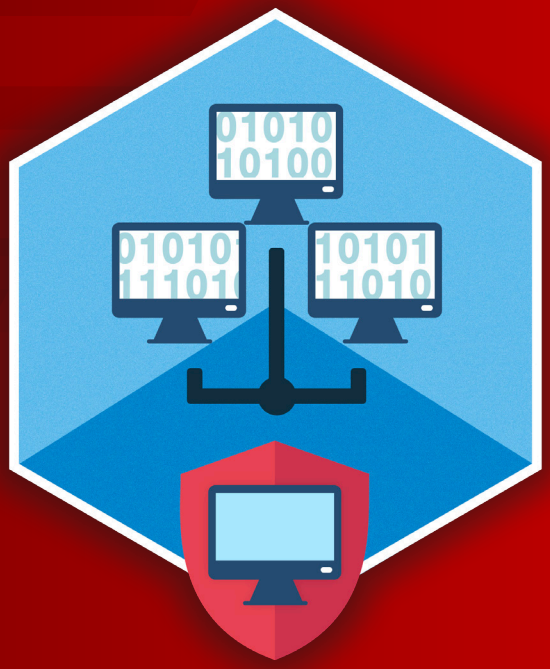
ELCOMSOFT
DESKTOP, MOBILE & CLOUD FORENSICS



VS

ELCOMSOFT DISTRIBUTED PASSWORD RECOVERY

HASHCAT

Hashcat is a great, free tool competing head to head with the tools we make. We charge several hundred dollars for what, in the end, can be done with a free tool. What are the reasons for our customers to choose ElcomSoft products instead of Hashcat, and is the expense justified? We did our best to compare the two tools to help you make the informed decision.

Both Hashcat and Elcomsoft Distributed Password Recovery are tools for breaking passwords.

Both tools can perform hardware-accelerated brute-force attacks using conventional video cards, and both tools can do dictionary and smart attacks.

Both tools can use distributed networks to speed up the attacks, yet Hashcat needs a third-party tool for that.

Both are thoroughly optimized to offer the highest performance on hundreds of file formats. On paper, the two tools have so many similarities they may look alike. The usage experience in typical use cases, however, could not be more different.

We did our best to compare the two tools to help you make the informed decision.

SYSTEM REQUIREMENTS

Elcomsoft Distributed Password Recovery

Elcomsoft Distributed Password Recovery is a Windows only product. You need Windows 7, 8, 8.1 or Windows 10 to run both the server and each of the clients. If you are using distributed attacks, each of the computers comprising the distributed network must run Windows.

In addition to Windows, you need proper drivers. Elcomsoft Distributed Password Recovery is comfortable with officially released versions of NVIDIA drivers; we've seen no nasty surprises with either Studio or Gaming drivers as long as they come with CUDA. AMD drivers are a bit sketchier, so we have to test major releases for compatibility. If you're using a different GPU, the correct OpenCL driver must be installed in order for the hardware acceleration to work.

Hashcat

Hashcat is a cross-platform tool supporting all major versions of Windows, macOS, and many Linux distributions. However, you'll be hard-pressed to find one single platform on which all of the Hashcat features work. From time to time, you'll need Hashcat to use several host operating systems to perform its duties. As an example, you'll need a Mac to extract encryption metadata from FileVault disks.

In addition to the host OS, you'll also need the following to utilize some or all Hashcat features.

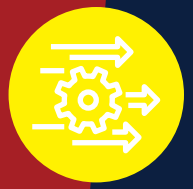
- git (optional but recommended)
- Perl
- Python
- C/C++ compiler, correctly configured and equipped with the required libraries. Some hash extraction utilities are available exclusively in source code with no precompiled binaries.

HARDWARE ACCELERATION

Both tools advertise hardware acceleration to speed up brute-force attacks. The devil is in the detail.

Elcomsoft Distributed Password Recovery

Elcomsoft Distributed Password Recovery offers GPU acceleration on NVIDIA video cards, while some formats can utilize AMD hardware. Our tool can also utilize OpenCL, the open standard for parallel programming of heterogeneous systems. Thanks to OpenCL support, Distributed Password Recovery can utilize the GPU units found in many Intel processors, which can deliver several times the typical performance of CPU cores. Should GPU acceleration be unavailable on a given system or for a given format, the tool transparently falls back to CPU-only implementation.



Hashcat

Hashcat is decidedly GPU only. The tool requires an AMD or NVIDIA video card to work, and you'll need a compatible version of the graphic driver. Where Hashcat requires the use of the OpenCL runtime to interface with compatible CPUs (which we struggled to enable), Distributed Password Recovery addresses the CPU cores directly, utilizes the native CUDA for interfacing with NVIDIA boards, and uses OpenCL for everything else.

