CHAPTER 3

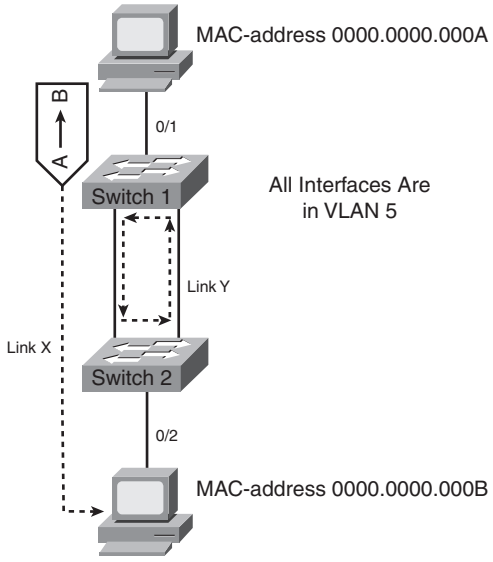
Attacking the Spanning Tree Protocol

Radia Perlman, a distinguished engineer at Sun Microsystems, named as one of the 20 most influential people in the industry in the 25th anniversary issue of *Data Communications* magazine and the original inventor of the 802.1D spanning-tree specification recently had a few words to say about the protocol: "It's time to redo (one of the Internet's most widely used technologies) in a way that is more robust and gives more efficient paths."[1]

Introducing Spanning Tree Protocol

Chapter 2, "Defeating a Learning Bridge's Forwarding Process," explained how Ethernet switches build their forwarding tables by learning source MAC addresses from data traffic. When an Ethernet frame arrives on a switch port in VLAN X with a destination MAC address for which there is no entry in the forwarding table, the switch floods the frame. That is, it sends a copy of the frame to every single port in VLAN X (except the port that originally received the frame). Although this is perfectly fine in a single-switch environment, interesting side effects are observed in multiswitch topologies, as Figure 3-1 shows. The figure represents a simple network composed of two LAN switches interconnected by two Ethernet links.

Figure 3-1 *Basic Network Setup*



In the next steps, MAC addresses are conveniently shortened to a single-letter format for clarity. A legitimate Ethernet MAC address is actually made up of 6 bytes. The following sequence of events occurs when an application on the top PC (MAC address A) communicates with the bottom PC (MAC address B):

1. The top PC sends a frame to the bottom PC (destination MAC address B).
2. Switch 1 learns that MAC address A is off port 0/1.
3. Switch 1 looks up MAC address B; no match is found.
4. Switch 1 sends out the frame on link X and Y (a process known as *flooding*).
5. Switch 2 receives the frame from A to B on link X and updates its forwarding table. (A is on link X.)

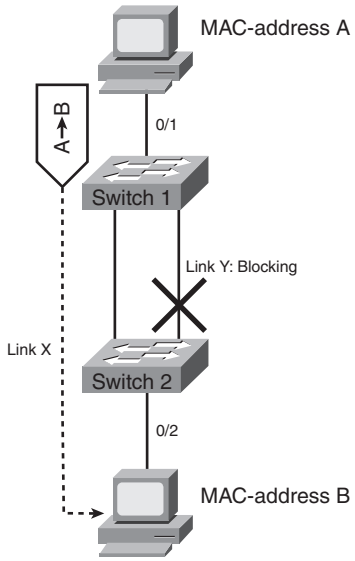
   A split-second later, switch 2 receives the exact same frame on link Y; this time, it causes a new update to the forwarding table. This is known as a race condition—whichever MAC address arrives first wins the race and gets installed in the forwarding table.
6. Switch 2 looks up MAC address B; no match is found. (B hasn't talked yet.)
7. Switch 2 sends out the frame on port 0/2 *and* link Y (or X, depending on the outcome of the race condition described in Step 5).

8. Switch 1 and PC B both receive the frame; however, this frame causes switch 1 to again update its forwarding table. (MAC address A is now off link Y or X.)
9. Return to Step 3 and loop forever. Even if B talks, nothing changes because both switches constantly update their forwarding tables with incorrect information (because of the never-ending packet loop).

There is no such thing as a Time to Live (TTL) field in Ethernet headers. No routing protocol distributes information related to MAC addresses and their whereabouts. Simply put, short of a power or link failure, nothing can stop the packets from looping endlessly between switch 1 and 2. There's no need for a broadcast or multicast frame; a simple unicast frame does fine.

The problem is hardly new. After Radia Perlman's work in the early 1990s, the IEEE ratified her protocol work into a standard known as 802.1D. 802.1D defines the original Spanning Tree Protocol (STP), whose task is to disable redundant paths from one end of the Layer 2 network to another, thereby achieving two goals: no packet duplication or loops while still providing automatic traffic rerouting in case of failure. If switch 1 or switch 2 (or both) were running the STP, the topology represented in Figure 3-1 would logically appear as what's shown in Figure 3-2.

Figure 3-2 *Loop-Free Topology Calculated by STP*



With link Y disabled by the spanning-tree algorithm running on switch 2, packets from the top PC to the bottom PC can no longer loop forever.

STP is an extremely pervasive protocol; it keeps virtually every single existing Ethernet-based LAN network loop free.

Types of STP

Today, various flavors of STP exist, either as IEEE specs (802.1Q Common STP, 802.1w Rapid STP, 802.1s Multiple STP) or as proprietary vendor extensions. All of them function in similar fashions; they are typically differentiated only by the time they need to recalculate an alternate topology in case of a link failure. Proper STP operation is critical, yet it is so fragile, which this chapter is about to demonstrate.

