This example creates an ACL numbered "10" which blocks all SNMP traffic (UDP 161 SNMP, UDP 162 SNMP traps) from any source to any destination address. The ACL is applied on the inside and outside ports in the inbound direction. Interface E16 is the outside port facing the public Internet and E1 is the Ethernet port facing the DMZ.

```
edgerouter(config)# access-list 101 deny udp any any eq 161
edgerouter(config)# access-list 101 deny udp any any eq 162
edgerouter(config)# access-list 101 permit ip any any

edgerouter(config)# interface ethernet 16
edgerouter(config-if-16)# ip access-group 101 in

edgerouter(config)# interface ethernet 1
edgerouter(config-if-1)# ip access-group 101 in
```

Configuring SNMP Views

An optional feature of SNMP on Foundry devices is the ability to specify SNMP views. SNMP views can be used in SNMP v2 and v3 installations to control the MIB objects that can be accessed by a particular user account. This adds a layer of security around SNMP by removing some of the MIB objects from user's with lower privileges. SNMP views reference MIB objects using object names, numbers, wildcards, or a combination of the three. The numbers represent the hierarchical location of the object in the MIB tree.

NOTE: Application – Medium to High Security Models that have many levels of administrators and where the security policy requires the limitation of SNMP access to the devices.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



To setup SNMP views, perform the following steps. The **snmp-server view** command supports the MIB objects as defined in RFC 1445.

Step 1: Configure the maximum number of SNMP views on the Foundry device. The default number of views is 10.

Syntax: system-max view <number-of-views>

EXAMPLE:

This example sets the maximum number of SNMP v2 or v3 views to 25

```
BigIron(config)# system-max view 25
```

Step 2: Add an SNMP view and define the MIB group or objects to be included or excluded.

Syntax: [no] snmp-server view <name> <mib_tree> included | excluded

The <name> parameter can be any alphanumeric name you choose to identify the view. The names cannot contain spaces.

The <mib_tree> parameter is the name of the MIB object or family. MIB objects and MIB sub-trees can be identified by a name or by the numbers called Object Identifiers (OIDs) that represent the position of the object or sub-tree in the MIB hierarchy. You can use a wildcard (*) in the numbers to specify a sub-tree family.

The **included** | **excluded** parameter specifies whether the MIB objects identified by the <mib_family> parameter are included in the view or excluded from the view.

NOTE: All MIB objects are automatically excluded from any view unless they are explicitly included; therefore, when creating views using the **snmp-server view** command, indicate which portion of the MIB you want users to access.

EXAMPLE:

This example creates a view called "admin" and allows all Foundry MIB objects that begin with the 1.3.6.1.4.1991 object identifier. All others are excluded by default.

```
BigIron(config)# snmp-server view admin 1.3.6.1.4.1.1991 included
```

NOTE: For a complete description of how to create SNMP views, refer to the *Foundry Security Guide* and/or *Foundry Switch and Router Command Line Interface Reference*.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Configuring SNMP v3 On Foundry Devices

SNMP v3 is much more secure than versions 1 and 2c and is highly recommended for installations that require authenticated and encrypted SNMP management access to their Foundry devices. If you are using SNMP to manage devices on networks outside of your firewall, such as the DMZ or screened subnets, SNMP v3 should be implemented to provide secure access and the SNMP management clients should be locked down by IP Address and by Ethernet Port.

The disadvantage of SNMP v3 is that not all network devices or management stations support SNMP v3. You must check with your management application before deploying SNMP.

SNMP v3 supports three methods of access:

- No authentication and no encryption (SNMP security keyword NoAuthNoPriv)
- Authentication and no encryption (SNMP security keyword AuthNoPriv)
- Authentication and encryption (SNMP security keyword AuthPriv)

By implementing both authentication and encryption, the device is protected against:

- Modification of SNMP information
- Identity spoofing
- Sniffing of SNMP traffic

SNMP v3 uses a user-based security model where usernames are used to authenticate access. In addition to usernames, SNMP views can also be used to limit the MIB objects that a user can access – increasing security measures for administrators that should have less access rights.

Step 1: Configuring the Network Manager System (NMS)

The first step is to make sure your SNMP management station supports SNMP v3 and configure the NMS agent with the necessary users that will be used to authenticate against the Foundry device. Refer to your NMS manual for the specific instructions on adding user accounts.

Step 2: Defining the Engine ID

Each Foundry device requires a unique Engine ID. By default, a unique Engine ID is generated on each Foundry device. You can use the "**show snmp engineid**" command to view the default Engine ID on the device. If you would like to change the default ID to another unique Engine ID, use the following command:

Syntax: [no] snmp-server engineid local <hex-string>

The <hex-string> variable consists of 11 octets, entered as hexadecimal values. There are two hexadecimal characters in each octet. There should be an even number of hexadecimal characters in an engine ID.

The default engine ID has a maximum of 11 octets:

- Octets 1 through 4 represent the agent's SNMP management private enterprise number as assigned by the Internet Assigned Numbers Authority (IANA). The most significant bit of Octet

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



1 is "1". For example, "000007c7" is the ID for Foundry Networks in hexadecimal. With Octet 1 always equal to "1", the first four octets in the default engine ID is always "800007c7" (which is 1991 in decimal).

- Octet 5 is always 03 in hexadecimal and indicates that the next set of values represent a MAC address.
- Octets 6 through 11 form the MAC address of the lowest port in the management module.

EXAMPLE:

This example changes the default Engine ID on the device to the "800007c70300e05290ab60" hex value.

```
BigIron(config)# snmp-server engineid local 800007c70300e05290ab60
```

Step 3: Defining an SNMP Group

SNMP groups are used to map SNMP users to SNMP views and can be used to configure a read view, write view, or both. Even if views are not used, the groups must be mapped to the default views in SNMP v3.

Syntax: [no] snmp-server group <groupname> v1 | v2 | v3 auth | noauth
[access <standard-acl-id>] [read <viewstring> | write <viewstring>]

The **group** <groupname> parameter defines the name of the SNMP group to be created.

The **v1**, **v2**, or **v3** parameter indicates which version of SNMP is used. In most cases, you will be using v3, since groups are automatically created in SNMP versions 1 and 2 from community strings.

The **auth** | **noauth** parameter determines whether or not authentication will be required to access the supported views. If **auth** is selected, then only authenticated packets are allowed to access the view specified for the user group. Selecting **noauth** means that no authentication is required to access the specified view.

The **access** <standard-acl-id> parameter is optional. It allows incoming SNMP packets to be filtered based on the standard ACL attached to the group.

The **read** <viewstring> | **write** <viewstring> parameter is optional. It indicates that users who belong to this group have either read or write access to the MIB.

The <viewstring> variable is the name of the view to which the SNMP group members have access. If no view is specified, then the group has no access to the MIB.

The value of <viewstring> is defined using the **snmp-server view** command. The SNMP agent comes with the "v1default" view, the default view that provides access to the entire MIB; however, it must be specified when creating the group. The "v1default" view also allows SNMP version 3 to be backwards compatible with SNMP version 1 and version 2.

NOTE: If you will be using a view other than the "v1default" view, that view must be configured before creating the user group.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

This example creates an SNMP group called "admin" and specifies the SNMP version as v3. The "admin" group requires authorization and the SNMP view that is associated for read access is called "v1default" and the SNMP view that is associated with write access is called "v1default". The "v1default" views provide access to the entire MIB. No access ACLs is associated with this group.

```
BigIron(config)# snmp-server group admin v3 auth read v1default write v1default
```

Step 4: Defining an SNMP User Account

The last step in implementing SNMP v3 is to create the User Accounts that will be used for SNMP authentication. The SNMP Group created in the previous step must be associated with the user and the authentication type must be specified.

Syntax: [no] snmp-server user <name> <groupname> v3 [[access <standard-acl-id>] [encrypted] [auth md5 <md5-password> | sha <sha-password>] [priv [encrypted] des <des-password>]]

The <name> parameter defines the SNMP user name or security name used to access the management module.

The <groupname> parameter identifies the SNMP group to which this user is associated or mapped. All users must be mapped to an SNMP group. Groups are defined using the **snmp-server group** command.

NOTE: The SNMP group to which the user account will be mapped should be configured before creating the user accounts; otherwise, the group will be created without any views. Also, ACL groups must be configured before configuring user accounts.

The **v3** parameter is required.

The **access** <standard-acl-id> parameter is optional. It indicates that incoming SNMP packets are filtered based on the ACL attached to the user account.

NOTE: The ACL specified in a user account overrides the ACL assigned to the group to which the user is mapped. If no ACL is entered for the user account, then the ACL configured for the group will be used to filter packets.

The **encrypted** parameter means that the MD5 or SHA password will be a digest value. MD5 has 16 octets in the digest. SHA has 20. The digest string has to be entered as a hexadecimal string. In this case, the agent need not generate any explicit digest. If the **encrypted** parameter is not used, the user is expected to enter the authentication password string for MD5 or SHA. The agent will convert the password string to a digest, as described in RFC 2574.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



The **auth md5 | sha parameter** is optional. It defines the type of encryption that the user must have to be authenticated. Choose between MD5 or SHA encryption. MD5 and SHA are two authentication protocols used in SNMP version 3.

The **<md5-password>** and **<sha-password>** define the password the user must use to be authenticated. These password must have a minimum of 8 characters. If the encrypted parameter is used, then the digest has 16 octets for MD5 or 20 octets for SHA.

NOTE: Once a password string is entered, the generated configuration displays the digest (for security reasons), not the actual password.

The **priv [encrypted] des <des-password>** parameter is optional. It defines the type of encryption that will be used to encrypt the privacy password. If the "encryption" keyword is used, enter a 16-octet DES key in hexadecimal format for the des-password. If the "encryption" keyword is not used enter a password string. The agent will generate a suitable 16-octet DES key from the password string.

Currently, DES is the only encryption type supported for priv password.

EXAMPLE:

This example creates a user named "bob" and associates him with the "admin" group created in the previous step. SNMP v3 is specified and a user ACL numbered "2" will be used to filter SNMP access for user "bob". Authentication using MD5 will be used with the password of "bobmd5" and the MD5 password will be encrypted with DES encryption using the "bobdes" password.

```
BigIron(config)# snmp-server user bob admin v3 access 2 auth md5 bobmd5
priv des bobdes
```

Other Useful SNMP v3 Commands

To display the Engine ID
 To display the SNMP Groups
 To display SNMP User Information

Syntax: show snmp engineid
Syntax: show snmp group
Syntax: show snmp user

NOTE: For a complete description of how to implement SNMP v3, refer to the *Foundry Security Guide* and *for Foundry Switch and Router Command Line Interface Reference*.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Removing Unnecessary Components

A big part of hardening routers or switches is to make sure there are no unprotected services that can offer an intruder a way to break in. A careful analysis of your network design and management access methods should reveal what is required to manage your infrastructure and all other unnecessary components, services, and protocols should be removed. Your Corporate Security Policy should also be consulted to see if which services are permitted on the device.

For example, your security policy may dictate that no infrastructure equipment in the DMZ may be managed with HTTP Web access. In this case, you must disable the Web management component of the Foundry devices in this area of the network.

By default, Foundry devices have the following services turned on:

- Telnet
- Web Management Interface
- SNMP (Read-Only)
- Source Routing
- Layer 2 Bridging
- ICMP

Disabling Telnet And Suppressing Telnet Rejection Messages

If your company's security policy is not to permit insecure remote access methods such as Telnet, it is highly recommended that you disable the Telnet service on the Foundry devices and configure Telnet server rejection messages to prevent the "Telnet server disabled" message from being displayed on the remote Telnet client's console. By default, Foundry devices have the Telnet service enabled and the Telnet rejection message disabled.

In the default mode, if a Foundry device denies Telnet management access to the device, the software sends a message to the denied Telnet client. You can optionally suppress the rejection message. When you enable the option, a denied Telnet client does not receive a message from the Foundry device. Instead, the denied client simply does not gain access.

NOTE: Application – Low, Medium, and High Security Models. Any installation where Telnet is not required.

NOTE: Hackers often use Telnet as a tool to discover more information about the device they are surveying. By turning off the Telnet rejection message, the Telnet session gains no further information.

