Rocky Enterprise Linux 9.2 Manual Pages on command 'perlsecpolicy.1'

$ man perlsecpolicy.1

PERLSECPOLICY(1) Perl Programmers Reference Guide PERLSECPOLICY(1)

NAME

perlsecpolicy - Perl security report handling policy

DESCRIPTION

The Perl project takes security issues seriously.

The responsibility for handling security reports in a timely and effective manner has been delegated to a security team composed of a subset of the Perl core developers.

This document describes how the Perl security team operates and how the team evaluates new security reports.

REPORTING SECURITY ISSUES IN PERL

If you believe you have found a security vulnerability in the Perl interpreter or modules maintained in the core Perl codebase, email the details to perl-security@perl.org <mailto:perl-security@perl.org>. This address is a closed membership mailing list monitored by the Perl security team.

You should receive an initial response to your report within 72 hours. If you do not receive a response in that time, please contact the Perl Steering Council <mailto:steering-council@perl.org>.

When members of the security team reply to your messages, they will generally include the perl-security@perl.org address in the "To" or "CC" fields of the response. This allows all of the security team to follow the discussion and chime in as needed. Use the "Reply-all" functionality of your email client when you send subsequent responses so that the entire security team receives the message.

The security team will evaluate your report and make an initial determination of whether it is likely to fit the scope of issues the team handles. General guidelines about how this is determined are detailed in the "WHAT ARE SECURITY ISSUES" section.

If your report meets the team's criteria, an issue will be opened in the team's private issue tracker and you will be provided the issue's ID number. Issue identifiers have the form perl-security#NNN. Include this identifier with any subsequent messages you send.

The security team will send periodic updates about the status of your issue and guide you through any further action that is required to complete the vulnerability remediation process. The stages vulnerabilities typically go through are explained in the "HOW WE DEAL WITH SECURITY ISSUES" section.

WHAT ARE SECURITY ISSUES

A vulnerability is a behavior of a software system that compromises the system's expected confidentiality, integrity or availability protections.

A security issue is a bug in one or more specific components of a software system that creates a vulnerability.

Software written in the Perl programming language is typically composed of many layers of software written by many different groups. It can be very complicated to determine which specific layer of a complex real-world application was responsible for preventing a vulnerable behavior, but this is an essential part of fixing the vulnerability.

Software covered by the Perl security team

The Perl security team handles security issues in:

? The Perl interpreter

? The Perl modules shipped with the interpreter that are developed in the core Perl repository

? The command line tools shipped with the interpreter that are developed in the core Perl repository

Files under the cpan/ directory in Perl's repository and release tarballs are developed and maintained independently. The Perl security team does not handle security issues for these modules.

Bugs that may qualify as security issues in Perl

Perl is designed to be a fast and flexible general purpose programming language. The Perl interpreter and Perl modules make writing safe and secure applications easy, but they do have limitations.

As a general rule, a bug in Perl needs to meet all of the following criteria to be considered a security issue:

? The vulnerable behavior is not mentioned in Perl's documentation or public issue tracker.

? The vulnerable behavior is not implied by an expected behavior.

? The vulnerable behavior is not a generally accepted limitation of the implementation.

? The vulnerable behavior is likely to be exposed to attack in otherwise secure applications written in Perl.

? The vulnerable behavior provides a specific tangible benefit to an attacker that triggers the behavior.

Bugs that do not qualify as security issues in Perl

There are certain categories of bugs that are frequently reported to the security team that do not meet the criteria listed above.

The following is a list of commonly reported bugs that are not handled as security issues.

Feeding untrusted code to the interpreter

The Perl parser is not designed to evaluate untrusted code. If your application requires the evaluation of untrusted code, it should rely on an operating system level sandbox for its security.

Stack overflows due to excessive recursion

Excessive recursion is often caused by code that does not enforce limits on inputs. The Perl interpreter assumes limits on recursion will be enforced by the application.

Out of memory errors

Common Perl constructs such as "pack", the "x" operator, and regular expressions accept numeric quantifiers that control how much memory will be allocated to store intermediate values or results. If you allow an attacker to supply these quantifiers and consume all available memory, the Perl interpreter will not prevent it.

Escape from a Safe compartment

Opcode restrictions and Safe compartments are not supported as security mechanisms. The Perl parser is not designed to evaluate untrusted code.

Use of the "p" and "P" pack templates

These templates are unsafe by design.

Stack not reference-counted issues

These bugs typically present as use-after-free errors or as assertion failures on the type of a "SV". Stack not reference-counted crashes usually occur because code is both modifying a reference or glob and using the values referenced by that glob or reference.

This type of bug is a long standing issue with the Perl interpreter that seldom occurs in normal code. Examples of this type of bug generally assume that attacker-supplied code will be evaluated by the Perl interpreter.