Understanding 802.1D and 802.1Q Common STP

Originally defined in 1993, the IEEE 802.1D document specifies an algorithm and a protocol to create a loop-free topology in a Layer 2 network. (At that time, there was no concept of VLAN.) The algorithm also ensures automatic reconfiguration after a link or device failure. The protocol converges slowly by today's standards: up to 50 seconds (sec) with the default protocol timers. The 802.1Q specification later augmented the 802.1D by defining VLANs, but it stopped short of recommending a way to run an individual spanning-tree instance per VLAN—something many switch vendors naturally implemented using proprietary extensions to the 802.1D/Q standards.

Understanding 802.1w Rapid STP

Incorporated in the 2004 revision of the 802.1D standard, the 802.1w (Rapid Reconfiguration of Spanning Tree) introduced significant changes, primarily in terms of convergence speeds. According to the IEEE, motivations behind 802.1w include the following:

- The desire to develop an improved mode of bridge operation that, while retaining the plug-and-play benefits of spanning tree, discards some of the less desirable aspects of the existing STP (in particular, the significant time it takes to reconfigure and restore service on link failure/restoration).
- The realization that, although small improvements in spanning-tree performance are possible by manipulating the existing default parameter values, it is necessary to introduce significant changes to the way the spanning-tree algorithm operates to achieve major improvements.
- The realization that it is possible to develop improvements to spanning tree's operation that take advantage of the increasing prevalence of structured wiring approaches, while still retaining compatibility with equipment based on the original spanning-tree algorithm.

The bottom line is that 802.1w usually converges in less than a second. All Cisco switches running recent software versions make 802.1w the default STP.

Understanding 802.1s Multiple STP

The 802.1s supplement to IEEE 802.1Q adds the facility for bridges to use multiple spanning trees, providing for traffic belonging to different VLANs to flow over potentially different paths within the virtual bridged LAN. The primary driver behind the development of 802.1s is the increased scalability it provides in large bridged networks. Indeed, an arbitrary number of VLANs can be mapped to a spanning-tree instance, rather than running a single spanning-tree instance per VLAN. The loop-breaking algorithm now runs at the instance level instead of at the individual VLAN level. With 802.1s, you can, for example, map a thousand VLANs to a single spanning-tree instance. This means that all these VLANs follow a single logical topology (a blocked port blocks for all those VLANs), but the reduction in terms of CPU cycles is significant.

STP Operation: More Details

To understand the attacks that a hacker is likely to carry out against STP, network administrators must gain a solid understanding of STP's inner workings. The protocol builds a loop-free topology that looks like a tree. At the base of the tree is a root bridge—an election process takes place to determine which bridge becomes the root. The switch with the lowest bridge ID (a concatenation of a 16-bit user-assigned priority and the switch's MAC address) wins. The root-bridge election process begins by having every switch in the domain believe it is the root and claiming it throughout the network by means of Bridge Protocol Data Units (BPDU). BPDUs are Layer 2 frames multicast to a well-known MAC address in case of IEEE STP (01-80-C2-00-00-00) or vendor-assigned addresses, in other cases. When receiving a BPDU from a neighbor, a bridge compares the sender's bridge ID with its own to determine which switch has the lowest ID. Only the one with the lowest ID keeps on generating BPDUs, and the process continues until a single switch wins the designated root-bridge election. STP assigns roles and functions to network ports. Every nonroot bridge has one root port: It is the port that leads to the root bridge.

STP uses a path cost–based method to build its loop-free tree. Every port is configured with a port cost—most switches are capable of autoassigning costs based on link speed.

A port's cost is inversely proportional to its bandwidth. Each time a port receives a BPDU, the port's path cost is added to the path cost contained in the BPDU. The root sends BPDUs with the path cost equal to 0, and the cost keeps increasing as the network diameter increases. When two BPDUs are received on a switch because of redundant links in the network, the one with the higher cost is logically disabled—it is put in *blocked* mode. The bridge that is responsible for forwarding packets on a given segment is called the designated bridge. After a while, ranging from less than a second to just under a minute depending on

the STP flavor, the network converges and a single-rooted loop-free tree is built. Before a port transitions to forwarding, it goes through several states:

- **Disabled**. The port is electrically inactive and does not send or receive any traffic. Once enabled, the port transitions to the next state (blocking).
- **Blocking**. Discards all data frames except BPDUs.
- **Listening**. Switches listen to BPDUs to build the loop-free tree. Data packets are not forwarded (15 sec by default with 802.1D timers).
- **Learning**. Forwarding tables are built using the source MAC addresses of data frames; data frames are not forwarded.
- **Forwarding**. Data traffic. At this point, the port is fully operational.

NOTE Although this chapter paints a detailed portrait of STP's inner workings, we recommend that you look at the reference material available online[2] if you are interested in a more detailed overview.

After the network converges, STP network-wide timers maintain its stability. (A network can be a VLAN.)

Network-Wide Timers

Several STP timers exist:

Hello. Time between each BPDU that is sent on a port. By default, this time is equal to 2 sec, but you can tune the time to be between 1 and 10 sec.

Forward delay. Time spent in the listening and learning state. By default, this time is equal to 15 sec, but you can tune the time to be between 4 and 30 sec.

Max age. Controls the maximum length of time that passes before a bridge port saves its configuration BPDU information. By default, this time is 20 sec, but you can tune the time to be between 6 and 40 sec.

Each configuration BPDU contains these three parameters. In addition, each BPDU configuration contains another time-related parameter, known as the *message age*. The message age is not a fixed value. The message age contains the length of time that has passed since the root bridge initially originated the BPDU. The root bridge sends all its BPDUs with a message age value of 0, and all subsequent switches add 1 to this value. Effectively, this value contains the information on how far you are from the root bridge when you receive a BPDU.

In 802.1D, bridges actually have no idea whether their BPDUs are heard by neighboring switches. For example, the root bridge is not sure that everyone acknowledges its presence—the protocol contains no provision to ensure this. The protocol simply relies on the timers (as just explained) to assume BPDUs are properly delivered to every bridge in the network. Table 3-1 represents an 802.1D BPDU.