To completely disable the Telnet server service and configure the Foundry device to suppress Telnet rejection messages, perform the following commands. Also remove any "enable telnet password" commands from the configuration.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Syntax: [no] telnet server

Syntax: [no] telnet server suppress-reject-message

EXAMPLE:

This example removes the previous Telnet enable password that was set for the device, removes the Telnet service, and implements the Telnet suppression reject message.

```
BigIron(config)# no enable telnet password telnet-password_text
BigIron(config)# no telnet server
BigIron(config)# telnet server suppress-reject-message
```

Disabling Web Management

Web management provides a user-friendly browser interface that some administrators prefer over the text based CLI interface. The disadvantage of using the HTTP protocol to manage devices that require a higher level of security is the ability to snoop the packets. HTTP passes the authentication information and the data payload in clear text. With the use of AAA protocols such as RADIUS and locking Web management clients to specific IP addresses and VLANs, you can add extra layers of security. If your corporate security policy prohibits the use of HTTP for managing secure network infrastructure components, you will need to disable the Web management service from your Foundry devices.

By default, Foundry device have the Web management service enabled.

NOTE: Application – Low, Medium, and High Security Models. Any installation where Web management is not required.

Syntax: [no] web-management

EXAMPLE:

This example turns off the Web management service on the Foundry device.

```
BigIron(config)# no web-management
BigIron(config)# write memory
```

Disabling SNMP

As useful as SNMP is to have for managing your network device, it can be lethal in the hands of a skillful hacker. SNMP can be used by intruders to gain valuable information on how your network devices are setup - even allowing them to reconfigure your routers and switches. As stated in the SNMP section above, SNMP can be secured in several ways by limiting access to the SNMP client and blocking SNMP access from the public Internet. For high security environments, this may not be enough and your corporate security policy may dictate that SNMP services be removed. An example of where this may be appropriate may be in shared co-location environments or in high risk DMZ's.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



To disable SNMP on your Foundry device, you should first remove any community strings used for read-only and read-write modes and then disable the SNMP service on the Foundry device. Disabling SNMP on your device will make them much more secure, but any management applications (IronView) that need SNMP to communicate with the device will no longer function.

NOTE: In high security environments where SNMP is not used for managing the Foundry devices, you should disable SNMP entirely for added security. This ensures that no SNMP client can make changes to the configuration or attempt to get information from the device.

Syntax: [no] snmp-server community <string> ro|rw

Syntax: [no] snmp disable

EXAMPLE:

This example removes the SNMP read-only and read-write community strings and disables the SNMP service.

```
BigIron(config)# no snmp-server community my-read-only-string ro
BigIron(config)# no snmp-server community my-read-write-string rw
BigIron(config)# snmp disable
BigIron(config)# write memory
```

Disabling Source Routing

As packets move through the network, routers are used to examine the destination IP address of each packet to determine the next hop to forward the packet to. With Source Routing, the sender of the packet can bypass the path that the router would normally take by configuring the packet with the next hop values. There are two forms of Source Routing – Strict Source Routing where the *exact route* is specified and Loose Source Record Route (LSRR) where the sender gives one or more hops that the packet must take.

Source Routing is normally used with tools such as traceroute for troubleshooting communications between hosts and for checking the performance of a particular path. But hackers are using Source Routing to bypass corporate security measures such as firewalls, intrusion detection systems, and even systems behind NAT'd subnets. If your company doesn't have applications that use source routing technology, you should disable Source Routing on every Foundry router.

NOTE: Application – Low, Medium, and High Security Models. Any installation where source routing applications are not required.

Syntax: [no] ip source-route

EXAMPLE:

This example turns off Source Routing on the Foundry router.

```
BigIron(config)# no ip source-route
```

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Disabling L2 Bridging

By default, Foundry Layer 3 devices with the Layer 2/3 switch routing code will bridge all protocols that are not specifically routed. Although this increases packet delivery services, it may pose security risks in certain areas of the network where high security routing is desired. Because turning off Layer 2 switching can cause certain applications to stop working, you should fully test this feature in a non-production environment before implementing it.

Layer 2 bridging can be turned off globally for the entire router or just for one interface. To disable bridging, use the "route-only" command.

NOTE: Application – Low, Medium, and High Security Models. Any installation where default Layer 2 bridging is not required.

Syntax: [no] route-only

EXAMPLE:

This example turns off Layer 2 bridging on the entire device and reboots the router.

```
BigIron(config)# route-only
BigIron(config)# write memory
BigIron(config)# exit
BigIron# reload
```

EXAMPLE:

This example turns off Layer 2 bridging on interface E3/2.

```
BigIron(config)# interface ethernet 3/2
BigIron(config-if-3/2)# route-only
BigIron(config-if-3/2)# write memory
```

Disabling ICMP Services

ICMP is a very common protocol that many of us are familiar with. ICMP is used by many applications and it is often the most popular troubleshooting tool of choice among IT professionals – ping and traceroute. Hackers also realize this and have used features of ICMP to collect information and fingerprint routers and network topologies as a prelude to an attack. For these reasons, it is important to review your company's ICMP requirements and match them up with your corporate security policy.

Although some companies have started to deny all ICMP services through their border routers, some ICMP services are still required for efficient exchanging of IP packets. By turning off some of the unnecessary ICMP functions, you can increase the level of security at your perimeter or internal LAN by thwarting some of the common ICMP data gathering techniques.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



The following are ICMP concerns and issues:

- ICMP MTU Discovery
- ICMP Redirects
- ICMP Directed Broadcasts
- ICMP Unreachables
- ICMP Timestamp and Information Requests
- Stopping Foundry Devices From Responding to Broadcast ICMP Requests

ICMP MTU Discovery

The MTU discovery feature of ICMP is required to allow hosts to successfully negotiate the optimum Maximum Transfer Unit size of the packets. By disabling the ICMP MTU Discovery type, you can impede the performance of data transfers and induce fragments on your network. ICMP MTU Discovery is ICMP Type 3 Code 4 and it must be allowed to pass both inbound and outbound to function properly.

For maximum ICMP protection, you can enable MTU Discovery and disable all other ICMP packets for your network segments that require this type of protection. Many companies use this strategy for their DMZ - where the corporate network attaches to the public Internet. This will also prevent any ping and traceroute functionality between the public Internet and your DMZ.

NOTE: Application – Medium, and High Security Models. At a minimum, this should be considered for DMZ's, screened subnets, high security subnets.

Create the following ingress ACL on your border router and apply it to all outside interfaces facing the public Internet.

EXAMPLE:

This example allows ICMP MTU Discovery (ICMP Type 3, Code 4) to pass and denies all other ICMP types. The final statement in the ACL allows all other IP traffic to pass and interface Ethernet 16 is the outside interface.

```
edgerouter(config)# access-list 101 permit icmp any any 3
edgerouter(config)# access-list 101 deny icmp any any
edgerouter(config)# access-list 101 permit ip any any

edgerouter(config)# interface ethernet 16
edgerouter(config-if-16)# ip access-group 101 in
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

This example allows ICMP MTU Discovery, ping, and traceroute to pass. It also denies all other ICMP types. The final statement in the ACL allows all other IP traffic to pass and interface Ethernet 16 is the outside interface.

```
edgerouter(config)# access-list 101 permit icmp any any 3 4
edgerouter(config)# access-list 101 permit icmp any any 0
edgerouter(config)# access-list 101 permit icmp any any 11
edgerouter(config)# access-list 101 deny icmp any any
edgerouter(config)# access-list 101 permit ip any any

edgerouter(config)# interface ethernet 16
edgerouter(config-if-16)# ip access-group 101 in
```

ICMP Redirects

ICMP Redirects are used to instruct the router how to redirect traffic through the network. This command can be used by hackers to steer traffic to a specific router for the purpose of hijacking sessions or monitoring and record traffic flows. Networks that use routing protocols such as RIP or OSPF will have valid route tables and should not need ICMP Redirects. In most cases, this function can be turned off without any impact to network functionality. This will help prevent hackers from manipulating your network traffic using ICMP Redirects.

NOTE: At a minimum, it is highly recommended that ICMP Redirects between your network and any external networks be disabled. Optimally, it is best to disable ICMP Redirects to your entire corporate network since compromised hosts can be used to send ICMP Redirects to your router interfaces.

ICMP Redirects can be sent from your router's interfaces or can be received from other sources. To block outbound ICMP Redirects generated by your Foundry routers and the associated ICMP Redirect messages, use the following commands:

Syntax: [no] ip redirect (turns off ICMP Redirect feature)
Syntax: [no] ip icmp redirects (turns off ICMP Redirect messages)

EXAMPLE:

This example disables the ICMP Redirect capabilities from the router's E3/11 interface and prevents the "ip icmp redirect" message from being sent.

```
BigIron(config)# int e 3/11
BigIron(config-if-e100-3/11)# no ip redirect
BigIron(config-if-e100-3/11)# no ip icmp redirects
BigIron(config-if-e100-3/11)# exit
BigIron(config)# write memory
```


WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



To block inbound ICMP Redirect packets (ICMP Type 5) from another source, ACLs are used. This example blocks ICMP Redirects only and allows all other IP traffic to pass. It is applied as an inbound ACL on the outside interface facing the public Internet.

```
edgerouter(config)# access-list 101 deny icmp any any redirect
edgerouter(config)# access-list 101 permit ip any any

edgerouter(config)# interface ethernet 16
edgerouter(config-if-16)# ip access-group 101 in
```

ICMP Directed Broadcasts

ICMP Directed Broadcasts were made famous by the Smurf attack where an ICMP echo request (ping) was sent to an entire network with a spoofed source IP address. This attack caused all the hosts that heard the ping request to respond back to the spoofed IP address, overwhelming it with ICMP echo reply packets.

On Foundry routers, IP Directed Broadcasts are disabled by default. To ensure that this function is disabled, you can use "no ip directed-broadcast" command at the global configuration level.

NOTE: Application – Low, Medium, and High Security Models.

Syntax: [no] ip directed-broadcast

EXAMPLE:

This example ensures that ICMP Directed Broadcasts cannot be used against the subnets configured on this Foundry router.

```
BigIron(config)# no ip directed-broadcast
BigIron(config)# write memory
```

ICMP Unreachable

ICMP Unreachable messages are used to inform the sender that a destination device is not available or a specific service is unavailable. Routers send this message to quickly inform the sender of a "no service" condition so they can move onto other functions. Without this message, the sender must wait and timeout according to their local timeout settings. This can take several seconds to several minutes depending on how the local timeout settings are setup.

Many modern scanning tools used by hackers to probe your networks also use ICMP Unreachable messages to get feedback from a victim device. When a probe is made to a device using a wide range of scanned ports, ICMP Unreachable messages are used to tell the scanning host that the port being scanned is unavailable and the scanner moves onto the next port to try. By turning off ICMP Unreachable messages from being sent by the router, you can slow these types of scans.

For legitimate users, turning this feature off may cause long pauses to occur while their requests are waiting to timeout. For thwarting hackers, this defense is a good approach as casual hackers often lack patience.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



ICMP Unreachable messages can be controlled on each router interface to allow you to control where to implement this feature. Foundry also allows you to block all ICMP Unreachable messages or to select specific messages to drop.

NOTE: Application – Medium, and High Security Models. You must evaluate the benefits of this security application versus the possible downsides of having to make legitimate users wait long periods for timeouts to occur. At a minimum, consider this feature for DMZ's, screened subnets, and high security subnets.

Syntax: [no] ip icmp unreachable [network | host | protocol | administration | fragmentation-needed | port | source-route-fail]

If you enter the command without specifying a message type, all types of ICMP Unreachable messages are disabled. If you want to disable only specific types of ICMP Unreachable messages, you can specify the message type. To disable more than one type of ICMP message, enter the **no ip icmp unreachable** command for each messages type.

- The **network** parameter disables ICMP Network Unreachable messages.
- The **host** parameter disables ICMP Host Unreachable messages.
- The **protocol** parameter disables ICMP Protocol Unreachable messages.
- The **administration** parameter disables ICMP Unreachable (caused by Administration action) messages.
- The **fragmentation-needed** parameter disables ICMP Fragmentation-Needed But Don't-Fragment Bit Set messages.
- The **port** parameter disables ICMP Port Unreachable messages.
- The **source-route-fail** parameter disables ICMP Unreachable (caused by Source-Route-Failure) messages.

EXAMPLE:

This example disables ALL ICMP Unreachable messages for the router's E3/2 interface.

