# Identifying Embedded Web Servers

Ensuring that hardware devices don't create an enterprise backdoor

## ABSTRACT

Today, everything from television sets to photocopiers comes with an IP address and an embedded web server (EWS) for device administration. Need to record a show? Start the DVR with a mobile app. Want a digital copy of a previously photocopied document? No problem. While embedded web servers are now as common as digital displays in hardware devices, sadly, security is not. What if that same convenience exposed photocopied documents online or allowed outsiders to record your telephone conversations? A frightening thought indeed. In this paper we will debate the challenges faced when identifying embedded web servers and highlight the risks that they can pose if not properly secured.

## Overview

In pursuing research on risks associated with embedded web servers (EWSs), we sought to find a scalable means of scanning millions of IP addresses in an effort to identify hardware devices serving up web based administrative consoles that may pose security threats when not properly secured. In the course of our research, it became clear that traditional scanning techniques were not always ideal, as they tended to focus on traditional web application servers and proved to be inaccurate when used with EWSs. In this paper we will debate the merits of various approaches to identifying EWSs on a large-scale network. At Blackhat USA 2011, we will expand on the content of this paper to walk through the findings of our research to illustrate a variety of real world threats, whereby EWSs in popular hardware devices are exposing sensitive data/functionality. We will also release brEWS, a web based scanner designed to easily identify the presence of EWSs on a network.

## Background

Hardware devices commonly provide an administrative console to permit users to change default settings as needed. Such functionality may include changing network settings, authentication credentials or enabling/disabling functionality. Whereas such consoles once required native applications to be run on a connected computer, today web based administrative consoles are common. Such a change makes sense for ease of use. Unfortunately, it also exposes the same administrative console to anyone with a web browser, should access not be properly controlled.

Appliances with EWSs typically ship with a majority of functionality enabled, while the administrative web application is protected only using default credentials shared among all devices. For an attacker, once an insecure device is identified, access is as trivial as downloading the user manual. A more secure approach for vendors involves access credentials that are unique to the device (i.e. MAC address) or preferably, not enabling the device at all, until the administrator chooses unique credentials. Unfortunately, this is rarely done; especially for consumer appliances or those targeted at small/medium sized businesses. When vulnerabilities in hardware devices are made public, it tends to be less likely that the EWS will be included in enterprise patch management programs that will ensure an upgrade of the firmware to the latest patch level. This is even less likely for consumer devices that do not have automated patching mechanism. Security risks in EWSs are further compounded by the fact that network misconfigurations can expose the devices to the Internet; greatly expanding exposure to would be attackers.

There is no universally accepted definition for exactly what constitutes an embedded web server. For our purposes, we require an EWS to have the following characteristics:

1. Web server installed on the hardware during the manufacturing process (not an optional component)
2. Not designed for high performance
3. Limited functionality
4. Serves as an administrative interface to the host hardware

## Scanning

Identifying EWSs on a sizable network is not as straightforward as one might expect. While identifying the existence of web servers is a challenge that any scanner can tackle, most tools and techniques focus on identifying web application servers (Apache, IIS, etc.) and not EWSs. Therefore, during a security audit, an EWS will often be identified, but lumped in with all other web servers. As such, it will be subjected to standard web application tests such as cross-site scripting (XSS) and SQL injection flaws. However, basic tests such as failing to be password protected and exposing unneeded and potentially dangerous functionality may be completely overlooked. For example, in a past blog post, we discussed how most HP scanners by default ship with Webscan functionality, which is not password protected[1]. Webscan appears to have been a standard addition to all HP Officejet/Photosmart scanners for several years now. The functionality permits a user to remotely operate the scanner through the admin console via a web browser. This is a somewhat curious feature; given the physical nature of a scanner and the fact that one must be in contact with the scanner to place a document on the scanning surface. Nonetheless, it exposes potentially millions of scanners to internal and external attackers, allowing them to operate the appliance remotely in the hopes of capturing an image of anything left on the device.