SUPPORTED FORMATS

Even the best password recovery tool is worth nothing if it does not support that one encrypted data format you urgently need to break. On paper, Hashcat supports a larger number of formats compared to our tool. While both Hashcat and Elcomsoft Distributed Password Recovery advertise hundreds of supported formats and generally tick all the basics, our tool covers a few things that Hashcat does not. These include:

- Office 97/2003 (key search)
- PDF with 40 bit encryption (key search)
- Apple iWork (Pages, Numbers, Keynote)
- Hangul/Hancom Office (HanWord, HanCell)
- PGP ZIP archives (.PGP) (password recovery)
- PGP secret key rings (.SKR) (passphrase recovery) (GPU accelerated)
- PGP disks with conventional encryption (.PGD) (password recovery) (GPU accelerated)
- PGP self-decrypting archives (.EXE) (password recovery)
- PGP whole disk encryption (password recovery) (GPU accelerated)
- Intuit Quicken
- macOS keychain password (GPU accelerated)
- BlackBerry backups (.IPD, .BBB) (password recovery) (GPU accelerated)
- FileMaker
- Apple Disk Image (.DMG) passwords
- DashLane password manager
- Tally ERP 9 Vault passwords

SETTING UP AND INITIAL CONFIGURATION

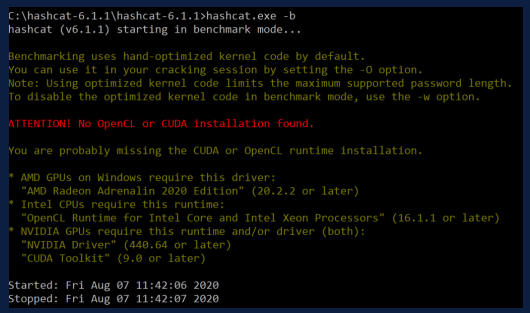
Elcomsoft Distributed Password Recovery

The greater differences are experienced when setting up and configuring the tools for the first time. When installed on a single computer, Elcomsoft Distributed Password Recovery is essentially a plug-and-play endeavor: you download the installation file, double-click to install, then launch the product from the Windows Start menu. That's pretty much it; the tool automatically launches the required components (the client, server, and interactive management console) and detects the available acceleration resources. By the time you see the main window, the tool is ready to run your first attack.

To sum it up, you need a supported OpenCL device to use Hashcat. NVIDIA boards are supported; you'll need some luck for AMD or Intel. If you use Ubuntu, you'll require an NVIDIA board. Elcomsoft Distributed Password Recovery, on the other hand, just… works.

Hashcat

Hashcat offers a typical Linux experience even if you are using it in Windows. All goes well if you already have the compatible GPU with compatible drivers. Should something go wrong, and you may need to spend extra time troubleshooting. This, for example, is what we've seen when trying to benchmark Hashcat on an Intel CPU in a Windows VM:



Apparently, an OpenCL runtime is required to run attacks on Intel CPUs. Obtaining OpenCL runtime for Intel Core was borderline amazing. The runtime is only available to developers with Intel developer accounts. Registering the account and waiting for its approval took a day; however, we were still unable to run Hashcat on that computer even with the OpenCL runtime installed.

To its credit, Hashcat would run normally in Windows once we installed a compatible NVIDIA board. We could also launch Hashcat in Linux using the same video card. This time the quest was less puzzling; all that we needed to do was using the particular version of Ubuntu, carefully uninstalling the pre-installed NVIDIA drivers (a wrong move bricks the system), adding the recommended drivers repository and installing the recommended driver from that repository. Piece of cake if you use NVIDIA. We failed the quest when we tried using an AMD card though.

WORKFLOW: LAUNCHING THE ATTACK

We have already noted that Hashcat is a command-line tool, while Elcomsoft Distributed Password Recovery is based on the GUI. You'd naturally expect some differences in the workflow, but there's more to that than meets the eye. The two tools are very different from the get-go.

Hashcat

Hashcat only attacks hashes; hence the name. You cannot just point Hashcat to a ZIP file or a Word document. Instead, you must first run a separate third-party script to extract the hash from the file you're about to attack. These scripts are not delivered with Hashcat, so you'll have to look for them and obtain them separately. As an example, when attacking a Microsoft Office document, you must first process the original document with *office2hashcat.py*.