```
BigIron(config)# interface e 3/2
BigIron(config-if-3/2)# no ip icmp unreachable
BigIron(config-if-3/2)# write memory
```

ICMP Timestamp and Information Requests

Two other ICMP types that may provide hackers with valuable information regarding your network devices are the ICMP Timestamp and Information Request packets. By implementing ACLs on your border routers to filter out these ICMP request types, you can help prevent additional methods of network fingerprinting. Timestamp can be used by hackers to find the time and date settings of a network device and allow them to craft packets to defeat time-dependent security defenses and Information Request can be used to find out the types of routers and hosts you have in your DMZ.

The Timestamp Request operates on ICMP Type 13 and the Information Request operates on ICMP Type 15.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



NOTE: Application – Medium, and High Security Models. At a minimum, this should be considered for DMZ's, screened subnets, high security subnets.

EXAMPLE:

This example uses an inbound ACL on the outside interface (facing the public Internet) to block ICMP Timestamp and Information Requests (ICMP Type 13 and 15) and allow all other IP traffic. The ACL is applied to the router's external interface E16.

```
edgerouter(config)# access-list 101 deny icmp any any 13
edgerouter(config)# access-list 101 deny icmp any any 15
edgerouter(config)# access-list 101 permit ip any any

edgerouter(config)# interface ethernet 16
edgerouter(config-if-16)# ip access-group 101 in
```

Stopping Foundry Devices From Responding to Broadcast ICMP Requests

By default, Foundry devices will respond to Broadcast ICMP requests. You can disable this ability on a global level to stop Foundry devices from participating in ICMP broadcast requests.

NOTE: Application – Low, Medium, and High Security Models.

Syntax: [no] ip icmp echo broadcast-request

EXAMPLE:

This example turns off the Foundry device's ability to respond to broadcast ICMP echo packets (ping requests). It is entered at the global configuration level and applies to all interfaces.

```
BigIron(config)# no ip icmp echo broadcast-request
```

Proxy ARP

On modern networks that have properly configured hosts using valid "default gateway" addresses, it is not necessary to support Proxy ARP on Foundry routers. Proxy ARP is used to perform an ARP request on behalf of a host that is not configured with a default route. Some hackers use the router's ability to perform Proxy ARP to spoof packets and to gather information about your router and network. By turning Proxy ARP off, you can prevent or slow down the hacker's attempt at using tools that take advantage of Proxy ARP.

By default, Foundry routers have Proxy ARP turned off. You can make sure that this function is disabled by using the "no ip proxy-arp" command.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



NOTE: Application – Low, Medium, and High Security Models.

Syntax: [no] ip proxy-arp

EXAMPLE:

This example turns off Proxy ARP for the entire router.

```
BigIron(config)# no ip proxy-arp
BigIron(config)# write memory
```

Securing Routing Protocols

Routing protocols such as RIP, OSPF, and BGP are used by routers to exchange information, build their route tables, and to maintain their state. If an attacker can introduce false routing information into router tables they can redirect traffic to another site, stop routing services on the network, or cause Denial of Service types of attacks on your network. There are several ways to prevent or stop this from happening by using static routes or routing protocols with security features such as authentication and encryption.

NOTE: For a complete description of how to implement routing protocol security, refer to the *Foundry Switch and Router Command Line Interface Reference*.

Static Routes

Static routes are the most secure way to create routing tables. Static routes are not dependant on any routing protocols to exchange information between routers so there is nothing an attacker can use to introduce new routes into the router. The disadvantage of using static routes is the manual overhead of programming each router. This approach is very secure, but not very scalable. For high security requirements, static routes should still be considered. Border routers are prime candidates for static routes since there is usually only one router connected to each side and the number of routes to program can be kept at a minimum.

NOTE: Application – Medium, and High Security Models. At a minimum, this should be considered for DMZ's and screened subnets.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



To add a static route to a Foundry router, use the following command:

Syntax: ip route <dest-ip-addr> <dest-mask> <next-hop-ip-addr> | atm <slotnum>/<portnum>.<subif> | ethernet <portnum> | pos <portnum> | ve <num> [<metric>] [distance <num>] [lsp <name> | static-lsp <name>]

EXAMPLE:

This example adds a static route entry into the routing table directing all 192.128.3.0/24 destination traffic to the router's next hop interface of 209.157.22.1 with a metric count of 1.

```
BigIron(config)# ip route 192.128.2.0 255.255.255.0 209.157.22.1 1
```

Secure Routing Protocols

Secure routing protocols such as RIP, OSPF, and BGP support route distribution filtering, passive interface, or protocol authentication to ensure that router table information is exchanged between authorized routers. Foundry's RIP protocols can use RIP Neighbor Filters and Group-Filters to ensure that RIP routing paths are only exchanged with authorized RIP neighbors. OSPF supports passive interfaces, route filtering, and two password authentication methods: simple text passwords and MD5.

Passwords are programmed on each router interface that is part of the router exchange and only the routers with the proper passwords are allowed to exchange routing information. This prevents unauthorized routers, without the passwords, from learning or modifying your route tables. To maximize the security of this solution, you must keep your router passwords secret and not expose them to unauthorized parties.

In addition to passwords and route distribution filtering, each router port can be programmed to redistribute or not to redistribute router information. Based on the design of your network, you may choose to prevent router information redistribution on specific subnets to prevent devices from learning your route table.

With the ability of Windows & Unix hosts to implement routing protocols such as RIP and OSPF, installations with medium to high security requirements should implement secure routing protocol features to ensure that no unauthorized router exchanges are possible. Hackers will often attempt to gain route table information to map the network.

NOTE: For a complete description of how to implement routing protocol security, refer to the *Foundry Switch and Router Command Line Interface Reference*.

NOTE: Application – Medium, and High Security Models.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Securing RIP

The RIP protocol can be secured using the **filter** and **filter-group** commands. The Filter command allows you to configure the router with the IP network numbers that it will learn from its neighbors. The command allows you to specify which routes are learned and which ones are ignored. Once you have defined all of the valid IP network number filters, they can be applied to any interface using the RIP protocol with the Filter-Group command.

Another way to limit which routers can participate in RIP exchanges is to use the **neighbor** filter command. The neighbor filter command allows you to specify which RIP routers to accept RIP exchanges from and which ones to deny exchanges from. This method is an all or nothing approach that filters on the physical router's primary interface address rather than specific routes. This method is also known as "Passive Interfaces".

NOTE: For inbound routes, a filter defines what routes will be stored in the router's IP route table. For outbound routes, the filter defines which routes are allowed to be advertised through a given interface. You can also specify all routes by using the value **any** instead of specifying a specific route.

Method 1: RIP Filter and Filter-Groups

Rip Filters and Filter-Groups allow you to define which networks to add into your router's IP route table. It is more granular in approach than the **neighbor** filter command. The filters can be applied on a global level to affect all interfaces programmed with RIP or on a specific interface.

Syntax: filter <filter-num> permit | deny <source-ip-addr> | any <source-ip-mask> | any [log]

EXAMPLE:

This example configures the router to accept RIP information from the following IP networks: 192.53.4.1, 192.53.5.1, 192.53.6.1 and 192.53.7.1

```
BigIron(config-rip-router)# filter 1 permit 192.53.4.1 255.255.255.0
BigIron(config-rip-router)# filter 2 permit 192.53.5.1 255.255.255.0
BigIron(config-rip-router)# filter 3 permit 192.53.6.1 255.255.255.0
BigIron(config-rip-router)# filter 4 deny 192.53.7.1 255.255.255.0
```

Once the filters are defined, you can implement them on any interface using the RIP protocol with the **filter-group** command. The **filter-group** command can be applied on a global level for the entire system or for each RIP enabled interface.

Syntax: filter-group in | out <1-64> [<1-64>]

EXAMPLE:

This example applies the RIP filters defined in the previous example and applies them in the outbound direction system wide on this Foundry Layer 3 device.

```
BigIron(config-rip-router)# filter-group out 1 2 3 4
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

The following example illustrates how to configure a “deny” filter to block all RIP exchanges, both inbound and outbound, with this router’s E1/2 interface and logs the denied RIP exchanges in the system log.

```
BigIron(config-rip-router)# filter 1 deny any any log
BigIron(config-rip-router)# int e 1/2
BigIron(config-if-e1000-1/2)# ip rip filter-group in 1
BigIron(config-if-e1000-1/2)# ip rip filter-group out 1
```

Method 2: RIP Neighbor Filtering

Foundry routers using RIP will normally exchange RIP information with all RIP routers. To increase security, you can define which routers are valid to exchange RIP information within your corporate network. The **neighbor** filter is applied at the global level and affects all interfaces programmed with the RIP protocol. This method either allows all or none of the network information to be exchanged with the neighbor router.

NOTE: A neighbor filter uses an implicit “deny” at the end of the filter list. Once you configure the first neighbor filter, you must order your filters correctly so you do not accidentally disable RIP exchanges with authorized routers.

Syntax: [no] neighbor <filter-num> permit | deny <source-ip-address> | any

The <filter-num> is the filter number used to identify the neighbor filter.

The <source-ip-address> specifies the router to accept or deny RIP exchanges with. The **any** parameter will allow you to permit or deny all RIP exchanges with any router.

EXAMPLE:

This example will configure the Layer 3 Foundry device to deny all RIP exchanges with its neighbors. In effect, disabling the default RIP exchange feature normally set on Foundry devices.

```
BigIron(config-rip-router)# neighbor 1 deny any
```

EXAMPLE:

This example denies all RIP exchanges with routers exchanging information on IP addresses 192.16.1.170 and 192.16.10.5. It accepts RIP exchanges from all other routers.

```
BigIron(config-rip-router)# neighbor 2 deny 192.16.1.170
BigIron(config-rip-router)# neighbor 3 deny 192.16.10.5
BigIron(config-rip-router)# neighbor 4 permit any
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Securing OSPF

Securing OSPF on Foundry routers can be done in several ways by using passwords, passive interfaces, or router filtering. Passwords are one of the most common ways of authenticating OSPF routers to ensure that only authorized routers are participating in route table exchanges. Passive interfaces can protect the route table by controlling the router interface's ability to send and receive OSPF route information. Passive interfaces can be used in situations control route exchanges, limit OSPF information on links with insufficient bandwidth, or to supplement static route implementations. Route filtering allows you to specify which networks are exchanged and which ones are not.

OSPF Passwords

Foundry's OSPF supports three levels of password protection: none, text password, and MD5. By default, Foundry Layer 3 devices have the password protection set to none. Using passwords is a quick and efficient way to ensure that only authorized OSPF routers with the correct password are participating in router information exchanges.

Using Text Passwords

To enable password protection for your network using OSPF's text password feature, each router interface must be programmed with the correct password key to exchange information.

Syntax: [no] ip ospf authentication-key [0 | 1] <string>

The <string> parameter specifies the password and can be up to eight alphanumeric characters.

The optional 0 | 1 parameter affects encryption. For added security, software release 07.1.10 and later encrypts displaying of the password or authentication string. Encryption is enabled by default. The software also provides an optional parameter to disable the encryption of a password or authentication string on an individual OSPF area or OSPF interface basis. When encryption of the passwords or authentication strings is enabled, they are encrypted in the CLI regardless of the access level you are using.

EXAMPLE:

This example configures the router port E4/1 to use a text password called, "P0ssKey". Passwords can be alphanumeric and up to eight characters in length. All other routers exchanging route table information with this router's interface must also be configured with the same text password.