Table 3-1 *802.1D BPDU Frame Format*

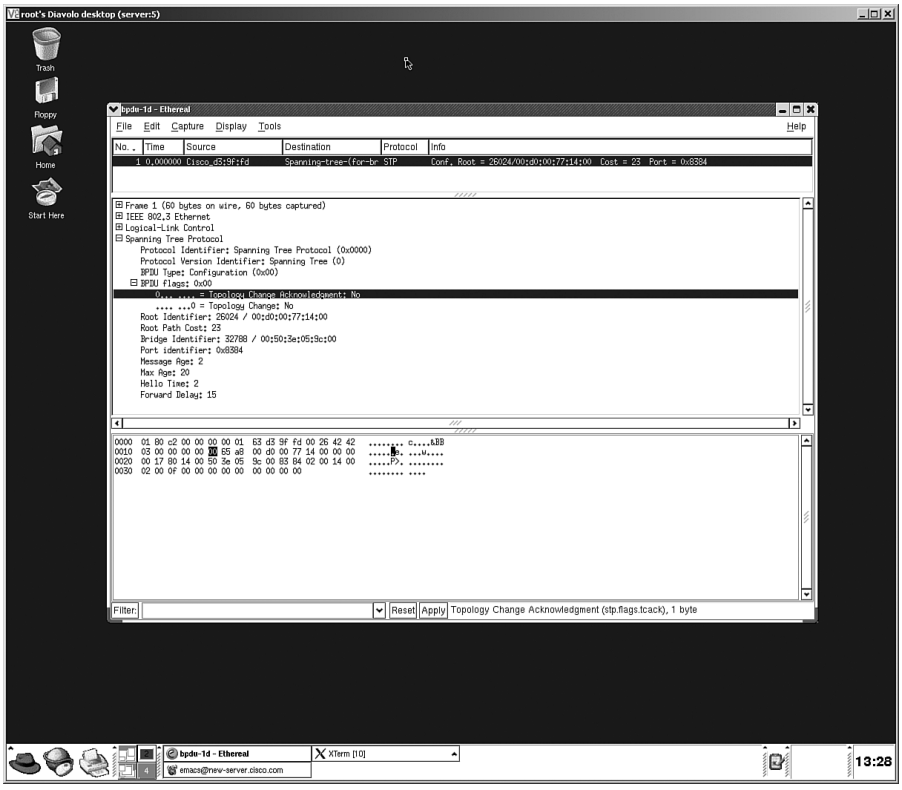
Field	Value
Destination MAC	**01 80 c2 00 00 00** IEEE reserved BPDU MAC
Source MAC	**00 00 0c a0 01 96** Port's MAC address
LENGTH	**00 26**
LLC HEADER	
Destination Service Access Point	**42**
Source Service Access Point	**42**
Unnumbered Information	**03**
PROTOCOL	**00 00**
PROTOCOL VERSION	**00**
BPDU TYPE	**00**
BPDU FLAGS	**00**
ROOT ID	**20 00 00 d0 00 f6 ba 04**
PATH COST	**00 00 00 00**
BRIDGE ID	**20 00 00 d0 00 f6 ba 04**
PORT	**81 14**
MESSAGE AGE	**00 00**
MAXIMUM AGE	**14 00**
HELLO TIME	**02 00**
FORWARD DELAY	**0f 00**

In a converged network, the root bridge sends a BPDU out each port every *hello interval* (2 sec, by default). Every BPDU contains an age field that represents how long it has been in transit. It starts from 0 at the root and increases as the BPDU makes its way through the switched network. A maximum valid age is defined for the network (*max_age* parameter—20 sec, by default). When a BPDU is received on a port, the switch extracts the age contained in the BPDU and starts running a port clock initialized with that value. For example, if the BPDU is 6 sec old, the clock starts counting from 6. Normally, the next

BPDU is supposed to arrive 2 sec later, but because of various conditions (packet loss, unreliable software, excessive CPU utilization, unidirectional links, and so on), BPDUs are known to sometimes fail to show on time. Meanwhile, the port clock runs until it reaches max_age. If it reaches max_age, the bridge starts the election process again, claiming to be the root! Ports go back to blocking/listening/learning before finally forwarding, potentially causing massive traffic blackouts.

Another property of the STP is its ability to influence the forwarding table's aging time by using a particular bit in the BPDU. Figure 3-3 shows the Flags field found in every BPDU.

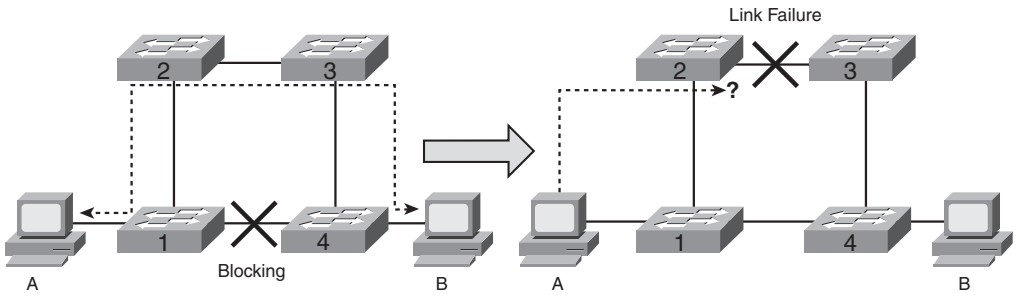
Figure 3-3 *BPDU Packet Capture —TC Bit*



In 802.1D, the Flags field can take two values: 1000 0000 or 0000 0001. When the low-order bit is set, it indicates that the BPDU is actually a topology-change notification (TCN) BPDU. It is a lightweight BPDU whose purpose is to inform the upstream switches all the way to the root bridge that a connectivity event occurred on this switch. A switch sends a TCN BPDU whenever a link or port transitions up or down. Bridges located between the originator of the TCN BPDU and the root immediately acknowledge the reception of the

TCN BPDU, without being certain that the root still exists. When the TCN BPDU finally reaches the root bridge, it acknowledges this by setting the high-order bit of the Flags field (TC-ACK bit) in BPDU it generates. This notifies every bridge to reduce its forwarding table's aging time to *forward_delay* sec (15, by default). The TC bit is set for a certain period of time (max_age + forward_delay sec, or 35 sec with timers using default values). Figure 3-4 shows a scenario where this mechanism plays a crucial role in restoring network connectivity faster.