As you can see, the script is a Python script, which means you'll have to install Python with a number of modules. On some systems, the pip module installer is not always installed with Python, so you'll have to obtain and install that (note that the workflow is different for the different host operating systems).

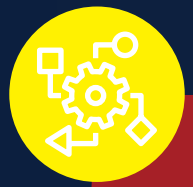
Once you install Python, run the script and extract the hash, you'll receive the hash string, e.g.

*office$*2010*100000*128*16*a1688e8975694550a7a61b5*

While "office$*2010" suggests that the hash belongs to a document saved in a certain Microsoft Office format, Hashcat won't take it for granted. When running the attack, you'll have to additionally specify the "-m 9500" parameter in the command line, which identifies the hash as an Office 2010 file. Why "9500"? Because RTFM.

Dealing exclusively with hashes has its pros and contras. The obvious drawback is the added complexity. If you have more than a single document (or maybe several thousand files in various formats), setting up the attack can take considerable time. Unless you name the files with hash data properly, you may get lost in which hash belongs to which file.

There is a positive side to hash-only recovery. You can keep the original files and their hashes on different computers or even in different networks. You can extract hashes in one lab and run the attack in another, or even send the hashes to a specialized password recovery service without the risk of leaking any personal or confidential information.



Elcomsoft Distributed Password Recovery

What about Elcomsoft Distributed Password Recovery? When originally released, we aimed for simplicity, allowing to simply open a file to launch the attack, end of story. Distributed Password Recovery is smart enough to analyze the file's content, automatically detecting the correct file format even if the file was renamed or has no extension.

Since May 2020, we are optionally offering tighter control over personal information with attacks on encryption metadata. If you prefer, you can use the straightforward classic workflow, but you are no longer limited to that. You can also run the included Elcomsoft Hash Extractor tool to extract encryption metadata (a.k.a. hashes) from certain file formats such as password manager databases or the various office documents. For encrypted disks, you'd use Elcomsoft Forensic Disk Decryptor instead, which is also included.

DISTRIBUTED COMPUTING

Just as the name suggests, Elcomsoft Distributed Password Recovery is designed to work on distributed networks from the get-go. Hashcat supports hashtopolis (previously known as hashtopussy), which is a separate project that works as a wrapper to enable distributed computing.

Hashcat

Enabling distributed computing with Hashcat requires installing a server module. The server is designed as a cross-platform module, yet the documentation is completely Linux-centric. In order to install the server module, you will have to ensure that all of the following components are installed and configured:

- MySQL
- Apache
- PHP
- Perl
- Python

After installing these packages (either individually or as the lamp-server package), you can install hashtopolis. You must have a clear understanding of Linux administration to install and run hashtopolis. The server can be controlled via the Web interface.

You will also need an agent installed on each computer that will become parts of the distributed network. The agent is a ZIP file that contains an app written in Python. The distribution kit is assembled dynamically for each agent; the server's IP address will be hard coded.

In order to run each agent, you'll need the Python environment. As a result, agents are running in the user space. If you were using Windows (which is already unlikely at this point), you'll have a problem if you want agents running as system services (i.e. without an authenticated user session). Needless to say, you must also install Hashcat on each computer that runs agents, and you have to make sure there are no unresolved issues with the drivers, OpenCL runtimes etc.

To sum it up, Hashcat can work in distributed networks, but rolling out the system is a pain. We found it's easier to write our own tool instead.

Elcomsoft Distributed Password Recovery

Elcomsoft Distributed Password Recovery installs the server automatically. No configuration is required, but you can do some fine-tuning. Agents can be manually installed or deployed silently over the network through the Windows domain. Once installed, the agents are immediately ready to work with zero configuration required.

Just as the name suggests, Elcomsoft Distributed Password Recovery is designed to work on distributed networks from the get-go.

CLOUD COMPUTING

Elcomsoft Distributed Password Recovery

Elcomsoft Distributed Password Recovery supports two major cloud providers offering Windows virtual machines: Amazon and Microsoft Azure. We supply a ready-made, prepackaged image that can be deployed into Amazon cloud in several clicks. The agent running in the newly created instance is automatically provided with the server's IP address.

Installing EDPR agents into Microsoft Azure is also automated. The supplied script will automatically install the latest version of the agent to any number of existing (grouped) instances. The only input required is your Microsoft Azure authentication credentials (the login and password). The installation process is completely transparent.

Hashcat

Cloud computing in Hashcat is available through a separate, third-party wrapper called hashtopolis. If you managed to set up and configure hashtopolis on a local computer, you can probably do it in the cloud as well. There are no tools to simplify cloud deployment, so depending on your skills and experience configuring Hashcat to work with cloud instances may take longer than you had originally expected.