```
BigIron(config)# int e 4/1
BigIron(config-if-4/1)# ip ospf authentication-key P0ssKey
BigIron(config-if-4/1)# end
BigIron# write memory
```

Using MD5 Passwords

To enable MD5 password protection for your OSPF network, each router interface must be configured with the same MD5 key to exchange information.

Syntax: [no] ip ospf md5-authentication key-activation-wait-time <num> | key-id <num> [0 | 1] key <string>

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



The **key-activation-wait-time** <num> parameter can be a value from 0 - 14400 seconds. The default is 300 seconds (5 minutes).

The **key-id** <num> specifies the key identification number and can be a value from 1 - 255.

The **key** <string> parameter specifies the authentication string and can be up to 16 alphanumeric characters long.

The optional **0 | 1** parameter affects encryption. For added security, software release 07.1.10 and later encrypts display of the password or authentication string. Encryption is enabled by default. The software also provides an optional parameter to disable encryption of a password or authentication string on an individual OSPF area or OSPF interface basis. When encryption of the passwords or authentication strings is enabled, they are encrypted in the CLI regardless of the access level you are using.

EXAMPLE:

This example shows you how to change the default key activation time from the default value of 300 seconds to 45 seconds and setups up the MD5 authentication for this router's Ethernet 2/5 interface with a key id of "3" and a key value of "TOughK3y"

```
BigIron(config)# int e 2/5
BigIron(config-if-2/5)# ip ospf md5-authentication key-activation-wait-time 30
BigIron(config-if-2/5)# ip ospf md5-authentication key-id 3 key TOughK3y
BigIron(config-if-2/5)# exit
BigIron(config)# write memory
```

OSPF Passive Interfaces

Passive Interfaces are used to **block all** OSPF router exchanges on specific router interfaces for one of three reasons:

- To increase security by not accepting or transmitting router information across the interface.
- To save bandwidth on links that is already saturated with traffic.
- To supplement static routing.

All outbound and inbound route exchanges are blocked when an interface is configured with the **passive** command.

Syntax: [no] ip ospf passive

EXAMPLE:

This example configures each of the router interfaces with the appropriate OSPF information and disables the port's ability to send and receive OSPF route information exchanges on interface E1/1 and E4/3.

```
BigIron(config)# int eth 1/1
BigIron(config-if-1/1)# ip address 10.32.1.254/24
BigIron(config-if-1/1)# ip ospf area 0.0.0.1
```

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



```
BigIron(config-if-1/1)# ip ospf hello 4
BigIron(config-if-1/1)# ip ospf dead 16
BigIron(config-if-1/1)# ip ospf passive

BigIron(config-if-1/1)# int eth 4/3
BigIron(config-if-4/3)# ip address 10.96.1.254/24
BigIron(config-if-4/3)# ip ospf area 0.0.0.5
BigIron(config-if-4/3)# ip ospf hello 4
BigIron(config-if-4/3)# ip ospf dead 16
BigIron(config-if-4/3)# ip ospf passive
BigIron(config-if-4/3)# write memory
```

OSPF Route Filtering

Route filtering allows you to configure the OSPF router interface to accept or deny specific networks that are inserted into the route table. Route filtering is much more granular than the Passive Interface feature that accepts or denies all OSPF exchange information and there are two ways to perform this function: OSPF Distribution Lists and OSPF Interface Passive/Ignore.

Route Distribution Lists are applied to the entire router and affects all OSPF interfaces with the access lists defined. OSPF Interface Passive/Ignore allows you to specify which networks are allowed into the route table based on the interface it is configured on.

OSPF Distribution Lists

Distribution Lists are created and implemented with Standard or Extended ACLs that specifies the network numbers that are allowed or denied in the OSPF route exchanges - it is a two-step process. The first step is to create the access list and the second step is to implement the ACL in an OSPF distribution list. OSPF Distribution lists are more granular than Passive Interfaces by allowing you to specify which networks are added to the route table. They are applied to the router as a whole and will **affect all router interfaces** that have the OSPF protocol configured.

NOTE: If you change the ACL after you configure the OSPF distribution list, you must clear the IP route table to place the changed ACL into effect. To clear the IP route table, enter the **clear ip route** command at the Privileged EXEC level of the CLI.

Step 1: Creating The Access List

The first step is to create the access list ACL that will contain the network information to be exchanged or withheld from the OSPF route table. The format is the same as creating a standard or extended ACL.

EXAMPLE:

This example will create an access list that denies two specific networks and allows all others to be exchanged and placed into the router's OSPF route table.

```
BigIron(config)# access-list 105 deny 10.32.1.0/24
BigIron(config)# access-list 105 deny 10.32.2.0/24
BigIron(config)# access-list 105 permit any any
BigIron(config)# write memory
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Step 2: Applying The OSPF Distribution List

Once the access lists have been created, they can be applied to the OSPF router. Distribution lists are turned on at the router protocol level and affect all interfaces configured with OSPF. The networks that were denied in the access lists will not be inserted into the route table.

Syntax: [no] distribute-list <acl-name> | <acl-id> in

EXAMPLE:

This example completes the OSPF Distribution List by turning it on and specifying which access list to use.

```
BigIron(config)# router ospf
BigIron(config-ospf-router)# distribute-list 105 in
BigIron(config-ospf-router)# write memory
```

OSPF Interface Passive/Ignore

In the previous example of using OSPF Distribution Lists, the list was applied to the entire router – affecting all OSPF interfaces with the access list defined. If you need more granular control in specifying which network numbers should or should not be inserted into OSPF route tables, use the Interface Passive/Ignore command. This command is applied to a specific interface and only affects the interface's OSPF route filtering.

For routers that have multi-netted interfaces (more than one subnet defined), this feature can help control which networks are included in OSPF exchanges.

Syntax: [no] ip address <ip-addr>/<mask-bits> [ospf-ignore | ospf-passive]

The <ospf-ignore> parameter is used to disable OSPF adjacency information and disables advertisement of the interface into OSPF. The network with this parameter assigned to it is completely ignored by OSPF.

The <ospf-passive> parameter is used to disable OSPF adjacency information with its OSPF neighbors. This network is still advertised to OSPF.

EXAMPLE:

This example configures a router interface with the necessary OSPF information and sets the interface to passive to disable adjacency information exchanges.

```
BigIron(config)# int eth 1/1
BigIron(config-if-1/1)# ip address 10.32.1.254/24 ospf-passive
BigIron(config-if-1/1)# ip ospf area 0.0.0.1
BigIron(config-if-1/1)# ip ospf hello 4
BigIron(config-if-1/1)# ip ospf dead 16
BigIron(config-if-1/1)# write memory
BigIron(config-if-1/1)#
```

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

This example configures a router interface with multiple networks and configures the first network to exchange OSPF router information. The other two remaining subnets are not used inserted into the OSPF route table.

```
BigIron(config)# int eth 1/1
BigIron(config-if-1/1)# ip address 10.32.1.254/24
BigIron(config-if-1/1)# ip address 10.32.2.254/24 ospf-ignore

BigIron(config-if-1/1)# ip address 10.32.3.254/24 ospf-ignore
BigIron(config-if-1/1)# ip ospf area 0.0.0.1
BigIron(config-if-1/1)# ip ospf hello 4
BigIron(config-if-1/1)# ip ospf dead 16
BigIron(config-if-1/1)# write memory
```

Securing BGP

To enable MD5 password protection for your BGP4 network, configure the neighbor or neighbor peer group with an MD5 authentication string. Each neighbor must have the same password in order to successfully exchange information.

Syntax: [no] neighbor <ip-addr> | <peer-group-name> password [0 | 1] <string>

The <ip-addr> | <peer-group-name> parameter indicates whether you are configuring an individual neighbor or a peer group. If you specify a neighbor's IP address, you are configuring that individual neighbor. If you specify a peer group name, you are configuring a peer group.

The **password** <string> parameter specifies an MD5 authentication string for securing sessions between the Layer 3 Switch and the neighbor. You can enter a string up to 80 characters long. The string can contain any alphanumeric characters, but the first character cannot be a number. If the password contains a number, do not enter a space following the number.

The **0 | 1** parameter is the encryption option, which you can omit (the default) or which can be one of the following.

- **0** - Disables encryption for the authentication string you specify with the command. The password or string is shown as clear text in the output of commands that display neighbor or peer group configuration information.
- **1** - Assumes that the authentication string you enter is the encrypted form, and decrypts the value before using it.

NOTE: If you want the software to assume that the value you enter is the clear-text form, and to encrypt display of that form, do not enter 0 or 1. Instead, omit the encryption option and allow the software to use the default behavior.

If you specify encryption option 1, the software assumes that you are entering the encrypted form of the password or authentication string. In this case, the software decrypts the password or string you enter before using the value for authentication. If you accidentally enter option 1 followed by the clear-text version of the password or string, authentication will fail because the value used by the software will not match the value you intended to use.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

This example configures BGP4 routing on the Foundry Layer 3 device. The AS number is 2 and a peer-group named "usaeast" is defined in the second line. The third line defines the password for the "usaeast" peer-group as "usaeastpassword". Line four adds the neighbor with IP address "10.10.200.102" to the peer-group "usaeast". The fifth line sets the MD5 password for the "10.10.200.102" neighbor as "mypassword".

The 10.10.200.102 neighbor is configured both as a peer-group and as an individual neighbor to show that different passwords can be used for a peer-group and an individual neighbor. Peer-groups can be omitted in smaller installations.

```
BigIron(config-bgp-router)# local-as 2
BigIron(config-bgp-router)# neighbor usaeast peer-group
BigIron(config-bgp-router)# neighbor usaeast password usaeastpassword
BigIron(config-bgp-router)# neighbor 10.10.200.102 peer-group usaeast
BigIron(config-bgp-router)# neighbor 10.10.200.102 password mypassword
BigIron(config-bgp-router)# write memory
```

NOTE: Another option to consider when using BGP is the control of "route flapping" when BGP routes converge. Route Flapping can cause high CPU utilization and cause slow downs on your network. This occurs when the route constantly transitions between up and down state and causes BGP to issue many update messages to the network during its converge attempts. See the BGP "dampening" command for more information.

Common ACLs for Anti-Spoofing

Spoofing is a common practice that is used by hackers to hide their tracks or to point an attack to another host. To protect your site from spoofed IP addresses that are ***claiming to be from your network***, anti-spoofing ACLs can be used to filter packets with spoofed source addresses. The only spoofed IP addresses that can be filtered are packets that are crafted with your internal IP address ranges. By using anti-spoofing ACLs, you decrease the ability for crafted packets to access your routers and switches and block any spoofed packets from traversing your network.

It is important to that Foundry's JetCore products perform ACLs in hardware and are extremely fast for inbound ACLs. Older IronCore products perform ACLs in software and can affect the performance of the router under very heavy load conditions. Important points to keep in mind when using ACLs are:

- Inbound ACLs are more efficient than outbound ACLs.
- Put the most specific rules at the top of the ACL list and the most general towards the bottom.
- Once a packet satisfies an ACL rule, it is executed by that ACL and stops flowing down the ACL list.
- Foundry uses an "implied deny" for ACLs, which means the last action is always to deny.
- The **log** option uses the CPU. Only log traffic when it's absolutely necessary.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Inbound Anti-Spoofing ACLs

Inbound anti-spoofing ACLs should be used at the ingress point where your DMZ connects to the Public Internet. These ACLs block all inbound packets that have source addresses of your internal networks. This prevents hackers from tricking your firewalls and hosts into believing that the packet originated from your corporate networks when in fact, they came from the Public Internet.

As a precaution, you should also block any packets with a source IP address claiming to be from any one of the reserved or private networks, loopback address, broadcast address, multicast networks. The list includes the following:

- 127.0.0.0/8 (loopback address)
- 10.0.0.0/8 (private network)
- 172.16.0.0/12 (private network)
- 192.168.0.0/16 (private network)
- 224.0.0.0/4 (multicast network)
- 240.0.0.0/5 (multicast network)
- 255.255.255.255/32 (broadcast address)

Apply inbound anti-spoofing ACLs to all your inbound router ports that are connected to the Public Internet or partner networks.

NOTE: Application – Low, Medium, and High Security Models.

EXAMPLE:

To illustrate how to implement inbound anti-spoofing ACLs, assume that your internal corporate network is using the private 10.0.0.0/8 address space and your DMZ is using the IP range 198.30.15.0/24 and NAT is being used to convert the 10.0.0.0/8 addresses to a valid 198.30.15.0/24 address at the firewall. When packets leave your network for the Internet, the source address should be labeled with a valid 198.30.15.0 address and no packets coming from the Internet should have a source address in the 198.30.15.0/24 range. If an incoming packet has a source with this range, it is improperly configured or crafted by an attacker.

