

IN-PLACE FILE CARVING

ScalpelFS

Golden G. Richard III, Vassil Roussev and Lodovico Marziale

Department of Computer Science, University of New Orleans

New Orleans, Louisiana 70148, USA

golden,vassil,lmarzial@cs.uno.edu

Abstract File carving is the process of recovering files from an investigative target without knowledge of the file system structures. The process is based on information about the format of the files of interest, as well as on assumptions of how the file data is usually laid out on the block-level device.

The output of current carving tools invariably is a set of new files created in the host file system of the investigator's machine. These files consist of copies of blocks from the target image that meet various carving rules. These new files are treated as *candidates* as they may constitute false positives—files that are either invalid or did not exist in the recovered form on the target. By recreating them as files in a functional file system, it becomes possible to validate them using file-based tools.

In practice, it is quite common to end up with a large volume false positives. Despite continuing work on improving the file carving tools in that respect, the inherent uncertainty of the process will always lead to a non-trivial number of false positives.

In this paper, we present an in-place approach to file carving, in which we recreate the file system metadata without actually copying the file contents. This results in significant reduction of the storage requirements (even in pathological cases), much shorter turnaround times, and opens up new opportunities to perform on-the-spot screening of evidence (triage).

Keywords: Digital forensics

1. Introduction

To understand the impetus behind our work, consider a recent data recovery case, in which carving a modest 8GB target was yielding about

1.1 million files with a total size of 256GB— an 'over-carving' factor of 32. Clearly, this is a pathological case and an expert tool user would likely be able to bring down the numbers significantly by tinkering with the carving rules, yet it demonstrates that surprises can come even in small packages. For the 200-300 GB typical hard disk currently packaged with the average desktop system even a more typical over-carving factor in the 2-3 range would put a significant strain on the resources of a high-end workstation and could easily add days of delay to an investigation.

It is not difficult to pinpoint the reasons for such a huge performance penalty. The particular tool we use in our experiments—*scalpel* []—allows us to easily separate the time it takes to identify the candidate files from the the time it takes to recreate them in the host file system. In *scalpel*, these two tasks are performed during two separate passes over the target. During the first pass, the entire target is read sequentially and is processed with respect to each carving rule. The result is a list of carving jobs indicating which sequences of blocks constitute candidate files. Given a non-trivial number of rules, the process is CPU-bound and, for the example 8GB image, takes approximately 70 minutes to complete.

During the second pass, the carving jobs are carried out by placing copies of the identified data blocks into newly created files. This is a completely I/O-bound process with write access to the host file system being the limiting factor. In the example, the second pass takes approximately 15 hours to complete.

Thus, although the root of the problem is false positives generated by the carving rules, the real penalty for the disastrous performance comes from the actual recreation of the files. Specifically, the main culprit is the randomized set of write accesses generated during this process, which brings out the worst performance in mechanical HDD. Note that the specific block writes are a function of the host file system and are beyond the control of the carver. Obviously, *any* level of over-carving will guarantee randomized access as the same data block is copied to more than one new location on the host file system and the corresponding file metadata is updated. It is worth observing that any optimizations based on the interleaving/pipelining of *scalpel*'s two passes (as some other tools do) has very limited potential and will not solve the problem, as the file creation clearly dominates the total execution time.

Arguably, the penalty for recreating the candidates goes beyond the carving itself and is carried into the subsequent processing steps. Consider our 8GB target and a 'reasonable' over-carving factor of two, and assume that the goal is to carve out all JPEG images. On a modern workstation with 4GB of RAM, 40-45 percent of the target could con-

