IT Security Report

GHCN-Monthly Version 4 Mean Temperature

Version 1.0 / November 15, 2016

1. Introduction

A security check for software associated with dataset releases is an NCEI requirement. Software is scanned by the Information Technology Branch (ITB) security team before any release to the public. In addition, all software is placed in the NCEI subversion repository for version control and archived as part of the GHCN-M v4 data archive package.

https://conman.ncdc.noaa.gov/svn-repos/ghcn-monthly/V4/

2. Security Reviews

The request for a security check of GHCN-M version 4 software was submitted to NCEI ITB in three parts; the v4 update process, the quality control process, and the bias correction process (Rejuvenated PHA code).

a. Update Process

The software for the v4 update process was submitted before the v4 beta release. Several issues were noted and corrected in a series of software updates as described in section 3. Following several improvements as noted below, the updated code was resubmitted to NCEI IT Security (Will Chatham) and passed review on 18 November 2015. No additional changes were made to this process before the operational release.

The following seven recommendations to the Update process software were made by ITB. The responses to each are noted.

i. Software changes to Update Process made in response to IT Security review

1) Explicitly Enable Taint Mode in all Perl files

Taint mode is suggested for any program that's running on behalf of someone else, or when external data is being used. It is employed at the top of the script when calling the Perl program:

```
#!/usr/bin/perl –T
```

While this does a good job at evaluating input datasets for ISTI / GHCN-M datasets, there poses a problem with **output**. When trying to write out metadata and/or data in ISTI / GHCN-M format, the following error occurs:

```
Insecure dependency in open while running with -T switch
```

More specifically, this occurs when opening a file that does not currently exist:

```
open(my $META_OUTPUT, ">", $meta_outfile") || die "cannot open file: $!";
```

The fact that we are trying to create a new file using ">" is not allowed in taint mode. Currently, there is no alternative to this method in Perl, and this must occur in our scripts so new files are created. This process is common with our projects and to write the program in a completely different language to resolve this is inefficient and will take time.

RESPONSE: While taint mode poses a problem with output, the author attempted to test taint mode for the rest of the script. This included commenting out parts of the code requiring output and seeing if taint mode failed on other items. The code was successful when not outputting data on all of the scripts in question. However data output is required as that is the purpose of the process. As such, the author has not implemented taint mode in the scripts.

2) Code exists before warnings are enabled Always use 'use warnings

RESPONSE: This has been implemented and tested without any issues.

3) Bare word file handle in use

This requires changing the file handle from simple text to a variable. See example of changing "INFILE_INPUT" to "$INFILE_INPUT"

OLD

```
open(INFILE_INPUT, "$infile") || die "cannot open file: $!";
```

```
while($readline = <INFILE_INPUT>)
{
  # Do Stuff
}
close(INFILE_INPUT);
```

NEW

```
open(my $INFILE_INPUT, "$infile") || die "cannot open file: $!";
while($readline = <$INFILE_INPUT>)
{
  # Do Stuff
}
close($INFILE_INPUT);
```

RESPONSE: This has been implemented and tested without any issues.

4) Do not use the two-argument form of open()

Similar to #3, this requires a small change to how files are opened. Instead of using two arguments, three should be used See below example, where the “<” handle is added as an argument:

OLD

```
Open($INFILE_INPUT, "$infile") || die "cannot open file: $!";
while($readline = <$INFILE_INPUT>)
{
  # Do Stuff
}
close($INFILE_INPUT);
```

NEW

```
open(my $INFILE_INPUT, “<”, "$infile") || die "cannot open file: $!";
while($readline = <$INFILE_INPUT>)
{
  # Do Stuff
}
close($INFILE_INPUT);
```

RESPONSE: This has been implemented and tested without any issues.

5) Close file handles as soon as possible after opening them

The programs in question read an infile (and outputs to outfile) on a line by line basis. As the code currently stands, the files are opened, read/written on a line by line basis, and then immediately closed. Depending on the input source, this can be done in only a few lines. However in some cases it can take some extra lines of code, due to the nature of the input source formatting.

RESPONSE: It is difficult to find a metric that would assess closing file handles “as soon as possible”. The only way to make them close “sooner” would be to input the data as a large variable, and then parse through that variable. The same would be said for outputting, as the formatting of the data would be done to a large variable, and then can be outputted in an adequate format. While this might be applicable, it would take a huge effort to change the scripts to accommodate these variables. In addition, holding an entire file as one variable could take up a large amount of memory, depending on the input source. Mitigating this would require extra validation and making sure data has not changed with the updated code, which could take time. Because this does not pose a critical risk, it is the author’s opinion to keep the code as is.

6) Subroutine does not end with "return"

RESPONSE: This has been implemented and tested without any issues.