This example will also add anti-spoofing rules for the common private network address ranges, loopback address, multicast networks, and broadcast address. It also turns on logging for any packets claiming to be from the DMZ's address space. The logging is only turned on for the most critical anti-spoofing ACL. In this case, we want to know when an intruder is attempting to access the DMZ by crafting packets claiming to be from the DMZ.

```
edgerouter(config)# access-list 10 deny 198.30.15.0/24 log
edgerouter(config)# access-list 10 deny 127.0.0.0/8
edgerouter(config)# access-list 10 deny 10.0.0.0/8
edgerouter(config)# access-list 10 deny 172.16.0.0/12
edgerouter(config)# access-list 10 deny 192.168.0.0/16
edgerouter(config)# access-list 10 deny 224.0.0.0/4
edgerouter(config)# access-list 10 deny 240.0.0.0/5
edgerouter(config)# access-list 10 deny 255.255.255.255/32
edgerouter(config)# access-list 10 permit any
```

```
edgerouter(config)# interface ethernet 16
edgerouter(config-if-16)# ip access-group 10 in
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Outbound Anti-Spoofing ACLs

Outbound anti-spoofing ACLs are used to prevent your site from passing spoofed packets to the Internet. This can often help prevent your sites from being used in Zombie attacks where a hacker has taken over some of your hosts and have installed remote control attack software to launch against another victim. Often, these attack tools will spoof their source IP addresses to cover their tracks and outbound anti-spoofing ACLs will help prevent this from happening.

The best location to place outbound ACLs is on the router interfaces of your internal subnets. By doing this on all of your internal and DMZ router ports, you are guaranteeing that only valid outbound traffic is being allowed into the network backbone and out to the Internet. Design your ACLs with performance and security in mind – use inbound ACLs whenever possible and place ACLs strategically to load balance anti-spoofing ACLs to maintain performance.

NOTE: Application – Low, Medium, and High Security Models.

EXAMPLE:

Let's take the same subnets we used in our Inbound ACL example to illustrate how Outbound ACLs would work. Let's assume that our DMZ is using the 198.30.15.0/24 subnet and our internal network has four different subnets.

- DMZ - 198.30.15.0/24
- Server Farm - 10.32.0.0/16
- Development - 10.33.1.0/24
- HR & Finance - 10.33.2.0/24
- Manufacturing - 10.33.3.0/24

On the border router, implement the following inbound ACL on the internal Ethernet port that is facing the DMZ. This is the gateway port for the 198.30.15.0/24 subnet that leads to the Internet.

```
edgerouter(config)# access-list 20 permit 198.30.15.0/24
edgerouter(config)# access-list 20 deny any

edgerouter(config)# interface ethernet 1
edgerouter(config-if-1)# ip access-group 20 in
```

NOTE: Notice that we used an Inbound ACL and called this an Outbound Anti-Spoofing ACL. If you think about the action and the direction of the traffic flow, we essentially applied the equivalent of an Outbound ACL on the inside Ethernet Port E1. We did this because Inbound ACLs are generally more efficient than Outbound ACLs and are handled in hardware in JetCore products - this provides much faster performance.

On the internal router that controls the 10.0.0.0 subnets, we will apply an Inbound ACL to stop all Outbound Spoofed packets. Notice that we are using Inbound ACLs on each of the individual router ports (which are the default gateway ports for each of the subnets) to block Outbound Spoofed

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



packets that do not have the source IP address of the valid subnet. This is more efficient than setting up an Outbound ACL and applying it at the border router.

By distributing the anti-spoofing ACLs to more router ports, the anti-spoofing load is shared between more router ports and network performance is maintained with greater security. Furthermore, stopping Spoofed packets at the gateway port of each subnet prevents spoofed packets from entering the network backbone.

```
internalrtr(config)# access-list 21 permit 10.32.0.0/16
internalrtr(config)# access-list 21 deny any log

internalrtr(config)# access-list 22 permit 10.33.1.0/24
internalrtr(config)# access-list 22 deny any log

internalrtr(config)# access-list 23 permit 10.33.2.0/24
internalrtr(config)# access-list 23 deny any log

internalrtr(config)# access-list 24 permit 10.33.3.0/24
internalrtr(config)# access-list 24 deny any log

internalrtr(config)# interface ethernet 1/1
internalrtr(config-if-1/1)# ip address 10.32.0.254/16
internalrtr(config-if-1/1)# ip access-group 21 in

internalrtr(config)# interface ethernet 1/2
internalrtr(config-if-1/2)# ip address 10.33.1.254/24
internalrtr(config-if-1/2)# ip access-group 22 in

internalrtr(config)# interface ethernet 1/3
internalrtr(config-if-1/3)# ip address 10.33.2.254/24
internalrtr(config-if-1/3)# ip access-group 23 in

internalrtr(config)# interface ethernet 1/4
internalrtr(config-if-1/4)# ip address 10.33.3.254/24
internalrtr(config-if-1/4)# ip access-group 24 in
```

NOTE: Turning on logging allows you to capture instances of spoofing activity on your network. Logging requires CPU intervention and can impede router performance if the number of spoofed packets are very high – in effect, turning this feature into a DoS attack.

Other Services To Filter

There are many other ports and services that can potentially be used by hackers to gain information about your company. Some of these may be necessary services that your company needs to conduct business - but many are probably not required. The following tables list the ports that you should consider blocking at the border router. The rule here is to block all potentially dangerous protocols and services and only allow them if there is a specific business need.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



The following table lists recommendations for services that should be **blocked completely** at the router in both inbound and outbound directions unless explicitly required:

Port (Transport)	Service
1 (TCP & UDP)	tcpmux
7 (TCP & UDP)	echo
9 (TCP & UDP)	discard
11 (TCP)	systat
13 (TCP & UDP)	daytime
15 (TCP)	netstat
19 (TCP & UDP)	chargen
37 (TCP & UDP)	time
43 (TCP)	whois
67 (UDP)	bootp
69 (UDP)	tftp
93 (TCP)	supdup
111 (TCP & UDP)	sunrpc
135 (TCP & UDP)	loc-srv
137 (TCP & UDP)	netbios-ns
138 (TCP & UDP)	netbios-dgm
139 (TCP & UDP)	netbios-ssn
177 (UDP)	xdmcp
445 (TCP)	netbios (ds)
512 (TCP)	rexec
515 (TCP)	lpr
517 (UDP)	talk
518 (UDP)	ntalk
540 (TCP)	uucp
1900, 5000 (TCP & UDP)	Microsoft UPnP SSDP
2049 (UDP)	nfs
6000 – 6063 (TCP)	X Window System
6667 (TCP)	irc
12345 (TCP)	NetBus
12346 (TCP)	NetBus
31337 (TCP & UDP)	Back Orifice

The following table recommends services to block from external clients:

Port (Transport)	Service
79 (TCP)	finger
161 (TCP & UDP)	snmp
162 (TCP & UDP)	snmp trap
513 (TCP)	rlogin
513 (UDP)	who
514 (TCP)	rsh, rcp, rdist, rdump
514 (UDP)	Syslog
550 (TCP & UDP)	New who

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Preventing DoS Attacks

Denial of Service (DoS) attacks are becoming more common as the tools available to perform these types of attacks are becoming more popular and more sophisticated. Furthermore, the availability of these tools and their ease of use make them a weapon of choice for "script kiddies" or serious hackers looking to shut down your company's services.

In a DoS attack, resources are taken up by the attack and legitimate service is degraded significantly or completely shut down. For network devices, this can be a significant event as DoS attacks on routers and switches can shut down subnets or entire sections of your corporate network. Foundry device includes several defenses to combat the two most popular types of DoS attacks: Smurf attacks where ICMP is used and TCP SYN attacks which use a partial TCP 3-way handshake.

Preventing Smurf Attacks

In a Smurf attack, the attacker crafts an ICMP ping request by spoofing the source IP address of the desired victim and sending it to the broadcast address of a specific subnet. By allowing the ICMP request to hit the broadcast address, every device that can respond to the ping request will respond almost simultaneously back to the spoofed source address. The real owner of the spoofed IP address becomes the victim of the attack as hundreds or thousands of ICMP echo responses are received back from the Smurf attack.

Foundry has several ways of protecting your network from ICMP attacks. The first one is to stop your router from forwarding directed-broadcasts. A Smurf attack must have an intermediary that propagates the ICMP request to the broadcast address and that intermediary is your router. This is accomplished by using the "no ip directed-broadcast" command on the router.

NOTE: Application – Low, Medium, and High Security Models.

Syntax: [no] ip directed-broadcast

```
BigIron(config)# no ip directed-broadcast
```

The second is to prevent your hosts from being a victim in a Smurf attack by controlling the number of ICMP packets that it can receive. Foundry has a unique command that allows you to configure what the normal-burst and maximum-burst levels of ICMP are on the device and the lockout duration to block excessive ICMP traffic. This command can be used to protect the Foundry device as well as hosts connected to an individual interface.

Syntax: ip icmp burst-normal <value> burst-max <value> lockup <seconds>

The <value> is the number of ICMP packets/second

The lockup <seconds> specifies the duration to drop excessive ICMP packets

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

This example configures the burst-normal setting to 5000 ICMP packets/second with a maximum burst level of up to 10000 ICMP packets/second for port E3/11. Once the maximum level has been reached, all further ICMP packets will be dropped for 300 seconds.

```
BigIron(config)# int e 3/11
BigIron(config-if-e100-3/11)# ip icmp burst-normal 5000 burst-max 10000
lockup 300
```

This next example configures ICMP protection for the router. It is configured to accept a normal-burst rate of 5000 ICMP packets/second and a maximum-burst rate of 10000 ICMP packets/second. Once the maximum threshold has been reached, all excessive ICMP packets will be dropped for 60 seconds.

```
BigIron(config)# ip icmp burst-normal 5000 burst-max 10000 lockup 60
```

Preventing TCP SYN Attacks

The other kind of popular attack is to bombard a host with spoofed SYN packets using a wide range of false addresses as the source IP address. As the victim host receives hundreds to thousands of these connection requests, it will build many "connection attempt" entries in its connection queue for completing the 3-way TCP handshake. The victim host will issue a SYN-ACK packet to each spoofed host and wait for the completing ACK handshake packet to return.

If the spoofed host is a legitimate device, it will realize that it never originated the SYN request and quickly issue a RST command to tell the victim host to drop the connection. If the spoofed IP address is not a legitimate host, the victim host will hold the incomplete connection entry in its connection queue until the timeout value is reached and then flush the entry from its queue. The victim host's resources can be depleted rather quickly if it is waiting for hundreds or thousands of hosts to respond – hence, the DoS attack is successful at degrading or stopping service on the victim host.

With all Foundry devices, you can prevent excessive SYN's from bombarding a host by telling the device to drop TCP SYN packets that are over a specified maximum-burst level. This command can be used to protect the router itself or it can be applied to a specific interface to protect individual hosts.

NOTE: Application – Low, Medium, and High Security Models.

Syntax: ip tcp burst-normal <value> burst-max <value> lockup <seconds>

The <value> is the number of SYN packets/second

The lockup <seconds> specifies the duration to drop excessive SYN packets

For networks that have many SYN requests, you should analyze what your network's peak SYN load is before setting the "burst-normal" and "burst-max" values. Using a network analyzer such as a sniffer, you can measure the number of SYN requests during peak times to obtain this information. Plot your data over several days to make sure your estimates are including peak and low usage times. Once you have a good understanding of the SYN request load on your network, the command can be implemented on your Foundry devices.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

This example protects the Foundry device from a SYN attack by configuring the normal-burst value to 30 SYN packets/second and the maximum-burst value to 100 SYN packets/second. Once the maximum value has been reached, all SYN packets are dropped for 60 seconds.

```
BigIron(config)# ip tcp burst-normal 30 burst-max 100 lockup 60
```

This next example protects a host on interface E3/11 by configuring the normal-burst value to 30 SYN packets/second and the maximum-burst value to 100 SYN packets/second. Once the maximum value has been reached, all SYN packets are dropped for 60 seconds.

