This document seeks to communicate some of the design rationale and challenges behind the methods. The discussion does not cover all the actions taking place within the core “gawk” pattern matching or character substitutions taking place in any of the scripts. I leave that to your analysis should you choose to explore the flow of operations or direct questions to eber0206@gmail.com.

Background: Overcoming the processing time and productivity losses imposed on shoveling through the record structure overburden of the NSRL design each time a new subset record-set might be wanted became the inspiration for developing the three "awk-centric" bash “data reduction” scripts "**NSRL2MD5.bash, prodcodecarver.bash, and recordcarverx.bash**" Together, run in the order listed, they enable the ability to ask questions like … ”Does the evidence image contain any files from John The Ripper, and also contain files from Python?” All I need do is carve all the MD5 and FileName fields for each from the NSRL and ingest them as “known bad” hashes to Autopsy, and run the analysis. What could be easier!

But…waaaa-a-ay back in 2000 ... ask NIST, the NSRL RDS design became infested with "data embedded-commas and spaces" that encumber the parsing process. NIST further burdened the structure using external “quote" marks as secondary delimiters. This enables embedding commas between other meta-data or extended descriptors in the FileName field, but it severely complicates evaluation scripting when attempting to carve specific fields out of the records. Because of these issues, dividing the carving process into 3 pieces proved key to enabling coherent "downstream" user productivity gains and flexibility. So to describe how is this done, I’ll discuss the madness to the hak.

Unconventionally (because of NIST design choices,) I wrote prodcodecarver.bash so it treats each NSRLProd.txt single quote “ symbol as the record delimiter (rather than pairs of quotes or commas which were probably the traditional NIST intent.) This enables the prodcodecarver.bash "gawk" to select records of field $2 “ProductName" (eg: Windows 7) and field $12 "ApplicationType" (eg: Operating System) and writes the field $1 "ProductCode" output value to prodcodecarver.txt". The list of extracted product codes is then provided to the third script recordcarverx.bash which outputs a final indexed MD5 and FileName record pair for ingest into Autopsy or some other compatible tool of your choice.

By replacing $2 or $12 values and strings in the "gawk" code line, you can carve NSRLProd.txt for other choices. Look inside the NSRLProd.txt for these other possible product choices. Your field carving choices are:
$1 = ProductCode (what we always carve for, so this is not a choice to change)
$2 = ProductName (what product is usually carved for, example: "Windows 7")
$3 = (a comma, a useless thing to carve for)
$4 = ProductVersion (not usually useful to carve for, but maybe you have a reason to)
$5 = (a comma, a useless thing to carve for)
$6 = OpSystemCode (not usually useful to carve for, but maybe you have a reason to)
$7 = (a comma, a useless thing to carve for)
$8 = MfgCode (not usually useful to carve for, but maybe you have a reason to)
$9 = (a comma, a useless thing to carve for)
$10= Langauge (not usually useful to carve for, but maybe you have a reason to)
$11= (a comma, a useless thing to carve for)
$12= ApplicationType (is a very useful value to carve, for example "Operating System", to enhance more selective "AND" carving with $2 “ProductName”)

Now we can discuss usage of the scripts in the correct order.

Script 1, NSRL2MD5.bash is used to reduce the **NSRLFile.txt** 8 field record structure of **"SHA-1","MD5","CRC32","FileName","FileSize","ProductCode","OpSystemCode","SpecialCode"** to only the fields needed for usage in tools like Autopsy. The only intermediate fields I end up using are **"MD5","FileName","FileSize","ProductCode"** but you could change the code to use SHA-1 for example if you prefer.