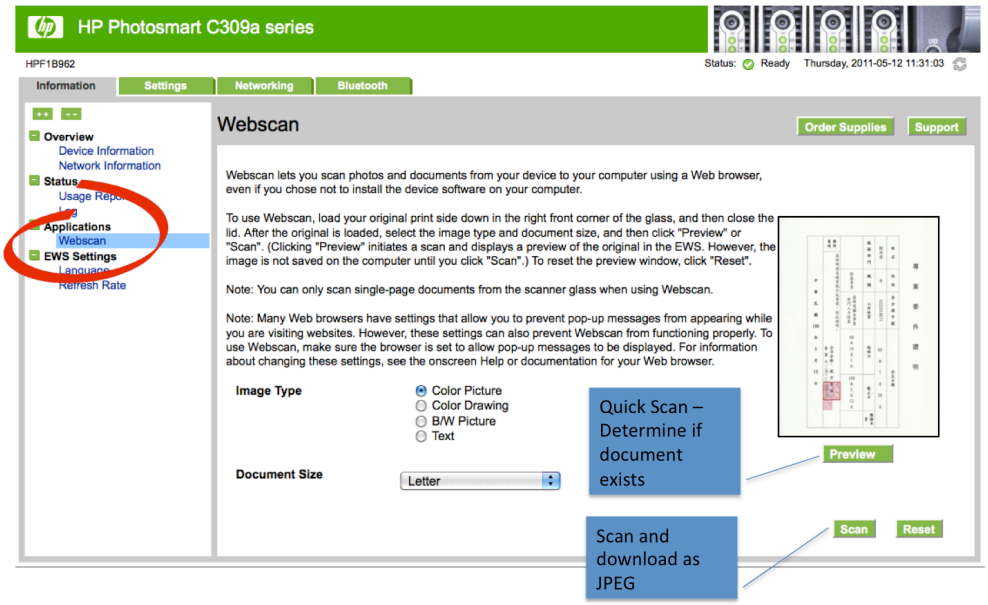


**Figure 1 - HP Webscan functionality**

[1] http://research.zscaler.com/2010/08/corporate-espionage-for-dummies-hp.html

## Network Scanners

Traditional network security scanners tend to have fingerprinting databases that focus on web application servers. As such, the results returned may identify the existence of a web server listening on port 80/443, but they are very likely to misidentify the actual device hosting the EWS. It can therefore be a challenge to accurately identify all EWSs on a network. For example, take a look at the following Nmap results for sample EWSs:

**Canon imageRUNNER C2880 Photocopier**

**Nmap Results**

```
$ sudo nmap -O 131.96.246.162
[snip]
Aggressive OS guesses: Apple AirPort Express WAP v6.3 (92%), AirSpan ProST WiMAX access point
(91%), m0n0wall FreeBSD-based embedded firewall version 1.22 - 1.23b1 (89%), Canon imageRUNNER
C5185 printer (89%), SonicWALL SonicOS Enhanced 5.2.0.1-21o (88%), FreeBSD 6.2-RELEASE (88%),
VxWorks: Apple AirPort Extreme v5.7 or AirPort Express v6.3; Canon imageRUNNER printer (5055,
C3045, C3380, or C5185); Kyocera FS-4020DN printer; or Xerox Phaser 8860MFP printer (87%), IBM
DCS9900 NAS device (87%), Nokia IP650 firewall (IPSO 4.0 and CheckPoint Firewall-1/VPN-1
software) (85%), HP LaserJet P2055dn printer (85%)
No exact OS matches for host (test conditions non-ideal).
```

**Server Headers**

```
HTTP/1.1 200 OK
Date: SUN, 16 JUL 2011 19:13:57 GMT
Server: CANON HTTP Server Ver2.21
Content-Type: text/html
Transfer-Encoding: chunked
```

**HP LaserJet 2300 Series Printer**

**Nmap Results**

```
$ sudo nmap -O 128.193.50.3
[snip]
Device type: printer|WAP|firewall
Running (JUST GUESSING) : HP embedded (96%), Linksys embedded (89%), Netgear embedded (89%),
Ricoh embedded (86%), Nokia IPSO 4.X (86%)
Aggressive OS guesses: HP LaserJet 4300 JetDirect FW R.25.57 (96%), Linksys WRT54G or WRT54G2, or
Netgear WGR614 or WPN824v2 wireless broadband router (89%), HP LaserJet 2200, 3030, or 4300
printer (86%), Ricoh Aficio AP400N printer (86%), Nokia firewall (IPSO 4.1Build19) (86%), HP
LaserJet 4200 printer (86%)
No exact OS matches for host (test conditions non-ideal).
```

**Server Headers**

```
HTTP/1.1 301 Moved Permanently
Location: hp/device/this.LCDispatcher
MIME-Version: 1.0
Server: HP-ChaiServer/3.0
Content-Length: 0
```

As can be seen, the Nmap scan did not definitively identify either the photocopier or printer in the sample scans. In the first scan, a variety of Canon copiers were included among the results with a confidence rating ranging between 87-89%, but they were mixed in with a host of other hardware devices ranging from wireless access points (WAPs) to network attached storage devices. Likewise, for the printer scan, HP LaserJet printers were included, but so too were other printers, WAPs and firewalls.