Depending on your skills and experience configuring Hashcat to work with cloud instances may take longer than you had originally expected.

THE ATTACKS

When recovering a tough password, almost everything depends on the quality of the wordlist and the type and configuration of the attack.
Let us see how the two tools compare.

Elcomsoft Distributed Password Recovery

Brute force and mask attacks

Plain brute force attacks almost never succeed on anything but the simplest passwords, yet they still might be effective against shorter passwords and uncomplicated protection.

Elcomsoft Distributed Password Recovery supports a similar concept, but allows specifying the minimum and maximum password length as well as the character set either globally (for the whole password) or in groups or characters, allowing to quickly build an attack that takes into consideration the human factor. Brute-force and masks attacks are configured in the GUI.

Dictionary attack

Elcomsoft Distributed Password Recovery provides automatic dictionary distribution to all connected agents including those working in the virtual instances. Dictionary attacks support a large number of pre-defined mutations, allowing to try the most common combinations of dictionary words such as Password, password1, Pa$$w0rd, password1965 and a lot more. We recommend our users trying a small dictionary with certain mutations first, as this often finds frequently used passwords. We also include the dictionary of the top 10,000 most popular passwords from several major leaks, which may help breaking up to 30% of cases in English-speaking countries.

Hashcat

Brute force and mask attacks

In Hashcat and hashtopolis, you specify the attack through the command line or list the attacks in a text file. In Hashcat, brute force attacks are a subset of the mask attack. As an example, if you want to try all passwords that consist of small Latin letters and are 1 to 5 characters long, you can specify the following attack:

```
?l
?l?l
?l?l?l
?l?l?l?l
?l?l?l?l?l
```

If you are using hashtopolis, you'll need to enter these five strings into the corresponding text field.

Dictionary attack

The two tools implement dictionary attacks differently. Hashcat accepts the dictionary as a wordlist, trying each entry the way it is stored in the dictionary. There are no pre-defined mutations and no ability to add a mask when running a dictionary attack (these are parts of the rule-based attack). No automatic distribution of the dictionary file is available if you are running hashtopolis; agents will only use the dictionaries that are already installed.

Combinator attack

This is a simple combination of two dictionary words from two dictionaries (same or different files). Both Hashcat and Elcomsoft Distributed Password Recovery support this attack.

Hybrid attack

While both tools support hybrid attacks, the term is used in different meanings.

Hashcat-style hybrid attacks combine one dictionary word (from a single dictionary) with one mask. This would be considered a particular case of a mask attack in Elcomsoft Distributed Password Recovery. In our tool, one can use multiple dictionaries, where any part of the mask can be a dictionary word. We believe that our solution has an edge over Hashcat here.

Rule-based attack

What Elcomsoft Distributed Password Recovery calls a “hybrid attack” is classified as a “rule-based attack” in Hashcat.

Otherwise, the implementations are very similar; we would even call them identical, as booth tools are following the same syntax as John The Ripper, the tool that originated this attack.

Attacks: conclusion

Booth tools support a wide range of attacks. Dare we say, setting up attacks in Elcomsoft Distributed Password Recovery is simpler and more straightforward compared to Hashcat. Everything is configured from the graphical user interface. A live preview of sample passwords is displayed for every mask or mutation enabled.

To its credit, Hashcat has the ability to run a set of several consecutive attacks, a feature that Elcomsoft Distributed Password Recovery lacks. One can, however, create multiple individual attacks on the password, and run them in the regular job queue.

BENCHMARKS

If you read to this point, you must be interested which tool is faster. While we could make and post numerous benchmarks, the truth is that both tools are highly optimized, and both can exploit the available hardware resources to the maximum. Trying over a hundred formats, we've seen performance fluctuations of around 10 per cent. For many formats, Elcomsoft Distributed Password Recovery offers a (very) slight edge over Hashcat thanks to the additional utilization of available CPU cores (Hashcat, as you remember, is GPU-only). Whichever tool you choose, you'll be happy with its raw performance – or, at least, it doesn't get much better if you jump ship.

THE COSTS

If you are an open-source fan, you're going to like this chapter; less so of you are a (paid) customer.

Elcomsoft Distributed Password Recovery

Elcomsoft Distributed Password Recovery is as much plug-and-play as a forensic grade distributed computing tool can be. Installing the tool on a single computer is no different from installing any other Windows program, while deploying agents to network computers is similar to deploying any other package in your Windows domain. Prepackaged agents make cloud instances straightforward to set up and to use. The attacks are as fast or slightly faster than Hashcat, while being significantly easier to set up.

