Nessus Vulnerability Assessment Report

This report summarizes the findings from the Nessus vulnerability scan. It focuses on High and Medium severity vulnerabilities discovered on the target system. These vulnerabilities, if exploited, may pose a significant risk to the security of the system. Each vulnerability entry includes details of the issue, risk factor, and recommended remediation steps.

Host: 127.0.0.1
Port/Service: 8834/www
Severity: Medium
Plugin: SSL Certificate Cannot Be Trusted (ID: 51192)
Risk Factor: Medium
Synopsis: The SSL certificate for this service cannot be trusted.
Description: The server's X.509 certificate cannot be trusted. This situation can occur in three different ways, in which the chain of trust can be broken, as stated below : - First, the top of the certificate chain sent by the server might not be descended from a known public certificate authority. This can occur either when the top of the chain is an unrecognized, self-signed certificate, or when intermediate certificates are missing that would connect the top of the certificate chain to a known public certificate authority. - Second, the certificate chain may contain a certificate that is not valid at the time of the scan. This can occur either when the scan occurs before one of the certificate's 'notBefore' dates, or after one of the certificate's 'notAfter' dates. - Third, the certificate chain may contain a signature that either didn't match the certificate's information or could not be verified. Bad signatures can be fixed by getting the certificate with the bad signature to be re-signed by its issuer. Signatures that could not be verified are the result of the certificate's issuer using a signing algorithm that Nessus either does not support or does not recognize. If the remote host is a public host in production, any break in the chain makes it more difficult for users to verify the authenticity and identity of the web server. This could make it easier to carry out man-in-the-middle attacks against the remote host.
Remediation: Purchase or generate a proper SSL certificate for this service.

Host: 127.0.0.1
Port/Service: 0/general
Severity: Medium
Plugin: Ruby WEBrick < 1.8.2 HTTP Request Smuggling (ID: 240854)
Risk Factor: Medium
Synopsis: The remote host has an application installed that is affected by an HTTP request smuggling vulnerability
Description: The version of the WEBrick Ruby library installed on the remote host is prior to 1.8.2. It is, therefore, affected by an HTTP request smuggling vulnerability in the read_header. This vulnerability allows remote attackers to smuggle arbitrary HTTP requests on affected installations of Ruby WEBrick. This issue is exploitable when the product is deployed behind an HTTP proxy that fulfills specific conditions. The specific flaw exists within the read_headers method. The issue results from the inconsistent parsing of terminators of HTTP headers. An attacker can leverage this vulnerability to smuggle arbitrary HTTP requests. Note that Nessus has not tested for this issue but has instead relied only on the application's self-reported version number.
Remediation: Upgrade to WEBrick version 1.8.2 or later.

Host: 127.0.0.1
Port/Service: 0/general
Severity: Medium
Plugin: Ruby REXML < 3.3.6 DoS vulnerability (ID: 242630)
Risk Factor: High
Synopsis: The remote host has an application installed that is affected by a DoS vulnerability.

Description: The version of the REXML Ruby library installed on the remote host is prior to 3.3.6. It is, therefore, affected by a DoS vulnerability. The vulnerability lies when it parses an XML that has many deep elements that have same local name attributes. If you need to parse untrusted XMLs with tree parser API like REXML::Document.new, you may be impacted to this vulnerability. If you use other parser APIs such as stream parser API and SAX2 parser API, this vulnerability is not affected. Note that Nessus has not tested for this issue but has instead relied only on the application's self-reported version number.
Remediation: Upgrade to REXML version 3.3.6 or later.

Host: 127.0.0.1
Port/Service: 0/general
Severity: High
Plugin: c-ares 1.32.3 < 1.34.5 Use After Free (macOS) (ID: 234803)
Risk Factor: High
Synopsis: The remote host is missing a security update.
Description: The version of c-ares installed on the remote host is affected by a use after free vulnerability. c-ares is an asynchronous resolver library. From 1.32.3 through 1.34.4, there is a use-after-free in read_answers() when process_answer() may re-enqueue a query either due to a DNS Cookie Failure or when the upstream server does not properly support EDNS, or possibly on TCP queries if the remote closed the connection immediately after a response. If there was an issue trying to put that new transaction on the wire, it would close the connection handle, but read_answers() was still expecting the connection handle to be available to possibly dequeue other responses. In theory a remote attacker might be able to trigger this by flooding the target with ICMP UNREACHABLE packets if they also control the upstream nameserver and can return a result with one of those conditions, this has been untested. Otherwise only a local attacker might be able to change system behavior to make send()/write() return a failure condition. This vulnerability is fixed in 1.34.5. Note that Nessus has not tested for this issue but has instead relied only on the application's self-reported version number.
Remediation: Upgrade to c-ares version 1.34.5 or later.