```
BigIron(config)# int e 3/11
BigIron(config-if-e100-3/11)# ip tcp burst-normal 30 burst-max 100 lockup 60
```

Preventing LAND Attacks

The Land Attack is a threat that uses the same IP Address in both the source and destination address fields in the IP header and the same port in the source and destination port of the protocol header. This attack is designed to cause a Denial of Service (DoS) condition on the router to decrease performance or to take the router down. Using ACLs, you can prevent this attack from occurring.

NOTE: Application – Low, Medium, and High Security Models. At a minimum, protect your border router's external interface from Land Attacks.

EXAMPLE:

This example creates an inbound ACL that blocks any IP packets destined for the border router's interfaces with the same source and destination IP Addresses. Assume that the DMZ's ethernet interface is E2/1 and its address is "198.30.15.254/24" and the router interface E2/16 is facing the ISP and its address is 196.100.102.23/28. We will need to make sure that both gateway addresses are protected against LAND attacks.

If the border router has the inbound anti-spoofing ACLs set for the DMZ network, the LAND Attack ACL can exclude the network address for this network.

```
BigIron(config)# access-list 105 deny ip host 198.30.15.254 host
198.30.15.254 log
BigIron(config)# access-list 105 deny ip host 196.100.102.23 host
196.100.102.23 log
BigIron(config)# access-list 105 permit ip any any
BigIron(config)# interface E2/1
BigIron(config-if-e100-2/1)# ip address 198.30.15.254/24
BigIron(config-if-e100-2/1)# ip access-group 105 in
BigIron(config-if-e100-2/1)# interface E2/16
BigIron(config-if-e100-2/16)# ip address 196.100.102.23/28
BigIron(config-if-e100-2/16)# ip access-group 105 in
BigIron(config-if-e100-2/16)# exit
BigIron(config)# write memory
```

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Limiting Broadcast Packets

Foundry devices also allow you to specify the maximum number of broadcast packets a port can forward per second. This may be a valuable tool to control the amount of broadcast traffic on your network and reduce the chances of a DoS attack that uses broadcast packets. Note that this feature is software based and requires all broadcast packets to be calculated using the CPU. For networks with very high levels of normal broadcast traffic, this feature may decrease performance - test this feature before implementing in production environments.

NOTE: Application – Low, Medium, and High Security Models where there are possibilities of broadcast storms that will take down critical services.

Syntax: broadcast limit <num>

Possible values: 0 - 4294967295; if you specify 0, limiting is disabled.

EXAMPLE:

This example limits the broadcast packets to 1000 packets per second on port Ethernet 1/3

```
BigIron(config-if-1/3)# broadcast limit 1000
```

Limiting ARP Requests

Foundry devices allow the limiting of ARP requests that it can receive. Because ARP packets are processed by the device's CPU, it is susceptible to DoS attacks that use ARP services. By configuring a maximum number of ARP requests that the Foundry device can receive per second, you can prevent or slow this type of attack from stopping services on your network devices.

NOTE: Application – Low, Medium, and High Security Models where there are possibilities of ARP storms that will take down critical services.

Syntax: [no] rate-limit-arp <num>

The <num> parameter specifies the number of ARP packets that can be accepted per second. The range can be from 0 - 100. If you specify 0, the device will not accept any ARP packets. If you want to change a previously configured the ARP rate limiting policy, you must remove the previously configured policy using the **no rate-limit-arp <num>** command before entering the new policy.

EXAMPLE:

This example configures the Foundry device to accept up to 50 ARP requests per second on the entire device. If more than 50 ARP packets are received during the one-second interval, they are dropped. The next one-second interval resets the counter.

```
BigIron(config)# rate-limit-arp 50
```

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Preventing Fragmentation Attacks

Fragmentation occurs naturally in TCP/IP communications when the MTU sizes do not match between the sending and receiving hosts. But hackers have used fragmentation as a means to bypass firewalls and IDS systems that do not support packet reassembly or perform it improperly.

How Fragmentation Works

Fragmentation occurs when the end-to-end MTU size is smaller than the packet being transmitted. ICMP is used to negotiate the MTU size between the hosts before data transfer begins and the sender typically uses the smallest MTU size for the path to avoid fragmentation – but this is not always possible and thus fragmentation occurs.

This example will illustrate the need for fragmentation. Sender A has a file it needs to send to host Receiver B and its normal datagram size is 4028 bytes. The negotiated maximum MTU size of all the routers in the path between host A and B is only 1500 bytes. Sender A will have to fragment the original packet into three separate packets to successfully transmit the information. Receiver B will be responsible for packet reassembly using the Fragment ID Number and Fragment Sequence Number. All fragments of the same packet must contain the following in the IP header:

- All fragments of the same original datagram must have the same Fragment ID Number.
- Each fragment must carry a Sequence Number that identifies its order or offset in the original packet.
- Each fragment must identify the length of the data carried in the fragment.
- Each fragment must inform the device if more fragments are coming after it using the More Fragment flag.
- The IP header is cloned from the first packet to every fragmented packet.
- The **protocol header is not cloned** to the fragmented packets and is only carried on the first packet.

How Hackers Use Fragmentation

By understanding fragmentation, you will be in a position to spot fragmentation attacks. Routers often do not look at every fragment in a fragmentation chain. Usually, it's only the first packet that is analyzed and depending on the rules of how to pass the packet, it is either passed or denied. Hackers rely on the fact that only the first packet in the fragmentation chain contains any protocol headers: TCP, UDP, ICMP and all subsequent fragment packets only contain the IP header.

By carefully crafting the first packet of the fragmentation chain to pass through the device's policy, all subsequent fragmented packets will also pass because the device has already made its decision based on the information provided in the first fragment (assuming the device isn't performing packet reassembly). By placing malicious code in the subsequent fragments, they can now bypass the router, firewall, or IDS system. Some of the most famous attacks that used fragmentation were the "Ping of Death" and "Teardrop".

Why don't devices such as routers analyze every fragmented packet? It is extremely resource intensive to keep track of each fragment's state – requiring each packet to be examined and stored before the decision can be made to either allow or drop the packet. Most routers make the assumption that if one of the fragments is missing or corrupted in the fragmentation chain, the entire packet will be resent. The router forwards all subsequent packets based on the first fragment to speed up transmission and maintain network performance.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



How Foundry Treats Fragmentation

By default, Foundry devices will analyze the first fragment of a fragmentation chain and apply any ACLs to either permit or deny the packet. All subsequent fragments will either be passed or dropped in hardware (JetCore products) based on the decision made from analyzing the first packet. Since most normal applications cannot use the packet if one of the pieces/fragments is missing or bad, this method works well for many applications.

On JetCore products, you can enable fragmentation features that may help secure your environment against fragmentation attacks.

NOTE: Application – Low, Medium, and High Security Models where there are possibilities of fragmentation type attacks (usually the DMZ subnets). Care must be taken to ensure that fragmented packets are not a frequent and normal part of your traffic flows. Use a protocol analyzer to review your traffic for fragmentation over a period of several days before using this feature.

NOTE: The fragmentation support described in this section applies only to JetCore devices and only to hardware-based ACLs. These commands are CPU based so care must be taken when using these features. Applications that use fragmentation frequently, such as NFS and some Microsoft management protocols, may cause performance to degrade if the amount of fragmented traffic is excessive.

CPU Inspection of Fragmented Packets

This command allows you to forward all fragmented packets to the CPU for inspection against the ACLs or to drop all fragments.

Syntax: [no] ip access-group frag inspect | deny

The **inspect** | **deny** parameter specifies whether the fragments to be sent to the CPU for inspection or dropped:

inspect - This option sends all fragments to the CPU for inspection to see if the fragment should be passed or dropped according to any ACLs defined.

deny - This option begins dropping all fragments received by the port as soon as you enter the command. This option is especially useful if the port is receiving an unusually high rate of fragments that are not part of your normal traffic - which can indicate a fragmentation attack.

EXAMPLE:

This example configures Ethernet 1/1 to pass all fragments to the CPU for inspection and to compare each fragment to the ACLs to determine if the fragment should be dropped or passed.

```
BigIron(config)# interface ethernet 1/1
BigIron(config-if-1/1)# ip access-group frag inspect
```


WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Controlling the Fragment Rate

If the device is configured to forward all fragmented packets to the CPU for inspection, there will be some additional CPU processing load. Normal fragmentation on most networks will not cause a heavy load on the CPU, but misconfigured hosts or hackers using fragmentation may introduce large amounts of fragmented packets and add a significant load on the CPU. The following command allows you to control the maximum number of fragments the device or interface can send to the CPU in a one-second interval.

To Set Fragmentation Rate On Entire Chassis

This command allows you to specify the maximum number of fragments that can be sent to the CPU for the entire chassis. Once the maximum level of fragments/second has been reached, the exceed-action is taken and an interval timer is started. The system will take the action specified for the number of minutes specified in the reset-interval.

Syntax: [no] ip access-list frag-rate-on-system <num> exceed-action drop | forward reset-interval <mins>

The **<num>** parameter specifies the maximum number of fragments the device or an individual interface can receive and send to the CPU in a one-second interval. The valid range is 600 – 12800 and the default is 6400.

The **drop | forward** parameter specifies the action to take if the threshold (<num> parameter) is exceeded:

- **drop** - fragments are dropped without filtering by the ACLs
- **forward** - fragments are forwarded in hardware without filtering by the ACLs

The **<mins>** parameter specifies the number of minutes the device will enforce the drop or forward action after a threshold has been exceeded. You can specify from 1 - 30 minutes.

EXAMPLE:

This example configures the JetCore device for a maximum of 10000 fragments/second for the entire chassis. If the maximum is reached, the action is to drop all subsequent fragments for duration of 5 minutes.

```
BigIron(config)# ip access-list frag-rate-on-system 10000 drop reset-int 5
BigIron(config)# write memory
```

To Set Fragmentation Rate On Interfaces

This command allows you to specify the maximum number of fragments that can be forwarded to the CPU from an individual interface. Once the maximum fragments/second has been reached, the exceed-action is taken and an interval timer is started. The system will take the action specified for the number of minutes specified in the reset-interval.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Syntax: [no] ip access-list frag-rate-on-interface <num> exceed-action drop | forward reset-interval <mins>

The **<num>** parameter specifies the maximum number of fragments the device or an individual interface can receive and send to the CPU in a one-second interval. The valid range is 300 – 8000 and the default is 4000.

The **drop | forward** parameter specifies the action to take if the threshold (<num> parameter) is exceeded:

- **drop** - fragments are dropped without filtering by the ACLs
- **forward** - fragments are forwarded in hardware without filtering by the ACLs

The **<mins>** parameter specifies the number of minutes the device will enforce the drop or forward action after a threshold has been exceeded. You can specify from 1 - 30 minutes.

EXAMPLE:

This example configures the JetCore device for a maximum of 1000 fragments/second for all interfaces on the chassis. If the maximum is reached, the action is to drop all subsequent fragments for duration of 5 minutes.

```
BigIron(config)# ip access-list frag-rate-on-interface 1000 drop reset-int 5
BigIron(config)# write memory
```

Dropping All Fragments For IronCore Products

IronCore products can be configured to drop all fragmented packets when the IP header information (source and destination address) matches the ACL - regardless if the accompanying ACL is designed to permit the packet. This command is used if you do NOT want to pass any fragments at all. Since all fragmented packets (besides the first one) have no Protocol Header information (source port, destination port, etc) to compare against the ACL, only the IP header information is used. This can pose a security risk if crafted packets are designed to bypass fragmentation algorithms.

Syntax: [no] ip access-group frag deny

NOTE: This command is applied on an interface level only and was added in release 07.5.04A and later revisions of IronCore software.

EXAMPLE:

This example configures the IronCore device to drop all fragmented packets that are forwarded to the CPU.

```
BigIron(config)# interface E 1/1
BigIron(config-if-1/1)# ip access-group frag deny
BigIron(config-if-1/1)# write memory
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Synchronizing Time

Although configuring accurate time on Foundry devices does not specifically protect your company from being attacked or hacked, it does allow for the centralization of accurate log records. As part of any good security policy, the ability to log, track, and identify the attack and trace the activity back to the attacker is crucial in creating a good "Defense In Depth" strategy. Having accurate logs are very necessary for complying with Corporate Security Policies and are imperative for giving your legal department and law enforcement the means to prosecute the hacker(s).