## Search Engine Hacking

Google Hacking (or Bing, Yahoo!, etc.), has been leveraged by attackers for some time as a simple means of identifying potentially vulnerable web servers. Enterprise security personnel may also include Google

Hacking during security tests to ensure that vulnerable servers have not been indexed within search results. While Google Hacking has the benefit of simplicity, it is not ideal for identifying EWSs for a variety of reasons:

1. Google is clearly suppressing search results for embedded web servers. Alternate search engines such as Shodan (see below) will reveal many more EWSs, despite the fact that the servers are publicly accessible and can be indexed by Google's bots.
2. Google may block access from a source IP address if automated queries are detected, resulting in a Google 'sorry page', which can negatively impact all users behind a NATed IP address.
3. Multiple queries are needed to cover content localization used by the administrative interfaces of various EWSs. Consider the example in Figure 2. The English language interface highlights unique search terms that can be used to identify this particular Canon photocopier, yet such queries will be of no value in identifying the same photocopier displaying the interface in an alternate language.
4. Public search engines are of no value when trying to identify EWSs on a private network.


Figure 2 - Canon photocopier admin console displaying localized languages

## Header Scanning

Header scanning, while not a perfect technique, offers a number of advantages when the goal requires scanning a large block of IP addresses.

1. Hardware vendors often insert unique strings for the Server header.
2. Even when an OEM web server is leveraged in a hardware device, it tends to be a specialized EWS (e.g. RomPager, Virata-Emweb, Boa, etc.). Therefore, despite the fact that the type of appliance may not be revealed, it is clear that an EWS has been identified.
3. EWS Server headers are much less likely to be spoofed, as needed functionality to do so is not exposed by the device.
4. Retrieving such information is highly scalable, as it requires only a HEAD request.
5. Scanning automation can be done via basic scripting languages

Shodan (http://shodanhq.com/) is an example of a commercial database, which permits querying web server headers. It can be a valuable tool for identifying publicly accessible EWSs, but cannot of course be used for identifying EWSs on a private LAN. Shodan uses a freemium model with the first ten results being publicly accessible to guests, fifty results are available to registered users and thereafter up to 10,000 results can be accessed for between $12.50-$25, depending upon how many 'credits' are purchased. Shodan is a powerful database. Unfortunately, it offers the greatest value for those looking for a randomly vulnerable server, as opposed to resources that they control and want to audit. As can be

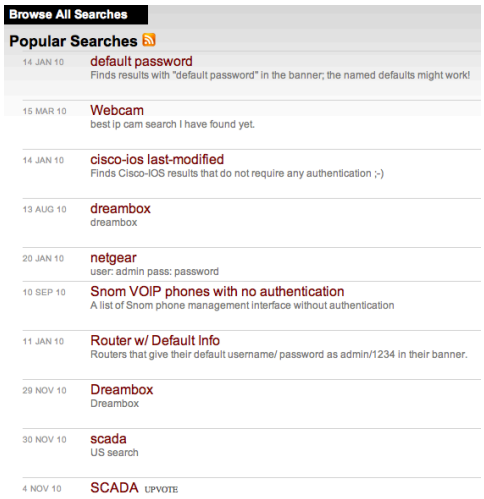


**Figure 3 - Popular Shodan Searches**

seen in Figure 3, searches tend to focus on either identifying EWSs (webcams, networking hardware, SCADA, etc.) or sites with default passwords/no authentication.

Most EWSs lie behind corporate firewalls. As such, for enterprises looking to take advantage of server header scanning to identify EWSs, it is necessary to effectively compile the equivalent of an enterprise specific Shodan database. Fortunately, this is not difficult to do. Simply looping through HTTP HEAD requests for all IP addresses on the LAN will reveal EWSs, assuming that the IPs are accessible (see Appendix for a simple Perl script to scan for server headers).

EWS scanners can be made more granular by further identifying static content served up by the page, such as the existence of a static URL path, MD5 hash of a standard image or a regex against page content. Given that the web based administrative consoles used by hardware vendors tend to change infrequently and be reused across many devices, this is a much more viable approach for EWSs than with traditional web applications. This is the approach taken by web fingerprinting scanners such as Andrew Horton's excellent WhatWeb scanner[2], which seeks to identify not only that a web server is present, but also which web application may be running on the server. The brEWS (Basic Request Embedded Web Server) scanner to be released by Zscaler at Blackhat USA 2011, takes a similar approach.