Figure 3-4 *TC Bit Plays a Crucial Role*



Suppose traffic flows between PC A and PC B through switches 1, 2, 3, and 4, and all forwarding tables are correctly populated, with switch 1 pointing to switch 2 to reach B. Now, the link between switches 2 and 3 fails. As a result, switch 4 removes the link to switch 1 from its blocked mode and puts it in forwarding. Traffic from A arrives on switch 1, only to be sent to switch 2. Indeed, nobody told switch 1 that it should use switch 4 to reach B. Naturally, this creates a temporary traffic "black hole." In this particular case, relying on the usual forwarding-table aging time alone is not sufficient. Thanks to the TCN/TC-ACK bits, however, switch 1's forwarding table can age out faster and soon point to the correct switch 1-to-4 link to reach B.

NOTE The rapid STP defined in 802.1w in 1999 introduces a proposal/agreement mechanism between switches, thereby significantly reducing the timer-based dependency. It also discards the information contained in the forwarding table altogether when a topology change occurs. Albeit faster than its 802.1D predecessor, 802.1w was designed with no concern for security. BPDUs are not signed or authenticated, the protocol is stateless, and an 802.1w implementation must be capable of understanding 802.1D BPDUs. Therefore, any attack launched against the 802.1D STP works on switches running 802.1w.

Many vendors have augmented the original 802.1D and 802.1w specs to provide a per-VLAN 802.1D or 802.1w for better flexibility in network design. Cisco's own proprietary

version of 802.1D and 802.1w is called per-VLAN (rapid) spanning-tree plus (PVST+). Other than a Cisco-specific destination MAC address and a Subnetwork Access Protocol (SNAP) frame header, the BPDU payload contains exactly the same information as a regular 802.1D or 802.1w BPDU, as Table 3-2 shows.

Table 3-2 *Cisco PVST+ BPDU in VLAN 10*

Field	Value	Explanation
DMAC	**01 00 0c cc cc cd**	Cisco SSTP BPDU MAC
SMAC	**00 02 fc 90 08 38**	Port MAC
PROTOCOL TYPE IDENTIFIER	**81 00**	802.1Q Ethertype
TAG CONTROL INFO	**00 0a**	COS and VLAN ID (VLAN 10)
LENGTH	**00 32**	
802.2 Logical Link Control HEADER		
DSAP	**Aa**	Indicates SNAP encap
SSAP	**Aa**	
UI	**03**	
SNAP HEADER		
VENDOR ID	**00 00 0c**	Cisco Systems
TYPE	**01 0b**	SSTP
PROTOCOL	**00 00**	
PROTOCOL VERSION	**00**	
BPDU TYPE	**00**	
BPDU FLAGS	**00**	
ROOT ID	**20 00 00 d0 00 66 2c 0a**	
PATH COST	**00 00 00 00**	
BRIDGE ID	**20 00 00 d0 00 66 2c 0a**	Bridge ID in VLAN 10
PORT	**81 41**	
MESSAGE AGE	**00 00**	
MAXIMUM AGE	**14 00**	
ROOT HELLO TIME	**02 00**	
ROOT FORWARD DELAY	**0f 00**	

Table 3-2 *Cisco PVST+ BPDU in VLAN 10 (Continued)*

Field	Value	Explanation
VLAN ID Type Length Value		
PAD	**34**	
TYPE	**00 00**	
LENGTH	**00 02**	
VLAN ID	**00 0a**	VLAN 10

NOTE The actual destination MAC address may vary depending on the flavor of STP you are running. For example, the address reserved by the IEEE is 01:80:C2:00:00:00. Cisco uses a MAC address of its choosing for its per-VLAN rapid spanning-tree implementation, because the standard itself does not define a per-VLAN specification.

Let the Games Begin!

Unfortunately, you are likely to come across LAN hackers that are intimately familiar with STP's inner workings. They also know that little or no attention is paid to STP security. They realize how gullible—for lack of a better term—the protocol actually is. STP attacks moved from the theoretical field to reality fairly recently. Black Hat Europe 2005 proposed a session that discussed various ways to exploit STP[3]. Packet-building libraries, such as libnet[4], have been shipping C-source code to help craft homemade BPDUs for some time now, but putting together an attack tool required some programming skills—a fact that probably deterred most *script kiddies*. It was only a matter of time before someone built a frontend to a libnet-based LAN protocol's packet-building machine. Probably the most successful result of that effort is a tool called *Yersinia.* Example 3-1 shows Yersinia's manual page.