Without accurate chronological records of the attack or intrusion, your case against the attacker(s) may be dismissed or the evidence gathered may not be admissible in court. Not only is accurate time crucial to security logging, it is also required for accurate network monitoring, correlation of log files, accounting, troubleshooting, and AAA authentication and authorization.

Network Time Protocol (NTP) has four modes of operation:

Client	As a client, the device allows its clock to be set and synchronized by an external NTP server. NTP clients can use multiple NTP servers to set their time and clients will not supply NTP services to any other devices.
Server	NTP servers are configured to supply accurate time to NTP Clients and can be configured to supply time to all NTP clients or just specific NTP clients. NTP servers will not synchronize their time with information from their NTP clients.
Peer	Peer NTP servers are used to provide redundancy. Peers share their time information with each other NTP peers to keep their times synchronized.
Broadcast/Multicast	NTP servers that use the broadcast/multicast configuration can be used to push its synchronization information to all clients. Broadcast mode requires that the clients be on the same broadcast subnet as the server and multicast requires that clients and servers have multicast access available and configured.

Foundry devices are used as clients in your NTP configuration and you must design a robust NTP environment to support accurate time of your network devices. There are various ways of doing this:

Simplest Method	The most cost effective way to obtain accurate time for all your Foundry devices is to use your ISP's timeservers. If you do not have this information, call their help desk to get at least two NTP server addresses. This may not provide the best solution if you have a large environment or if your corporate security policy prevents the NTP protocol from passing through the firewall.
-----------------	---

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Central Server Method

You can also create your own NTP environment by purchasing time servers and setting up several NTP servers which peer to each other. This provides redundancy and allows your network devices to use at least two time sources. The time servers can obtain accurate time from an authoritative source on the Internet or from dedicated antennas.

Setting SNTP on Foundry Devices

The Simple Network Time Protocol (SNTP) command is used on Foundry devices to synchronize their clocks with an external NTP server. NTP is very efficient and requires no more than one packet per minute to synchronize all devices on your LAN to within one millisecond and to within 10 milliseconds on your WAN.

Syntax: `sntp server <ip-addr> | <hostname> [<version>]`

The **<version>** parameter specifies the NTP version the server is running and can be from 1 - 4. The default is 1. You can configure up to three NTP servers by entering three separate **sntp server** commands.

By default, Foundry devices poll the NTP server every 30 minutes (1800 seconds). This value can be changed with the "sntp poll-interval" command.

Syntax: `[no] sntp poll-interval <1-65535>`

Time synchronization can also be manually polled with the following command.

Syntax: `sntp sync`

EXAMPLE:

This example configures the Foundry device with a primary NTP server 10.96.1.10 and a secondary NTP server 10.34.1.10. Both NTP servers are running version 3 of the NTP protocol and the SNTP polling interval has been changed to every 5 minutes. A manual poll is used as the last command to start the synchronization process.

```
BigIron(config)# sntp server 10.96.1.10 3
BigIron(config)# sntp server 10.34.1.10 3
BigIron(config)# sntp poll-interval 300
BigIron(config)# exit
BigIron# sntp sync
```

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Other Important SNTP Commands

Displaying SNTP Associations
Displaying SNTP Status

Syntax: show sntp associations
Syntax: show sntp status

NOTE: Foundry devices also have other commands that allow you to set the system clocks. The "clock set" command along with the ability to set daylight savings time are used to set time and date parameters for each Foundry device. The drawback of using these commands is that the time and date are not kept after a system reboots and the internal clocks are not synchronized in any way. The date and time must be entered each time the device is brought back up. For this reason, it is best to use SNTP to set accurate time information.

Implementing Logging Strategies

Making sure your Foundry device's time parameters are accurate and synchronized with NTP servers is the first step in ensuring accurate network logs. Accurate logs can help you troubleshoot your network more effectively, provide audit trails, spot signs of problem areas that can become outages, and show intrusions or attack activity.

Logs are only useful if they are monitored on a regular basis. Having logs with accurate and useful information will be of little use if they contain so much information that they overwhelm the reviewer to the point where they don't perform regular daily analysis of the information. Your logging strategy should coincide with your Corporate Security Policy and you should make sure there are adequate resources to perform the necessary daily operations of maintaining your network and reviewing the necessary log information. As a general rule, the more the system is exposed to the dangers of intrusion or hacking, the more monitoring and proactive reviewing is required.

To make the task of gathering and reviewing log information easier, many third-party applications are available to help automate the task of analyzing log files, filtering out unwanted events, and selecting events of importance. A good strategy for setting up these automated log-reviewing programs is to:

- Instruct them to ignore all events that you know are safe or unimportant
- Highlight the events that you know are dangerous
- Email or print out all other events so they can be reviewed by the operations team

Fine-tuning log-reviewing programs over several weeks will start revealing the dangerous events and the number of "false-positives" will start decreasing as you groom and educate the application. It will take some time and resources to set the automated procedure up correctly, but it's well worth the time and effort as a custom configuration emerges from your efforts.

NOTE: Centralize your logging efforts if possible. This will make it easier to develop trends and coordinate the information that may be necessary to spot attacks across multiple devices. It will also make your logging and review process more streamlined and easier to perform.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Foundry devices can log events in several ways:

Console Logging	Console log messages are displayed on the console port or through a remote connection to the Privilege EXEC and Console CLI. You must be connected to the console port or remotely logged into the CLI to see these messages or to log the screen output.
Buffered Logging	Buffered logs are stored in the Foundry device's logging buffer memory space. There is a limited memory space in the buffered area and newer log entries overwrite the oldest entries when the buffer is filled.
Syslog	Foundry devices allow you to specify a maximum of six external Syslog servers to send event information to - allowing you to store all device messages.
SNMP Traps	Foundry devices can be configured with SNMP to send alerts or traps to third-party SNMP servers to log specific events.
AAA Accounting	Foundry devices can be configured to send events and system messages to external TACACS+ and RADIUS servers using the AAA authentication, authorization, and accounting features.

NOTE: Aside from system log messages, logging denied ACLs could help you spot intrusion or hack attempts. For example, by logging the denies in your Anti-Spoofing ACLs, you can see which hosts are trying to use spoofing to either access your network or send packets out of your network. Also note that logging ACLs requires CPU overhead and plan accordingly.

Setting Up Logging Features

To help you setup logging on your Foundry devices, several commands are available to control the log details, where to send the logs, and how to view the logs. By default, all Foundry devices have the following setup for logging features:

Messages of all severity levels (Emergencies - Debugging) are logged

- Up to 50 messages are retained in the local Syslog buffer
- No Syslog server is specified

For a complete logging feature list of your Foundry device, refer to your hardware and software revision of the release notes.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Enabling Syslog

By default, Foundry devices are configured for system logging. To disable or to reset logging on your Foundry device, use the following command:

Syntax: [no] logging on [<udp-port>]

The **<udp-port>** allows you to reconfigure the default port used for the Syslog service. The Foundry default is UDP 514.

EXAMPLE:

This example turns logging on for the Foundry device.

```
BigIron(config)#logging on
BigIron(config)#
```

Console Logging

By default, Foundry devices write their log event information to the local system log. Sometimes it is very valuable to see the real-time messages as they are being written without having to issue the "show logging" command to refresh the screen with the latest log events. You can enable real-time Syslog monitoring for the console's serial interface or for a Telnet or SSH session.

Use the following command to enable real-time logging on the console serial interface:

Syntax: [no] logging console

```
BigIron(config)# logging console
```

Use the following command to enable real-time logging on a Telnet or SSH session:

Syntax: [no] terminal monitor

```
telnet@BigIron# terminal monitor
Syslog trace was turned ON
```

Buffered Logging

Foundry devices can hold up to a maximum 100 or 1000 log messages (depending on the device and the default is 50 log messages) in its local logging buffer before overwriting them. These local system logs are not saved and will be lost in the event of a power failure or a reboot. New log events overwrite the old when the maximum message limit has been reached. The local logs are turned on by default and can be managed and viewed with the following log commands.

Displaying local buffer log
Clearing the local buffer log

Syntax: show logging
Syntax: clear logging [dynamic-buffer | static-buffer]

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Foundry devices use two buffers to store system log information: static-buffer and dynamic-buffer. The static-buffer region stores system hardware messages such as power supply failures, fan failures, temperature warnings, or shutdown messages. Static-buffer messages are chronologically written to the static-buffer and only one message is kept for each event with the event message overwriting the previous one. For example, if you look at the fan failure event, only the last failure message will be displayed, even if there were four previous fan failure messages.

Dynamic-buffer messages log all other message types and will fill the buffer space chronologically as they are received and will overwrite the oldest entries when the buffer is filled.

EXAMPLE:

A sample of a Syslog display on a Foundry BigIron switch:

```
BigIron(config)# show logging

Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
  Buffer logging: level ACDMEINW, 3 messages logged
  Level code: A=alert C=critical D=debugging M=emergency E=error
              I=informational N=notification W=warning

Static Log Buffer:
Dec 15 11:04:14:A:Fan 1, fan on right connector, failed

Dynamic Log Buffer (50 entries)
Dec 15 17:24:58:I:Interface Ethernet 2/8, state up
Dec 15 18:07:44:I:Bridge topology change, vlan 4096, interface 4, changed
state to forwarding
Dec 15 18:06:14:I:Warm start
```

Changing Local Buffer Entries

By default, Foundry devices hold 50 system log entries in its local log buffer. This value can be changed to reflect your needs – a maximum of 1000 system log entries can be held locally on Layer 3 Switches and a maximum of 100 entries on other devices.

Syntax: logging buffered <num>

EXAMPLE:

This example sets the maximum system log entries to 100 for the Foundry device.

```
BigIron(config)# logging buffered 100
```

Setting Up External Syslog Servers

When you setup external Syslog servers, Foundry devices will write the system information to both the local system log and to the external Syslog servers defined on the device. Using an external Syslog server is much more robust and allows you to record much more information than the local buffered system log or the console logging methods discussed earlier. Long-term storage of system events is crucial for a good security and monitoring design. Most Unix operating systems come with Syslog servers and there are utilities that enable Syslog servers on Windows platforms.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



The advantages of using an external Syslog server to store your device logs include:

- Log events are not overwritten like they are on local system logs, creating audit trails
- Centralization of log events is possible
- Coordination of log events are possible through the Syslog's timestamp
- Log events are not lost when a device is rebooted
- Multiple severity levels are supported
 - Emergencies
 - Alerts
 - Critical
 - Errors
 - Warnings
 - Notifications
 - Informational
 - Debugging

NOTE: Application – Low, Medium, and High Security Models. External Syslog servers are highly recommended for all implementations and you should configure all Foundry devices to send its system logs to two external Syslog servers if possible. All Foundry devices should also have their system clocks synchronized with two NTP servers if possible.

NOTE: For a full explanation of using Syslog servers with Foundry devices, refer to the following web site: http://www.foundrynet.com/services/documentation/sribcg/Syslog_msgs.html

Foundry devices support up to six external Syslog servers and uses UDP port 514 for NTP. To redirect your log events and messages to an external Syslog server, use the following command to setup the Syslog servers on your Foundry device.

Syntax: logging <ip-addr> | <server-name>

EXAMPLE:

This example configures the Foundry device to use three external Syslog servers.

```
BigIron(config)# logging 10.96.1.99
BigIron(config)# logging 10.98.1.99
BigIron(config)# logging 10.99.1.99
BigIron(config)# write memory
```

Disabling Logging Of A Message Level

As stated earlier, Foundry devices log eight different message level types by default. If you don't need certain message levels logged, you can disable them with the **"no logging buffered"** command.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Syntax: [no] logging buffered <level> | <num-entries>

The <level> parameter allows you to specify one of the eight message levels.

- Emergencies
- Alerts
- Critical
- Errors
- Warnings
- Notifications
- Informational
- Debugging

EXAMPLE:

This example configures the Foundry device to stop system logging for message levels debugging and informational.

```
BigIron(config)# no logging buffered debugging
BigIron(config)# no logging buffered informational
```

Classification Using Syslog Facilities

External Syslog servers allow you to classify or separate the system log messages into different services using the “facilities” parameter. This is advantageous if your installation uses different Syslog servers to capture different categories of system log messages from various devices. This separation simplifies the reviewing and auditing process of log files and supports the separation of duties for large network operation teams.