Host: 127.0.0.1
Port/Service: 8834/www
Severity: Medium
Plugin: SSL Certificate Cannot Be Trusted (ID: 51192)
Risk Factor: Medium
Synopsis: The SSL certificate for this service cannot be trusted.
Description: The server's X.509 certificate cannot be trusted. This situation can occur in three different ways, in which the chain of trust can be broken, as stated below : - First, the top of the certificate chain sent by the server might not be descended from a known public certificate authority. This can occur either when the top of the chain is an unrecognized, self-signed certificate, or when intermediate certificates are missing that would connect the top of the certificate chain to a known public certificate authority. - Second, the certificate chain may contain a certificate that is not valid at the time of the scan. This can occur either when the scan occurs before one of the certificate's 'notBefore' dates, or after one of the certificate's 'notAfter' dates. - Third, the certificate chain may contain a signature that either didn't match the certificate's information or could not be verified. Bad signatures can be fixed by getting the certificate with the bad signature to be re-signed by its issuer. Signatures that could not be verified are the result of the certificate's issuer using a signing algorithm that Nessus either does not support or does not recognize. If the remote host is a public host in production, any break in the chain makes it more difficult for users to verify the authenticity and identity of the web server. This could make it easier to carry out man-in-the-middle attacks against the remote host.
Remediation: Purchase or generate a proper SSL certificate for this service.

Host: 127.0.0.1
Port/Service: 0/general

Severity: Medium
Plugin: Ruby WEBrick < 1.8.2 HTTP Request Smuggling (ID: 240854)
Risk Factor: Medium
Synopsis: The remote host has an application installed that is affected by an HTTP request smuggling vulnerability
Description: The version of the WEBrick Ruby library installed on the remote host is prior to 1.8.2. It is, therefore, affected by an HTTP request smuggling vulnerability in the read_header. This vulnerability allows remote attackers to smuggle arbitrary HTTP requests on affected installations of Ruby WEBrick. This issue is exploitable when the product is deployed behind an HTTP proxy that fulfills specific conditions. The specific flaw exists within the read_headers method. The issue results from the inconsistent parsing of terminators of HTTP headers. An attacker can leverage this vulnerability to smuggle arbitrary HTTP requests. Note that Nessus has not tested for this issue but has instead relied only on the application's self-reported version number.
Remediation: Upgrade to WEBrick version 1.8.2 or later.

Host: 127.0.0.1
Port/Service: 0/general
Severity: Medium
Plugin: Ruby REXML < 3.3.6 DoS vulnerability (ID: 242630)
Risk Factor: High
Synopsis: The remote host has an application installed that is affected by a DoS vulnerability.
Description: The version of the REXML Ruby library installed on the remote host is prior to 3.3.6. It is, therefore, affected by a DoS vulnerability. The vulnerability lies when it parses an XML that has many deep elements that have same local name attributes. If you need to parse untrusted XMLs with tree parser API like REXML::Document.new, you may be impacted to this vulnerability. If you use other parser APIs such as stream parser API and SAX2 parser API, this vulnerability is not affected. Note that Nessus has not tested for this issue but has instead relied only on the application's self-reported version number.
Remediation: Upgrade to REXML version 3.3.6 or later.

Host: 127.0.0.1
Port/Service: 0/general
Severity: High
Plugin: c-ares 1.32.3 < 1.34.5 Use After Free (macOS) (ID: 234803)
Risk Factor: High
Synopsis: The remote host is missing a security update.
Description: The version of c-ares installed on the remote host is affected by a use after free vulnerability. c-ares is an asynchronous resolver library. From 1.32.3 through 1.34.4, there is a use-after-free in read_answers() when process_answer() may re-enqueue a query either due to a DNS Cookie Failure or when the upstream server does not properly support EDNS, or possibly on TCP queries if the remote closed the connection immediately after a response. If there was an issue trying to put that new transaction on the wire, it would close the connection handle, but read_answers() was still expecting the connection handle to be available to possibly dequeue other responses. In theory a remote attacker might be able to trigger this by flooding the target with ICMP UNREACHABLE packets if they also control the upstream nameserver and can return a result with one of those conditions, this has been untested. Otherwise only a local attacker might be able to change system behavior to make send()/write() return a failure condition. This vulnerability is fixed in 1.34.5. Note that Nessus has not tested for this issue but has instead relied only on the application's self-reported version number.
Remediation: Upgrade to c-ares version 1.34.5 or later.