Example 3-1 *Yersinia Manual Page*

```
YERSINIA(8)

NAME
       Yersinia - A FrameWork for layer 2 attacks

SYNOPSIS
       yersinia  [-hVID]  [-l  logfile]  [-c  conffile]  protocol [-M]
  [protocol_options]

DESCRIPTION
       yersinia is a framework for performing layer 2 attacks. The following
protocols have  been implemented in Yersinia current version: Spanning Tree
Protocol (STP), Virtual Trunking Protocol (VTP), Hot Standby Router Protocol
(HSRP), Dynamic  Trunking Protocol (DTP), IEEE 802.1Q, Cisco Discovery Protocol
```

continues

Example 3-1 *Yersinia Manual Page (Continued)*

```
(CDP) and finally, the  Dynamic  Host  Configuration Protocol (DHCP).
       Some of the attacks implemented will cause a DoS in a network, other will
help to perform any other more advanced attack, or both. In addition, some of
them  will be first released to the public since there isn't any public
implementation.
```

The tool basically covers all the most common LAN protocols deployed in today's networks: STP, VLAN Trunk Protocol (VTP), Hot Standby Router Protocol(HSRP), Dynamic Trunking Protocol (DTP), Cisco Discovery Protocol (CDP), DHCP—they are all in there. Even worse, it comes with a GUI! According to Yersinia's home page,[5] it proposes these STP attacks:

- Sending RAW Configuration BPDU
- Sending RAW TCN BPDU
- Denial of Service (DoS) sending RAW Configuration BPDU
- DoS Sending RAW TCN BPDU
- Claiming Root Role
- Claiming Other Role
- Claiming Root Role Dual-Home (MITM)

Basically, Yersinia has everything that anyone interested in messing around with STP would ever need. The GUI is based on the ncurses library (for character-cell terminals, such as VT100). Figure 3-5 shows Yersinia's protocols.

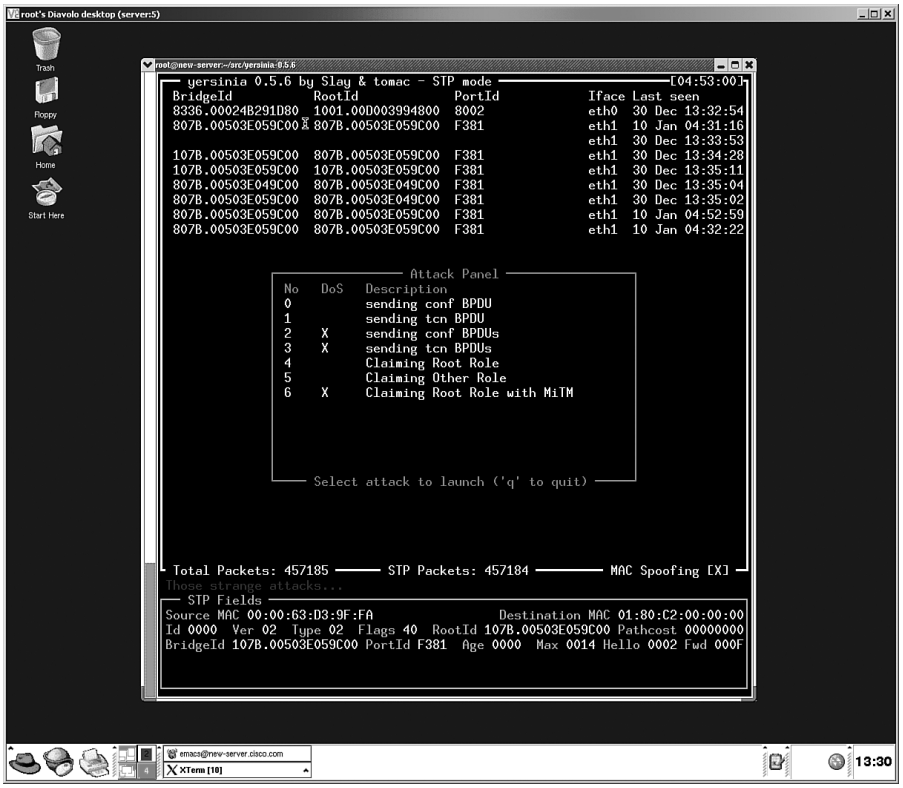
Yersinia continuously listens for STP BPDUs and provides instant decoded information, including current root bridge and timers it is propagating—all this for 802.1D, 802.1w, and Cisco BPDUs. The following sections review the major STP attacks and offer appropriate countermeasures.

Figure 3-5 *Yersinia's Protocols*

Attack 1: Taking Over the Root Bridge

Taking over a root bridge is probably one of the most disruptive attacks. By default, a LAN switch takes any BPDU sent from Yersinia at face value. Keep in mind that STP is trustful, stateless, and does not provide a solid authentication mechanism. The default STP bridge priority is 32768. Once in root attack mode, Yersinia sends a BPDU every 2 sec with the same priority as the current root bridge, but with a slightly numerically lower MAC address, which ensures it a victory in the root-bridge election process. Figure 3-6 shows Yersinia's STP attack screen, followed by a **show** command capture on the LAN switch under attack.

Figure 3-6 *Yersinia's STP Attacks*



Example 3-2 shows the result of the attack on the switch. (The hacker running Yersinia is connected to port F8/1.)

Example 3-2 *Cisco IOS Command to Display Port-Level STP Details*

```
6K-2-S2#show spanning-tree vlan 123 interface f8/1 detail
 Port 897 (FastEthernet8/1) of VLAN0123 is root forwarding
   Port path cost 19, Port priority 240, Port Identifier 240.897.
   Designated root has priority 32891, address 0050.3e04.9c00
   Designated bridge has priority 32891, address 0050.3e04.9c00
   Designated port id is 240.897, designated path cost 0
   Timers: message age 15, forward delay 0, hold 0
   Number of transitions to forwarding state: 2
   Link type is point-to-point by default
   Loop guard is enabled by default on the port
   BPDU: sent 29, received 219
6K-2-S2#
! The previous command show the status of the port for a given VLAN, and
! the number of BPDU received on the port. Here, something abnormal is
```

Example 3-2 *Cisco IOS Command to Display Port-Level STP Details (Continued)*

```
! happening: a root port should typically be sending many more BPDUs than
! it is receiving. The opposite is taking place here, indicating suspicious
! activity.
6K-2-S2#sh spanning-tree bridge address ¦ inc VLAN0123
VLAN0123         0050.3e05.9c00
6K-2-S2#
6K-2-S2#sh spanning-tree vlan 123 root

                                        Root    Hello Max Fwd
Vlan                Root ID             Cost    Time  Age Dly  Root Port
---------------- ------------------- --------- ----- --- ---  ------------
VLAN0123         32891 0050.3e04.9c00        19    2   20  15  Fa8/1
6K-2-S2#
```

Notice this bridge's MAC address versus the MAC generated by Yersinia (0050.3e05.9c00 vs 0050.3e04.9c00). Yersinia wins (04 < 05), and the switch is convinced that the root bridge is located off port 8/1.