By default, Foundry devices use the “user” facility to send its system logs to external Syslog servers. You can change the default facility with the following command:

Syntax: logging facility <facility-name>

The <facility-name> can be one of the following:

- kern - kernel messages
- user - random user-level messages
- mail - mail system
- daemon - system daemons
- auth - security/authorization messages
- Syslog - messages generated internally by Syslog
- lpr - line printer subsystem
- news - netnews subsystem
- uucp - uucp subsystem
- sys9 - cron/at subsystem
- sys10 - reserved for system use
- sys11 - reserved for system use
- sys12 - reserved for system use
- sys13 - reserved for system use

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



- sys14 - reserved for system use
- cron - cron/at subsystem
- local0 - reserved for local use
- local1 - reserved for local use
- local2 - reserved for local use
- local3 - reserved for local use
- local4 - reserved for local use
- local5 - reserved for local use
- local6 - reserved for local use
- local7 - reserved for local use

NOTE: You can specify only one facility for each Foundry device. If you configure the Foundry device to use two Syslog servers, the device uses the same facility on both servers.

EXAMPLE:

This example configures the Foundry device to use the "local0" facility to send its system logs.

```
BigIron(config)# logging facility local0
```

TACACS+ and RADIUS Logging

TACACS+ and RADIUS support external logging (accounting) through the AAA functionality. Please refer the TACACS+ and RADIUS sections for more information on setting up the associated accounting features.

Other Logging Recommendations

Other areas that can be logged by the system log function include Access Control Lists (ACLs) violations and remote management access to Foundry devices if they were restricted through ACLs. By adding the "log" parameter after the proper ACL, violations can be logged into the system log. This can help you determine when and who is trying to spoof IP addresses on the network or who is trying to login to your network devices.

You must be careful in logging ACL restrictions as placing the "log" option on the wrong ACL can cause a lot of traffic to be logged in your system logs. Also note that ACL logging requires action on part of the CPU and turning on the log function for ACLs that return very high positive match conditions can degrade device performance.

ACL logging will identify the date, time, and type of violation. On extended ACLs you will also get additional information with regards to the source of the violation.

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



EXAMPLE:

This example illustrates how turning on logging can help identify IP Spoofing on your network. Subnet 198.30.15.0/24 is the IP address range of the DMZ. Any external packets entering the DMZ should not claim to be from this IP range and the ACL denies access and creates an entry in the system log.

```
edgerouter(config)# access-list 10 deny 198.30.15.0/24 log
edgerouter(config)# access-list 10 deny 127.0.0.0/8
edgerouter(config)# access-list 10 deny 10.0.0.0/8
edgerouter(config)# access-list 10 deny 172.16.0.0/12
edgerouter(config)# access-list 10 deny 192.168.0.0/16
edgerouter(config)# access-list 10 deny 224.0.0.0/4
edgerouter(config)# access-list 10 deny 240.0.0.0/5
edgerouter(config)# access-list 10 deny 255.255.255.255/32
edgerouter(config)# access-list 10 permit any
```

```
edgerouter(config)# interface ethernet 16
edgerouter(config-if-16)# ip access-group 10 in
```

EXAMPLE:

This example shows how internal ACLs placed on inbound points of your router interfaces can help stop and log outbound IP spoofing attempts. Access list 21 allows only 10.32.0.0/16 traffic to go through and it is placed on the router interface E1/1 that is the gateway for the 10.32.0.0/16 subnet as an inbound ACL. All other traffic with any other source address is dropped and logged.

```
internalrtr(config)# access-list 21 permit 10.32.0.0/16
internalrtr(config)# access-list 21 deny any log

internalrtr(config)# interface ethernet 1/1
internalrtr(config-if-1/1)# ip address 10.32.0.254/16
internalrtr(config-if-1/1)# ip access-group 21 in
```

EXAMPLE:

This example shows how failed remote access attempts from unauthorized management stations can be logged in the system log. In order to use this feature, you must setup restrictions using ACLs to regulate which management stations can and can't access the Foundry devices. For more information on this topic, refer to the section "Restricting Management Access". You can use the method for restricting and logging SSH, Telnet, SNMP, and Web management access.

Access List 10 permits a single host of 10.1.0.25 and all other hosts from subnets 10.2.1.0/24, 10.2.3.0/24, and 10.2.5.0/24. No other clients are allowed to Telnet into this device and any attempts are logged.

```
BigIron(config)# access-list 10 permit host 10.1.0.25
BigIron(config)# access-list 10 permit 10.2.1.0 0.0.0.255
BigIron(config)# access-list 10 permit 10.2.3.0 0.0.0.255
BigIron(config)# access-list 10 permit 10.2.5.0/24
BigIron(config)# access-list 10 deny any log
```

```
BigIron(config)# telnet access-group 10
BigIron(config)# write memory
```

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



NOTE: You can also log the permit ACLs to track the successful logons as well. Since there shouldn't be too much traffic associated with remote management, there should be very little performance considerations.

Locking Down Ports

One of the easiest and most common ways intruders access networks from the inside of your corporation is by simply plugging into an unused RJ-45 network outlet to get a live network connection that isn't locked down. With the popularity of DHCP addressing, the unauthorized user is usually up and running in a matter of seconds. Foundry offers several different security solutions for limiting unauthorized access.

1. As part of your security policy, lock down all unused ports by disabling them. The "disable" command can be used on a range of ports or on individual ports.
2. Use MAC address locking to associate a MAC address with the interface to prevent unauthorized use by another host. The Port Security feature can be turned on for the entire chassis to protect all interfaces (*except uplink ports*) or it can be turned on at the interface level to protect each port individually. A maximum of 64 MAC addresses can be locked by each interface.
3. Implement 802.1x port authentication in high security installations. This requires the user to authenticate to a RADIUS server before network access is granted to the host.

EXAMPLE - Disable Port

This example quickly disables all the unused ports on blade 4 ports 1 through 24.

```
FastIron(config)# interface E4/1 to 4/28
FastIron(config-mif-e4/1-24)# disable
FastIron(config-mif-e4/1-24)# write memory
```

EXAMPLE – Port Security MAC Lock

This example turns on port security for interface Ethernet 7/1 and enables MAC address locking for one MAC address. The workstation that will be using this interface is in an unsecured location so it will be locked down indefinitely.

```
BigIron(config)# int e 7/1
BigIron(config-if-e100-7/1)# port security
BigIron(config-port-security-e100-7/1)# enable
BigIron(config-port-security-e100-7/1)# maximum 1
```

EXAMPLE – 802.1x Port Authentication

This example will setup the Foundry device to use AAA RADIUS authentication for 802.1x and enable it for the entire chassis. The second command configures the device to use RADIUS server "209.157.22.99" with the authentication and accounting port numbers and specifies the key.

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Configure the Foundry device to use RADIUS for authenticating 802.1x security.

```
BigIron(config)# aaa authentication dot1x default radius
```

Configure the device to use RADIUS server "209.157.22.99" with the authentication and accounting port numbers of "1812" and "1813" and specifies the key to authenticate to the RADIUS server.

```
BigIron(config)# radius-server host 209.157.22.99 auth-port 1812 acct-port 1813 default key radpassword dot1x
```

Enable 802.1x security for the Foundry device.

```
BigIron(config)# dot1x-enable
```

Enable 802.1x port security for all interfaces on the device.

```
BigIron(config-dot1x)# enable all
```

To configure 802.1x for individual ports, you can use the "enable" command with the port number. A range can also be specified.

```
BigIron(config-dot1x)# enable Ethernet 2/1 to 2/24
BigIron(config-dot1x)# enable Ethernet 3/1 to 3/12
BigIron(config-dot1x)# write memory
```

For all interfaces using 802.1x authentication, enable the control mode to either "force-authorized", "force-unauthorized", or "auto". Auto leaves the port in unauthorized mode until the RADIUS server validates the authentication.

```
BigIron(config)# interface e 3/1
BigIron(config-if-3/1)# dot1x port-control auto
```

NOTE: For more information on MAC Address Locking and 802.1x authentication, refer to the *Foundry Switch and Router Command Line Interface Reference* and *Foundry Security Guide*.

Staying On Top of OS Versions

Security is an evolving task that changes frequently. As part of your security policy and practice, it is imperative that you keep up-to-date on software releases for your Foundry devices and apply the necessary code upgrades to keep your device's OS up to date.

If your devices are under warranty or your company has an active maintenance contract, the latest versions of all Foundry device OS's can be found on its web site at:

<http://www.foundrynet.com/services/support/index.html>

WHITE PAPER: IRONSHIELD BEST PRACTICES

HARDENING FOUNDRY ROUTERS & SWITCHES



Physical Security

Besides implementing all of the security features discussed in this document, it is very important that physical access to your network devices be secured. Good physical security is at the foundation of keeping your network secure. Nothing will prevent a knowledgeable intruder from gaining access to your devices if they can physically gain access to the device itself. Your Corporate Security Policy should contain the minimum levels of physical security required for your network devices and how they should be secured.

A good physical security policy will address and define the following levels of physical security:

- A secure location that places routers and switches in a locked room. The room should not have other access points other than the locked door(s) – such as raised floors or false ceilings that are not properly extended to prevent access from adjacent areas.
- All doors should have key locks or badge readers that limit the access to authorized personnel only. Solid doors without windows should be used for added physical security. Automatic door closers should be used to make sure the door isn't accidentally left open. They should never be propped open.
- Your locks should match your security requirements. For example, high security installations may require biometric readers over key code locks or keys that can be duplicated or lost.
- Limiting only authorized personnel into your network rooms is critical in keeping your network secure. Keep the number of authorized personnel down to a minimum and if possible, design the network rooms to be separate from other corporate functions such as telecom to keep the access lists separated.
- Secure backups should be kept for all device configurations and OS versions. You should encrypt the backup configurations and keep them on a secure server with limited access to only authorized personnel. Never leave your configurations on TFTP servers after they have been downloaded.
- Protection against fire, water, heat, humidity, dirt and dust is a requirement as these elements can cause severe damage to network devices.
- Cable splice points, repeater panels, or patch panels should be physically secured to prevent packet snooping.
- Protection against electrical surges or brownouts is required to ensure a reliable network. Uninterruptible Power Supplies (UPS's) and power filters should be installed for mission critical network components.
- Keep the rooms neat and tidy. Remember, it is a network room and not a storeroom.

Recommended Security Links

DDoS Information Sites

Sites with valuable information on Distributed Denial of Service Attack and Tools can be found at the following web sites:

<http://staff.washington.edu/dittrich/misc/ddos>

<http://www.theorygroup.com>

<http://razor.bindview.com>

Security Information Sites

Sites with general and specific attack information can be found on the following web sites:

www.cert.org

<http://www.sans.org>

<http://www.microsoft.com/technet/security> (Microsoft security site)

<http://www.sans.org/top20> (top 20 SANS/FBI attacks and how to secure them)

http://www.iss.net/security_center/advice/Exploits/Ports/ (Site for TCP/IP port numbers)

<http://www.gocsi.com/> (Computer Security Institute)

<http://www.nsa.gov/isso/index.html> (NSA security hardening recommendations for various OS's)

Total Cost of Ownership

Tools that can help you diagnose Total Cost of Ownership or give examples on how to calculate IT costs and security costs can be found at the following web sites:

<http://www.networkbuyersguide.com/search/105196.htm>

<http://www.strassmann.com>

WHITE PAPER: IRONSHIELD BEST PRACTICES HARDENING FOUNDRY ROUTERS & SWITCHES



Foundry Networks, Inc.
Headquarters
2100 Gold Street
P.O. Box 649100
San Jose, CA 95164-9100

U.S. and Canada Toll-free: (888) TURBOLAN
Direct telephone: +1 408.586.1700
Fax: 1-408-586-1900
Email: info@foundrynet.com
Web: <http://www.foundrynet.com>

Foundry Networks, BigIron, EdgeIron, FastIron, NetIron, ServerIron, and the "Iron" family of marks are trademarks or registered trademarks of Foundry Networks, Inc. in the United States and other countries. All other trademarks are the properties of their respective owners.

© 2003 Foundry Networks, Inc. All Rights Reserved.