[2] http://www.morningstarsecurity.com/research/whatweb

## Threats

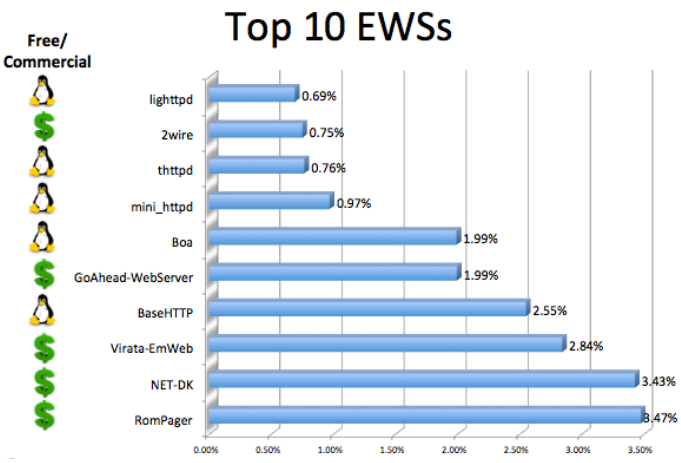


**Figure 4 - Most common EWSs encountered during testing**

Insecure EWSs can expose enterprises to significant security threats, which should not be overlooked. EWSs should be included along with traditional web servers in any enterprise security audit, to ensure that they are appropriately locked down and do not expose sensitive data or functionality that could negatively impact overall security. Security teams often fail to properly analyze EWSs as they incorrectly assume that such devices do not present a risk, even if not properly hardened. At Blackhat USA 2011, we will discuss a variety of identified threats such as scanners/photocopiers being used to access sensitive data, VoIP systems permitting eavesdropping, network devices that can be remotely reconfigured, etc. EWSs are also often considered to be a purely internal threat. Results from our own Internet scans or services such as Shodan or Google hacks, show that misconfigured networks regularly expose EWSs to the Internet. Even when this is not the case, exposing sensitive data or functionality to employees and contractors can be equally as serious. You wouldn't leave sensitive HR documentation lying around or permit anyone to listen in on phone calls, and yet insecure EWSs can expose an enterprise to such threats. EWSs, while offering limited functionality, pose a unique threat due to the following:

1. Default passwords are often not changed
2. Firmware on hardware devices often not considered during the patch management process
3. Hardware devices are often not considered in enterprise security audits

## Conclusion

EWSs present a threat to both enterprises and consumers that is often overlooked. We tend to consider hardware devices to be black boxes with fixed capabilities, but that is not the case when an EWS is included. The EWS by design permits remote parties to access data on the device and change functionality. When EWSs are not properly hardened during deployment, unauthorized parties (internal and external) may find a wide open backdoor. At Blackhat USA 2011 we will walk through a variety identified threats posed by popular EWSs, from photocopiers to VoIP systems and also provide access to brEWS, a tool designed to both help identify EWSs and share valuable fingerprinting data among researchers.

# Appendix

## webheaderscan.pl

```
#!/usr/bin/perl
# Author: Michael Sutton
# Company: Zscaler
# Purpose: Simple web server header scanner
# Sept. 7, 2010
########################

use LWP::UserAgent;
use Net::IP;
use Parallel::ForkManager;

if ($#ARGV != 0) {
        print "usage: perl webheaderscan.pl 1.1.1.1-2.2.2.2\n";
        exit;
}

# Threads
my $MAX_IPS = 100;
my $pm_ip = new Parallel::ForkManager($MAX_IPS);

my $ua = new LWP::UserAgent;
# $ua->agent('Mozilla/5.5 (compatible; MSIE 5.5; Windows NT 5.1)');
$ua->timeout(5);

# Scan class C
my $ip=new Net::IP($ARGV[0]) or die (Net::IP::Error());


while ($ip++) {
        $pm_ip->start and next;  # Fork child thread
        $site = $ip->ip() || warn ($ip->error());;
        my $url = "http://$site";
        my $request = HTTP::Request->new('HEAD');
        $request->url($url);
        my $response = $ua->request($request);
        my $server = $response->header('Server');
        my $headers = $response->headers_as_string;
        my $code = $response->code;
        if ($response->is_success( )) {
                print "$site\n$code\n$headers<<<--->>>\n";
        } else {
                print "Error: " . $response->status_line . "\n<<<--->>>\n";
        }
        $pm_ip->finish;          # Terminate child
}
$pm_ip->wait_all_children;  # Wait for children to finish
exit(0);
```