7) Input Sanitization

This requires the code to test if the input data has a format that is expected. For example, we expect infiles to have only certain ASCII characters in it. Therefore, before opening and reading an infile, this subroutine is called:

```
&sanitize_input($infile);
```

And the subroutine is below:

```
sub sanitize_input
{
  $input_infile = shift;
  die "Input contains unsanitary characters, stopped: ".$input_infile."\n" if
($input_infile =~ m|[^-A-Za-z0-9_/.~]|);
  return;
}
```

RESPONSE: This has been implemented and tested without any issues.

ii. ITB Decision

ITB approved of the above changes on November 18, 2015.

On Tue, Nov 17, 2015 at 11:47 AM, Jason Symonds - NOAA Federal <jason.symonds@noaa.gov> wrote:

I appreciate Jared's willingness to work with us and mitigate the findings where possible. I think for the issues that can't be corrected, with the mitigation of the context under which the applications will be run lessens the risk enough to be acceptable.

Please tag the latest version for review to validate the fixes that have been implemented are in place for the ORR and we should be good to go.

Thanks,
Jason

On 11/18/2015 10:58 AM, Will Chatham - NOAA Affiliate wrote:
Thanks Jared, this is much improved.

On Wed, Nov 18, 2015 at 11:50 AM, Jay.Lawrimore <jay.lawrimore@noaa.gov> wrote:
Will, thanks. Just to confirm - can I write this down as passing IT Security review?

Subject:	Re: ghcnm_code_review_response.docx - Invitation to view
Date:	Wed, 18 Nov 2015 12:17:55 -0500
From:	Will Chatham - NOAA Affiliate <will.chatham@noaa.gov>
To:	Jay.Lawrimore <jay.lawrimore@noaa.gov>
CC:	Jared Rennie - NOAA Affiliate <jared.rennie@noaa.gov>, Jason Symonds - NOAA Federal <jason.symonds@noaa.gov>, Mark Noto - NOAA Federal <mark.noto@noaa.gov>, Byron Gleason - NOAA Federal <byron.gleason@noaa.gov>

```
As far as I am concerned, yes.
```

b. Quality Control Process

The quality control software (ghcnm_qc.f95) was submitted to NCEI IT Security (Will Chatham) on 9 September 2016. No security issues were found as noted below.

Fwd: Update: Your incident 22029 has been resolved. Subject: Security code review

Byron Gleason NOAA
Federal <byron.gleason@noaa.gov> Fri, Sep 9, 2016 at 2:33 PM
To: Jay Lawrimore NOAA
Federal <jay.lawrimore@noaa.gov>
FYI, security review for ghcnm_qc.f95 ... no issues found, see below.

Forwarded
message From:
Servicedesk <ldsd.ncdc@noaa.gov>
Date: Fri, Sep 9, 2016 at 2:09 PM
Subject: Update: Your incident 22029 has been resolved. Subject: Security code review
To: Byron.Gleason@noaa.gov

Dear Byron Gleason,
Your ticket has been resolved by Will Chatham. Below you will find the resolution information.
If you have any questions please contact an ITB Representative.

=== Regarding: ===
User: Byron Gleason
Service Request 22029 / Security code review
Details: Need security code review of:
https://conman.ncdc.noaa.gov/svnrepos/
bgleason/ghcnm_qc/ghcnm_qc1.2/
ghcnm_qc.f95
in preparation for ORR at the end of the month.

=== Summary: ===
No security issues of note. Thanks.
Resolved by: Will Chatham

c. Rejuvenated PHA Code

The rejuvenated PHA bias correction software was submitted to NCEI IT Security (Will Chatham) on 9 September 2016. No security issues were found as noted below.

Update: Your incident 22028 has been resolved. Subject: Requesting followup security review

Servicedesk <ldsd.ncdc@noaa.gov> Fri, Sep 9, 2016 at 9:51 AM
ReplyTo:
Servicedesk <ldsd.ncdc@noaa.gov>
To: Diana.Kantor@noaa.gov

Dear Diana Kantor,
Your ticket has been resolved by Will Chatham. Below you will find the resolution information.
If you have any questions please contact an ITB Representative.

=== Regarding: ===
User: Diana Kantor
Service Request 22028 / Requesting followup

security review

Details: The ghcnmpha
project had an initial security review last May (ticket 19880). The code has not changed in any
significant architectural way, but there have been some code changes related to bug fixes and enhancements. We have an ORR on September 28.
Here is a link to the latest tag:
https://conman.ncdc.noaa.gov/svnrepos/
ghcnmonthly/
ghcnmpha/
tags/ghcnmpha01.01.06/
There is a README file in the project root directory.
Thank you!

=== Summary: ===
No new security issues found. Thanks.
Resolved by: Will Chatham