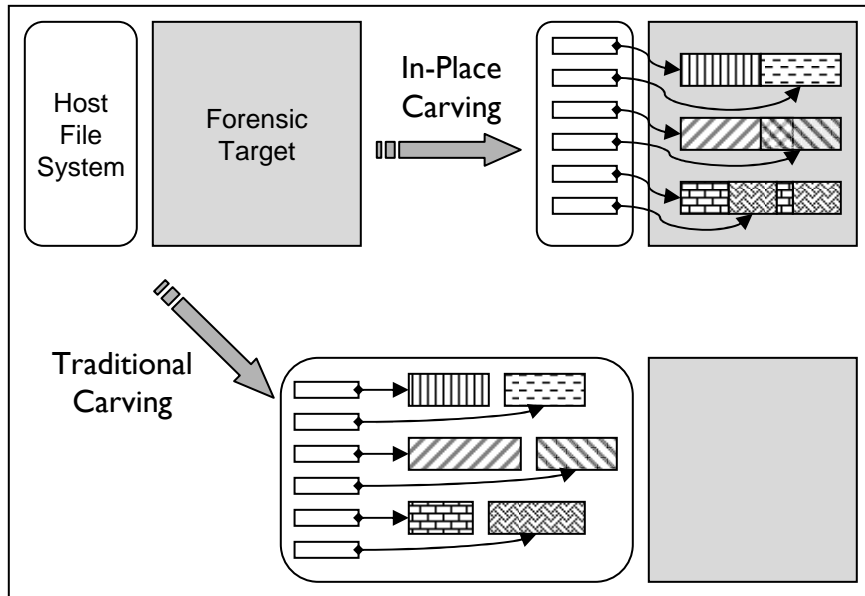


Figure 1. Conceptual differences between traditional and in-place carving

ceivably be cached. With proper clustering of file access based on the work queues, it is possible to actually take advantage of the caching effects most of the time. In contrast, reading 16GB of recreated files yields *no* tangible caching benefits due to non-overlapping file operations.

In summary, the current practice of carving out the candidates into new files carries a huge performance penalty that is inherent and cannot be solved by optimizing the carver. The *only* justification of this approach is that virtually all tools used in subsequent viewing/processing require a file-based interface to the data.

In other words, if we provide a file system interface to the candidate files without physically recreating them we could still use our tools without the new-file overhead. We refer to this approach as *in-place carving*. It is very similar to creating a file system based on the block-level provided by the target with the notable exception that file metadata is stored *outside* the target. In both cases the result is a list of candidates accessible via a standard file system interface. Figure [?] provides a quick illustration of the conceptual differences between traditional and in-place carving.

For the rest of this section we explore two baseline scenarios where in-place carving can make a significant difference.

Large-scale carving. It is almost self-evident that carving on a terabyte scale target today would qualify as a large-scale job. There, the problems are compounded by the fact that high-capacity drives tend to be noticeably slower than the fastest drives available. However, we use the term 'large-scale' here in a relative sense—any target that would stretch (or go beyond the limits of) the currently available resources would be viewed as large-scale by the examiner so even a 100GB could prove to be a challenge.

It is relatively straightforward to argue that in-place carving is more scalable as far as storage requirements are concerned. The contents of a target is dominated by file data with metadata usually taking up well under five percent. By choosing not to copy file content (often more than once), we limit the overhead to a small fraction of the target size. In our extreme 8GB example, the new metadata for 1.1 million candidate files would take up less than 128MB if we allow 128 bytes of metadata per file, or 1.6 percent. Extrapolating to a 1TB target, the metadata is small enough to fit on a miniature USB drive.

The other aspect of scalability here is turnaround time—how long after initiating the carving can the examiner begin work in earnest on the results. Using *scalpel*'s preview mode, which performs only the first carving pass and outputs the work queues, we are ready to deliver any of the candidate files after 70 minutes of work on the 8GB target. Indeed, our prototype described later uses exactly this approach as the work queues provide enough information to build the filesystem metadata.

A third aspect of providing a scalable solution is the flexibility to quickly react to the specifics of the target and adjust the carving settings to minimize false positives or refocus the investigation. In our example, we could perform over 10 different in-place carving runs in the time it takes to perform a single traditional one. As we discuss in future work, it is not hard to conceive a system that will automatically correlate identified false positives and the carving rules that generate them. This would be considerably more difficult to achieve using traditional carving methods because that kind of information gets lost during file generation.