Forging Artificially Low Bridge Priorities

It is no problem for an attack tool to generate a BPDU with both the priority and the bridge ID set to 0, as Example 3-3 shows.

Example 3-3 *Cisco IOS Command to Verify Root Bridge Status*

```
6K-2-S2#show spanning-tree vlan 123 root

                                        Root    Hello Max Fwd
Vlan                Root ID             Cost    Time  Age Dly  Root Port
---------------- ------------------- --------- ----- --- ---  ------------
VLAN0123             0 0000.0000.0000        19    2   20  15  Fa8/1
6K-2-S2#
```

Such a BPDU is absolutely impossible to beat, because no switch would ever generate an all-0 bridge ID.

Two other minor variations of the *taking root ownership* theme exist:

- **Root ownership attack: alternative 1**. Another disruptive attack alternative could consist in first taking over the root bridge, and then never setting the TC-ACK bit in BPDUs when receiving a TCN BPDU. The result is a constant premature aging of the entries in the switches' forwarding tables, possibly resulting in unnecessary flooding.

- **Root ownership attack: alternative 2**. For an even more negative effect, a sequence where the attack tool generates a superior BPDU claiming to be the root followed by a retraction of that information seconds later (see Yersinia's "claiming other role" function) could be used. This is guaranteed to cause lots of process churn because of constant state machine transitions, with high CPU utilization as a result and a potential DoS.

Fortunately, the countermeasure to a root takeover attack is simple and straightforward. Two features help thwart a root takeover attack:

- Root guard
- BPDU-guard

Root Guard

The root guard feature ensures that the port on which root guard is enabled is the designated port. Normally, root bridge ports are all designated ports, unless two or more ports of the root bridge are connected. If the bridge receives superior BPDUs on a root guard–enabled port, root guard moves this port to a root-inconsistent state. This root-inconsistent state is effectively equal to a listening state. No traffic is forwarded across this port. In this way, root guard enforces the position of the root bridge. See the first entry in the section, "References," for more details.

BPDU-Guard

The BPDU-guard feature allows network designers to enforce the STP domain borders and keep the active topology predictable. Devices behind ports with BPDU-guard enabled are unable to influence the STP topology. Such devices include hosts running Yersinia, for example. At the reception of a BPDU, BPDU-guard disables the port. BPDU-guard transitions the port into the *errdisable* state, and a message is generated. See the second entry in the section, "References," for more details.

Example 3-4 shows root guard blocking a port receiving a superior BPDU.

Example 3-4 *Root Guard in Action*

```
6K-2-S2# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
6K-2-S2(config)# interface fastethernet 8/1
6K-2-S2(config-if)# spanning-tree rootguard
6K-2-S2(config-if)# ^Z
*Dec 30 18:25:16: %SPANTREE-2-ROOTGUARD_CONFIG_CHANGE: Rootguard enabled on
port FastEthernet8/1 VLAN 123.

Dec 30 18:33:41.677: %SPANTREE-SP-2-ROOTGUARD_BLOCK: Root guard blocking port Fa
stEthernet8/1 on VLAN0123.
6K-2-S2#sh spanning-tree vlan 123 ac
```

Example 3-4 *Root Guard in Action (Continued)*

```
VLAN0123
  Spanning tree enabled protocol rstp
  Root ID    Priority    32891
             Address     0050.3e05.9c00
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32891  (priority 32768 sys-id-ext 123)
             Address     0050.3e05.9c00
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 300

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Fa8/1            Desg BKN*19        240.897  P2p *ROOT_Inc
Fa8/45           Desg FWD 19        128.941  P2p
Gi9/14           Desg FWD 4         128.1038 P2p
Gi9/15           Desg FWD 4         128.1039 Edge P2p

! "Desg" means designated port role; BKN means status blocking;
! FWD means forwarding. Notice the "ROOT Inc" status for port Fa8/1.
```

If the attack stops, or if it was fortuitous, the port swiftly moves back to forwarding. This can take as little as three times the hello interval (6 sec, by default) if only a single superior BPDU was received.

Unless explicitly configured to bridge—which is a rare occurrence—end stations, such as PCs running any sort of operating system (OS), IP phones, printers, and so on, should never generate BPDUs, let alone superior BPDUs. Therefore, BPDU-guard is, and should be, usually preferred to root guard on access ports. BPDU-guard is much less forgiving than root guard: It instructs STP to error-disable a port in case any BPDU arrives on it. After a port is placed in the error-disabled state, there are two ways to recover from the action: either through a manual intervention (do/do not shut down the port) or through an automatic recovery timer whose minimum value is 30 sec. Example 3-5 shows how to configure this using Cisco IOS on a Catalyst 6500. (As usual, consult your switch's documentation for the exact syntax and availability of the feature.)