Thawing attacker-supplied data with Storable

Storable is designed to be a very fast serialization format. It is not designed to be safe for deserializing untrusted inputs.

Using attacker supplied SDBM_File databases

The SDBM_File module is not intended for use with untrusted SDBM databases.

Badly encoded UTF-8 flagged scalars

This type of bug occurs when the ":utf8" PerlIO layer is used to read badly encoded data, or other mechanisms are used to directly manipulate the UTF-8 flag on an SV.

A badly encoded UTF-8 flagged SV is not a valid SV. Code that creates SV's in this fashion is corrupting Perl's internal state.

Issues that exist only in blead, or in a release candidate

The blead branch and Perl release candidates do not receive security support. Security defects that are present only in pre-release versions of Perl are handled through the normal bug reporting and resolution process.

CPAN modules or other Perl project resources

The Perl security team is focused on the Perl interpreter and modules maintained in the core Perl codebase. The team has no special access to fix CPAN modules, applications written in Perl, Perl project websites, Perl mailing lists or the Perl IRC servers.

Emulated POSIX behaviors on Windows systems

The Perl interpreter attempts to emulate "fork", "system", "exec" and other POSIX behaviors on Windows systems. This emulation has many quirks that are extensively documented in Perl's public issue tracker. Changing these behaviors would cause significant disruption for existing users on Windows.

Bugs that require special categorization

Some bugs in the Perl interpreter occur in areas of the codebase that are both security sensitive and prone to failure during normal usage.

Regular expressions

Untrusted regular expressions are generally safe to compile and match against with several caveats. The following behaviors of Perl's regular expression engine are the developer's responsibility to constrain.

The evaluation of untrusted regular expressions while "use re 'eval';" is in effect is never safe.

Regular expressions are not guaranteed to compile or evaluate in any specific finite time frame.

Regular expressions may consume all available system memory when they are compiled or evaluated.

Regular expressions may cause excessive recursion that halts the perl interpreter.

As a general rule, do not expect Perl's regular expression engine to be resistant to denial of service attacks.

DB_File, ODBM_File, or GDBM_File databases

These modules rely on external libraries to interact with database files.

Bugs caused by reading and writing these file formats are generally caused by the underlying library implementation and are not security issues in Perl.

Bugs where Perl mishandles unexpected valid return values from the underlying libraries may qualify as security issues in Perl.

Algorithmic complexity attacks

The perl interpreter is reasonably robust to algorithmic complexity attacks. It is not immune to them.

Algorithmic complexity bugs that depend on the interpreter processing extremely large amounts of attacker supplied data are not generally handled as security issues.

See "Algorithmic Complexity Attacks" in perlsec for additional information.

HOW WE DEAL WITH SECURITY ISSUES

The Perl security team follows responsible disclosure practices. Security issues are kept secret until a fix is readily available for most users. This minimizes inherent risks users face from vulnerabilities in Perl.

Hiding problems from the users temporarily is a necessary trade-off to keep them safe.

Hiding problems from users permanently is not the goal.

When you report a security issue privately to the perl-security@perl.org <mailto:perl-security@perl.org> contact address, we normally expect you to follow responsible disclosure practices in the handling of the report. If you are unable or unwilling to keep the issue secret until a fix is available to users you should state this clearly in the initial report.

The security team's vulnerability remediation workflow is intended to be as open and transparent as possible about the state of your security report.

Perl's vulnerability remediation workflow

Initial contact

New vulnerability reports will receive an initial reply within 72 hours from the time they arrive at the security team's mailing list. If you do not receive any response in that time, contact the Perl Steering Council <mailto:steering-council@perl.org>.

The initial response sent by the security team will confirm your message was received and provide an estimated time frame for the security team's triage analysis.

Initial triage

The security team will evaluate the report and determine whether or not it is likely to meet the criteria for handling as a security issue.

The security team aims to complete the initial report triage within two weeks' time. Complex issues that require significant discussion or research may take longer.

If the security report cannot be reproduced or does not meet the team's criteria for handling as a security issue, you will be notified by email and given an opportunity to respond.

Issue ID assignment

Security reports that pass initial triage analysis are turned into issues in the security team's private issue tracker. When a report progresses to this point you will be provided the issue ID for future reference. These identifiers have the format perl-security#NNN or Perl/perl-security#NNN.

The assignment of an issue ID does not confirm that a security report represents a vulnerability in Perl. Many reports require further analysis to reach that determination.

Issues in the security team's private tracker are used to collect details about the problem and track progress towards a resolution. These notes and other details are not made public when the issue is resolved. Keeping the issue notes private allows the security team to freely discuss attack methods, attack tools, and other related private issues.

Development of patches

Members of the security team will inspect the report and related code in detail to produce fixes for supported versions of Perl.