Triage. It is often desirable to preview a possible target to provide a preliminary assessment on the degree of relevance to the inquiry. This can be done either in a lab setting or in the field to either prioritize or filter out targets. In the case of carving, only an in-place approach can deliver the necessary short turnaround time.

Going a step further, it is possible to perform a carve on a live machine which cannot be taken down for practical and/or legal reasons, such as cases of voluntary cooperation. Certainly, it would be more difficult to present such results directly in court, however, any positive results would

be more than likely to provide a probable cause. In a less restrictive legal environment, such as an internal corporate inquiry, where the goal is not to go to court but to solve a specific problem, live carving would have even more applications.

Another important aspect of in-place carving is that it preserves the privacy of the original information because no copies of the file data are made. Furthermore, it is easy to convince the owner of that since the investigator is unlikely to need anything more than a flash USB drive to initiate and store the carving results. A nice bonus of the in-place approach is that, once performed, the generated metadata would be readily usable on any copies performed later.

The forensic triage (whether in the lab or on the spot) could potentially be performed in parallel on a group of machines that could be controlled by a single investigator over the network. The idea of on-the-spot forensic investigation has already been explored in a more general setting by *Bluepipe*[]. In that context, file carving is just a special function that could be performed and controlled in parallel.

In summary, we have argued that traditional file carving approaches, which create new files for each candidate, are wasteful and carry significant and unavoidable performance overhead. The main contribution of this work is to demonstrate that the benefits of file-based access to the candidates can be realized by recreating file metadata without copying file contents. We expect the resulting system to scale much better with target size and to enable new uses of carving, such as in on-the-spot triage.

2. Related Work

3. Design of ScalpelFS

4. Experience

5. Conclusions

Traditional file carving relies on the complete recreation of candidate files identified by carving rules into the host file system. This results in poor performance primarily because the set of resulting block-level write commands issued by the host filesystem is randomized. This problem is magnified by the presents of non-trivial number of false positives, which are the result of inherent uncertainties in interpreting the original data. In pathological cases, the amount of data carved can exceed the size of the target by an order of magnitude. For larger targets, even more typical over-carving factors of 2-3 can require too much disk space and take too long.

In this paper, we propose the alternative of *in-place* carving, which recreates the file metadata outside the target but uses the original image for retrieving file contents. This is a viable approach, which allows carving to be performed at a fraction of the cost without losing any functionality. In addition to supporting new usage scenarios, such as large-scale carving and on-the-spot carving, the in-place approach holds promise with respect to dealing with false positives and providing short turnaround times.

We presented a proof of concept system (*scalpelFS*), which presents the results of the file carving performed by the *scalpel* file carver using the standard filesystem interface.

[A COUPLE OF PARAGRAPHS ON EXPERIENCE HERE]

6. Future Work

The first task on our to-do list is to create a more robust version of our code and to perform more extensive testing to better characterize the capabilities and shortcomings of our system. As soon as the code is stable enough, we expect to release our implementation in an open-source fashion.

A longer-term goal is to further reduce the wait time by presenting a dynamically updated view of the file system—from the point of view of the examiner, files will literally appear as they become available. This allows for an either manual or automation step to be overlapped with the carving process.

Going a step further, the information from the validation step could be fed back into the carving engine, which could be used to disable, or delay to a second pass the evaluation rules that generate an inordinate number of false positives. For example, on a target in excess of 100GB it may well be the case that by the time 10-20 percent of it is processed for carving, some preliminary feedback may significantly reduce the processing of the rest.

References

- [1] B.Carrier, Sleuthkit and Autopsy, <http://www.sleuthkit.org>.
- [2] Forensics Toolkit (FTK), <http://www.accessdata.com>.
- [3] "FUSE: Filesystem In User Space," <http://fuse.sourceforge.net/>.