Example 3-5 *How to Configure BPDU-Guard*

```
6K-2-S2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
6K-2-S2(config)#int f8/1
6K-2-S2(config-if)#spanning-tree bpduguard enable
6K-2-S2(config-if)#exit
6K-2-S2(config)#exit
6K-2-S2#
6K-2-S2(config)#errdisable recovery cause bpduguard
6K-2-S2(config)#errdisable recovery ?
```

continues

Example 3-5 *How to Configure BPDU-Guard (Continued)*

```
  cause     Enable error disable recovery for application
  interval  Error disable recovery timer value

6K-2-S2(config)#errdisable recovery inter
6K-2-S2(config)#errdisable recovery interval ?
  <30-86400>  timer-interval(sec)

6K-2-S2(config)#errdisable recovery interval 30
```

Immediately after a BPDU is received on the port, these messages are printed:

```
Dec 30 18:23:58.685: %LINEPROTO-5-UPDOWN: Line protocol on Interface
  FastEthernet8/1, changed state to down
Dec 30 18:23:58.683: %SPANTREE-SP-2-BLOCK_BPDUGUARD: Received BPDU on port
  FastEthernet8/1 with BPDU Guard enabled. Disabling port.
Dec 30 18:23:58.683: %PM-SP-4-ERR_DISABLE: bpduguard error detected on Fa8/1,
  putting Fa8/1 in err-disable state
```

If this BPDU was the result of an accident, the port is restored 30 sec later:

```
Dec 30 18:24:28.535: %PM-SP-4-ERR_RECOVER: Attempting to recover from bpduguard
  err-disable state on Fa8/1
```

By using the following command, it is possible to globally enable BPDU-guard on all portfast-enabled ports:

```
6K-2-S2(config)#spanning-tree portfast bpduguard ?
  default  Enable bdpu guard by default on all portfast ports
```

Portfast

Portfast is a port-based setting that instructs the port on which it is enabled to bypass the listening and learning phases of STP. The effect is that the port directly moves to forwarding, accepting, and sending traffic. The setting is typically applied to ports where end devices are attached, such as laptops, printers, servers, and so on.

Unlike root guard, BPDU-guard is not limited only to root takeover attempts. Any incoming BPDU disables the port—period. On many Cisco IOS versions, BPDU-guard no longer requires a port to be portfast-enabled.

Attack 2: DoS Using a Flood of Config BPDUs

Attack number 2 in Yersinia (sending conf BPDUs) is extremely potent. With the cursors GUI enabled, Yersinia generated roughly 25,000 BPDUs per second on our test machine (Intel Pentium 4 machine running Linux 2.4–20.8). This seemingly low number is more

than sufficient to bring a Catalyst 6500 Supervisor Engine 720 running 12.2(18)SXF down to its knees, with 99 percent CPU utilization on the switch processor:

```
6K-3-S720#remote command switch show proc cpu ¦ incl second
CPU utilization for five seconds: 99%/86%; one minute: 99%; five minutes: 76%
```

At that point, serious side effects start to happen. HSRP suffered from continuous flapping during the attack:

```
6K-3-S720#
Dec 30 18:59:21.820: %STANDBY-6-STATECHANGE: Vlan448 Group 48 state Standby ->
  Active
6K-3-S720#
```

The attack's purpose is fulfilled: The switch is quickly DoS'd. Unless BPDU-guard is enabled, detecting this attack is not easy. Although it could, as the 802.1w specification suggests,[6] the STP does not complain about handling thousands of incoming BPDUs. It just tries to process as many as it can until its processing power is exhausted. High CPU utilization and an extremely high and quickly increasing count of received BPDUs off a given port indicate a BPDU flooding attack, as Example 3-6 shows.

Example 3-6 *Port Receiving Too Many BPDUs Too Quickly*

```
6K-3-S720#show spanning-tree vlan 123 interface f8/1 detail
 Port 897 (FastEthernet8/1) of VLAN0123 is root forwarding
   Port path cost 19, Port priority 240, Port Identifier 240.897.
   Designated root has priority 0, address 9838.9a38.3cf0
   Designated bridge has priority 52067, address 9838.9a38.3cf0
   Designated port id is 0.0, designated path cost 0
   Timers: message age 20, forward delay 0, hold 0
   Number of transitions to forwarding state: 4
   Link type is point-to-point by default, Peer is STP
   BPDU: sent 1191, received 7227590
```

Frequent transitions of a port from blocking to forwarding in a short interval confirm suspicions (use the Cisco IOS command **logging-event spanning-tree status** under the interface, if available):

```
5w2d: %SPANTREE-SP-6-PORT_STATE: Port Fa5/14 instance 1448 moving from blocking
  to blocking
5w2d: %SPANTREE-SP-6-PORT_STATE: Port Fa5/14 instance 1448 moving from blocking
  to forwarding
```

Three countermeasures exist for this attack. Two are available to most switches, and one has hardware dependencies:

- BPDU-guard
- BPDU filtering
- Layer 2 PDU rate limiter

BPDU-Guard

BPDU-guard was introduced in the previous section. Because it completely prevents BPDUs from entering the switch on the port on which it is enabled, the setting can help fend off this type of attack.

BPDU Filtering

There is actually another method to discard incoming *and* outgoing BPDUs on a given port: BPDU filtering. This feature *silently* discards both incoming and outgoing BPDUs. Although extremely efficient against a brute-force DoS attack, BPDU filtering offers an immense potential to shoot yourself in the foot. Enable this feature on the incorrect port, and any loop condition goes undetected forever, which causes instantaneous network downtime. On the other hand, not sending out BPDUs is actually a good thing when faced with a hacker using Yersinia. Yersinia listens for BPDUs in order to craft its own packets based on information contained in genuine BPDUs. If the tool isn't fed any data to start with, it slightly complicates the hacker's job; I say it only "slightly complicates" because Yersinia is a powerful tool when it comes to exploiting STP: It comes with a prefabricated BPDU ready to be sent on the wire! Because of its danger potential, use BPDU filtering with extreme caution and only after you clearly understand its potential negative effects. Suppose, for example, that a user accidentally connects two ports of the same switch. STP would normally take care of this loop condition. With BPDU filtering enabled, it is not taken care of, and packets loop forever! Only enable it toward end-station ports. It is enabled on a port basis using the **spanning-tree bpdufilter enable** command, as Example 3-7 shows.