If the team discovers that the reported issue does not meet the team's criteria at this stage, you will be notified by email and given an opportunity to respond before the issue is closed.

The team may discuss potential fixes with you or provide you with patches for testing purposes during this time frame. No information should be shared publicly at this stage.

CVE ID assignment

Once an issue is fully confirmed and a potential fix has been found, the security team will request a CVE identifier for the issue to use in public announcements.

Details like the range of vulnerable Perl versions and identities of the people that discovered the flaw need to be collected to submit the CVE ID request.

The security team may ask you to clarify the exact name we should use when crediting discovery of the issue. The "Vulnerability credit and bounties" section of this document explains our preferred format for this credit.

Once a CVE ID has been assigned, you will be notified by email. The vulnerability should not be discussed publicly at this stage.

Pre-release notifications

When the security team is satisfied that the fix for a security issue is ready to release publicly, a pre-release notification announcement is sent to the major redistributors of Perl.

This pre-release announcement includes a list of Perl versions that are affected by the flaw, an analysis of the risks to users, patches the security team has produced, and any information about mitigations or backporting fixes to older versions of Perl that the security team has available.

The pre-release announcement will include a specific target date when the issue will be announced publicly. The time frame between the pre-release announcement and the release date allows redistributors to prepare and test their own updates and announcements. During this period the vulnerability details and fixes are embargoed and should not be shared publicly. This embargo period may be extended further if problems are discovered during testing.

You will be sent the portions of pre-release announcements that are relevant to the specific issue you reported. This email will include the target release date. Additional updates will be sent if the target release date changes.

Pre-release testing

The Perl security team does not directly produce official Perl releases. The team releases security fixes by placing commits in Perl's public git repository and sending announcements.

Many users and redistributors prefer using official Perl releases rather than applying patches to an older release. The security team works with Perl's release managers to make this possible.

New official releases of Perl are generally produced and tested on private systems during the pre-release embargo period.

Release of fixes and announcements

At the end of the embargo period the security fixes will be committed to Perl's public git repository and announcements will be sent to the perl5-porters <https://lists.perl.org/list/perl5-porters.html> and oss-security <https://oss-security.openwall.org/wiki/mailing-lists/oss-security> mailing lists.

If official Perl releases are ready, they will be published at this time and announced on the perl5-porters <https://lists.perl.org/list/perl5-porters.html> mailing list.

The security team will send a follow-up notification to everyone that participated in the pre-release embargo period once the release process is finished. Vulnerability reporters and Perl redistributors should not publish their own announcements or fixes until the Perl security team's release process is complete.

Publicly known and zero-day security issues

The security team's vulnerability remediation workflow assumes that issues are reported privately and kept secret until they are resolved. This isn't always the case and information occasionally leaks out before a fix is ready.

In these situations the team must decide whether operating in secret increases or decreases the risk to users of Perl. In some cases being open about the risk a security issue creates will allow users to defend against it, in other cases calling attention to an unresolved security issue will make it more likely to be misused.

Zero-day security issues

If an unresolved critical security issue in Perl is being actively abused to attack systems the security team will send out announcements as rapidly as possible with any mitigations the team has available.

Perl's public defect tracker will be used to handle the issue so that additional information, fixes, and CVE IDs are visible to affected users as rapidly as possible.

Other leaks of security issue information

Depending on the prominence of the information revealed about a security issue and the issue's risk of becoming a zero-day attack, the security team may skip all or part of its normal remediation workflow.

If the security team learns of a significant security issue after it has been identified and resolved in Perl's public issue tracker, the team will request a CVE ID and send an announcement to inform users.

Vulnerability credit and bounties

The Perl project appreciates the effort security researchers invest in making Perl safe and secure.

Since much of this work is hidden from the public, crediting researchers publicly is an important part of the vulnerability remediation process.

Credits in vulnerability announcements

When security issues are fixed we will attempt to credit the specific researcher(s) that discovered the flaw in our announcements.

Credits are announced using the researcher's preferred full name.

If the researcher's contributions were funded by a specific company or part of an organized vulnerability research project, we will include a short name for this group at the researcher's request.

Perl's announcements are written in the English language using the 7bit ASCII character set to be reproducible in a variety of formats. We do not include hyperlinks, domain names or marketing material with these acknowledgments.

In the event that proper credit for vulnerability discovery cannot be established or there is a disagreement between the Perl security team and the researcher about how the credit should be given, it will be omitted from announcements.

Bounties for Perl vulnerabilities

The Perl project is a non-profit volunteer effort. We do not provide any monetary rewards for reporting security issues in Perl.

The Internet Bug Bounty <https://internetbugbounty.org/> offers monetary rewards for some Perl security issues after they are fully resolved. The terms of this program are available at HackerOne <https://hackerone.com/ibb-perl>.

This program is not run by the Perl project or the Perl security team.