Here is a complete NSRLFile.txt eight field record for example ProductCode 14801, starting with the SHA-1 at the far left:

"006F91693F31C64179211E72A7F63B4CEFDF95A3","66D15C0E298A89F2F8446A0C7A7F536F","910E01EC","advfea19.jpg",3922,14801,"358",""

The script NSRL2MD5.bash transforms the NSRLFile.txt eight field records to 4 fields of **MD5/FileName/FileSize/ProductCode** with "**/**" as the replacement separator for " or , and saves it to the file NSRLFile2.txt. NSRL2MD5.bash can take about 3 and half hours of pre-processing for all the records of the NSRL on a dual core cpu but only needs to be done once for each new NSRL release. This step was done to simplify confused record delimiters and reduce the NSRL, as well enable more simple downstream scripting. The benefit is increased productivity if you carve multiple sub-sets out of the full NSRL. For instance, using this method, carving all 12 codes for all "Windows 7, Operating System" records from the reduced and optimized NSRLFile2.txt record-set takes 20 minutes instead of 1 hour and 35 minutes.

Example NSRL2MD5.bash output: **66D15C0E298A89F2F8446A0C7A7F536F/advfea19.jpg/3922/14801/**

We leave that new hash database file for now and turn to reducing the **NSRLProd.txt** to only a list of product codes for those we want. We need to use the second script "prodcodecarver.bash" to do this. NSRLProd.txt has variable data that creates problems by intermingling field delimiters in with the data field. Thus, I need to discuss how I handle this in the prodcodecarver.bash script. "/" was chosen as my replacement field delimiter because it is the only common character that is invalid as part of a filename for both Windows and Linux, and so thus makes an ideal field delimiter of choice while permitting NIST to embed commas in the records. Unfortunately, they also embed a few "/" in the extended information of the FileName field but in my application they become a "don't care" condition.

Consider the NSRLProd.txt record 14801 below. It is a perfect poster-child example of an "embedded-comma" NSRL design mess. This extreme record is a mashup of data and 24 commas! Because of all the data embedded commas, bash cannot sort out the fields using comma-delimitation. Instead, it uses the quote symbols as singles (previously described at the beginning of this document). The example red coded objects represent the odd numbered fields and the blue ones are the evens from $1 to $12.

14801,"Dell Optiplex Resource CD for Reinstalling Device Drivers and using Diagnostics, Utilities and System Requirements","June 2001","190","257","Chinese-Simplified,Chinese-Traditional,Dutch,English,French,German,Italian,Japanese,Korean,Norwegian,Polish,Portuguese,Portuguese Brazil,Russian","Diagnostic,Documentation,Drivers,system utility"

prodcodecarver.bash (this script) searches for your target strings in **$2** and the **$12** and saves the value 14801 of $1 to productcodecarver.txt (the script strips off the quote and the comma).

Prodcodecarver.bash submits productcodecarver.txt to recordcarverx.bash to again reduce the NSRLFile2.txt record from 66D15C0E298A89F2F8446A0C7A7F536F/advfea19.jpg/3922/14801/ to only MD5 and FileName of 66D15C0E298A89F2F8446A0C7A7F536F advfea19.jpg (with one space delimiter between each field, no commas, no quotes, no forward-slash) This compatible file (and companion idx file) can now be submitted to Autopsy (or other tool) for ingest as a hash table for evidence file matching analysis.

To NIST I encourage they re-design the record structure to use "/" forward-slash as the field delimiter for all NSRL field delimiters in the records instead of mixed quotes and commas/embedded commas.

Since the character "/" is the only common disallowed character in linux and Windows filenames, it makes the "best choice" to use as a delimiter (of course, please remove any / embeds in your data)

Quoting Jesse Kornblum, "The National Software Reference Library: The Best Reference Which Nobody Uses." And I know why, but I also now have the means to use it effectively, so that dog can go sleep. This is no excuse for NIST to not reform their NSRL structure however.

Because I discuss much of prodcodecarver.bash in this document, I include it here as ready reference.

 # SCRIPTNAME: prodcodecarver.bash
CODE START

gawk 'BEGIN { FPAT = "([^\"]+)" } (**$2** ~ "**Windows 7**" && **$12** ~ "**Operating System**") { gsub (/,/,"",**$1**); print $1 }' IGNORECASE=1 NSRLProd.txt | sort | uniq > prodcodecarver.txt

Pass in the carved codes to the NSRLFile2.txt hash-carver script
./recordcarverx.bash $(< prodcodecarver.txt)