Example 3-7 *How to Enable BPDU Filtering on a Port*

```
6K-3-S720(config)#interface f5/14
6K-3-S720(config-if)#spanning-tree bpdufilter enable
6K-3-S720(config-if)#^Z
6K-3-S720#
*Dec 30 19:26:37.066: %SYS-5-CONFIG_I: Configured from console by vty0
 (10.48.82.102)
6K-3-S720#sh spanning-tree vlan 1448 int f5/14 detail ¦ include filter
   Bpdu filter is enabled
6K-3-S720#
```

As soon as either BPDU-guard or BPDU filtering is enabled, the CPU utilization returns to normal.

Layer 2 PDU Rate Limiter

Available only on certain switches, such as the Supervisor Engineer 720 for the Catalyst 6500, a third option to stop the DoS from causing damage exists. It takes the form of a hardware-based Layer 2 PDU rate limiter. It limits the number of Layer 2 PDUs (BPDUs, DTP, Port Aggregation Protocol [PAgP], CDP, VTP frames) destined for the supervisor engine's processor. The feature works only on Catalyst 6500/7600 that are *not* operating in truncated mode. The switch uses truncated mode for traffic between fabric-enabled modules when both fabric-enabled and nonfabric-enabled modules are installed. In this mode, the router sends a truncated version of the traffic (the first 64 bytes of the frame) over the switching fabric. (For more information about the various modes of operation of the Catalyst 6500 switch, see the third entry in the section, "References.") The Layer 2 PDU rate limiter is configured as follows:

```
Router(config)# mls rate-limit layer2 pdu 200 20 → 200 L2 PDUs per second, burst of
20 packets
```

Fine-tuning the rate limiter can be time consuming and error prone, because it is global to the switch and applicable to traffic received across all VLANs for various Layer 2 protocols. However, it can be safely enabled with a fairly high threshold. As a rough guideline, 2000 PDUs per second is a high watermark figure for an enterprise class switch. (The rate limiter prevents only a DoS attack. It does not stop the other attacks described in this chapter [root hostile takeover, and so on].)

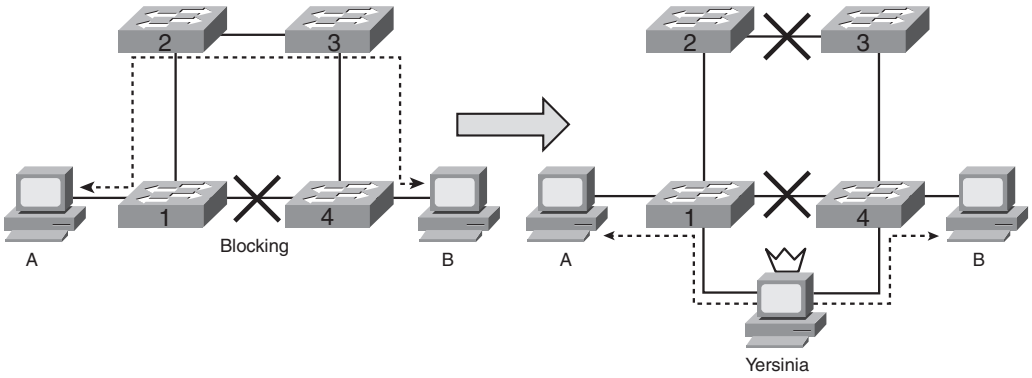
Attack 3: DoS Using a Flood of Config BPDUs

Closely resembling the previous attack, this attack continuously generates TCN BPDUs, forcing the root bridge to acknowledge them. What's more, all bridges down the tree see the TC-ACK bit set and accordingly adjust their forwarding table's timers; this results in a wider impact to the switched network. When the TC bit is set in BPDUs, switches adjust their bridging table's aging timer to *forward_delay* seconds. The protection is the same as before: BPDU-guard or filtering.

Attack 4: Simulating a Dual-Homed Switch

Yersinia can take advantage of computers equipped with two Ethernet cards to masquerade as a dual-homed switch. This capability introduces an interesting traffic-redirection attack, as Figure 3-7 shows.

Figure 3-7 *Simulating a Dual-Homed Switch*



In Figure 3-7, a hacker connects to switches 1 and 4. It then takes root ownership, creating a new topology that forces all traffic to cross it. The intruder could even force switches 1 and 4 to negotiate the creation of a trunk port and intercept traffic for more than one VLAN.

Again, BPDU-guard stands out as the most advantageous solution to deter the attack.

Summary

Conducting STP attacks is now within the reach of a wide population, thanks to the availability of point-and-shoot attacks tools, such as Yersinia. Elaborated two decades ago, the protocol didn't include security as a critical component of its design. This lack of consideration for security attracted hackers' attention all over the world, as recently shown at Black Hat Europe 2005, for example.[5] The only vaguely reassuring fact is that, to perform an attack, a miscreant needs direct connectivity with the LAN infrastructure. Nonetheless, STP attacks are extremely disruptive because the protocol lays the foundation for most modern LANs. Attacks can cause traffic black holes, DoS attacks, excessive flooding, redirection of traffic to the hacker's computer, and more. Fortunately, simple features widely available on a range of switches, such as BPDU-guard, provide effective measures against spanning-tree–based exploits.

References

[1] http://www.cisco.com/en/US/tech/tk389/tk621/technologies_tech_note09186a00800ae96b.shtml.

[2] http://www.cisco.com/warp/public/473/65.html.

[3] http://www.cisco.com/en/US/products/hw/switches/ps708/products_configuration_guide_chapter09186a0080160a5e.html.

[4] http://www.cisco.com/en/US/tech/tk389/tk621/technologies_tech_note09186a0080094797.shtml.

[5] https://www.blackhat.com/presentations/bh-europe-05/BH_EU_05-Berrueta_Andres/BH_EU_05_Berrueta_Andres.pdf.