That level of user-friendliness and simplicity comes at a cost. We charge $599 for a license covering the parallel use of 5 agents. If you are expanding into the cloud, you will notice the differences in cost between Linux and Windows-powered VMs.

The cost of hardware should be the same for both products. The more and the faster video cards you install, the faster the attacks.

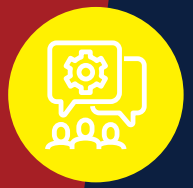
Hashcat

Hashcat is free. Hashcat, hashtopolis, most hash extraction scripts and the host OS (Linux) are absolutely free to use, modify, recompile, or make a custom version just for you, on one condition: you do it yourself or you pay someone to do it for you. Using Hashcat requires skills and experience. Setting up Hashcat on a single computer requires deeper skills, more experience and quite some time. Preparing the tool to run on a distributed network requires skills and experience in Linux administration, a lot of time and a bit of trial and error. Writing attack scripts takes time, and even adding an agent to hashtopolis when you expand your distributed network takes time. It takes time processing disk images of encrypted disk, and it takes time to obtain hash extraction scripts for the occasional new file format.

TECHNICAL SUPPORT

Elcomsoft Distributed Password Recovery

Elcomsoft Distributed Password Recovery is supported by us via the standard ticket system. We stand behind our product, quickly resolving the issues once they are reported.



Hashcat

If you have a problem, Hashcat has a wide community of users and an active forum where you can search for or post questions about issues. What you may not like is the answers. While we had no problem finding the answers to our issues (e.g. attacking a large ZIP archive), finding that what you want is simply not an option can be a little… discouraging.

THE EXTRA FEATURES

There are a few features that can greatly improve usability of Elcomsoft Distributed Password Recovery over Hashcat.

Password cache

If we find a password to one job, we add that password into a custom dictionary. Once the next job is started, we'll try passwords from that dictionary first. This helps save time if the same password is reused across multiple files or data formats.

Distributed updates

When you update **Elcomsoft Distributed Password Recovery**, all components of the distributed network are updated automatically including all connected agents.

Hashtopolis makes updates a journey. While Hashcat and hashtopolis can be updated automatically, updating the agents requires an effort. You'll have to manually stop each agent, obtain the updated version from the server, and manually install it on each computer.

Running as a service

Elcomsoft Distributed Password Recovery agents do not require an authenticated user session to work. Agents can run as a system service, starting along with the system.

Hashcat can do the same in Linux. However, the Windows version requires an authenticated session to run hashtopolis Python scripts.

Non-ASCII characters and Unicode support

Hashcat supports non-ASCII and Unicode characters, but adding those to the attack requires a concise effort. If you want to add certain non-ASCII characters, you'll have to use the --hex-charset option and add the extra characters as HEX codes. If you want to use a dictionary with non-ASCII words, you'll have to do the same.

Elcomsoft Distributed Password Recovery is Unicode throughout.

CASE STUDY 1: BREAKING AN OFFICE 97/2003 DOCUMENT

So you have successfully installed Hashcat, git and Python. Assuming that the Python installer had the required modules, you're good to go and ready to launch an attack.

In the first case study, we'll try to break a document in the Office 97/2003 format (the ".doc" extension; the file might have been saved in "Compatibility mode" by a newer version of Microsoft Office). The file is protected with a 40-bit RC4 key.

Hashcat

After processing the document with office2hashcat.py, we've got the hash. Hashcat started the attack on the password; considering the (high) speed of the attack, we estimated the attack to complete in about 1 to 1.5 years.

Elcomsoft Distributed Password Recovery

Elcomsoft Distributed Password Recovery correctly recognized the file format and offered an option to brute-force the 40-bit encryption keys. The estimated recovery time was 48 hours using a CPU alone. Note that the same document can be recovered in a matter of minutes if you use the Thunder Tables attack in Advanced Office Password Breaker.

CASE STUDY 2: BREAKING ZIP PASSWORD

The second case study deals with a ZIP archive protected with AES-256 encryption.

Hashcat

Hashcat requires the use of a third-party tool to extract hashes from the target. This time around, you'll need zip2john, which is a part of the John the Ripper package.

When attacking ZIP encryption, a single small hash file is not enough. The last step of the attack calculates hash sum of the entire encrypted file. The resulting hash file extracted with zip2john is about 2MB. However, we could not make Hashcat to open the file. The tool had crashed with the following error:

Counted lines in c:\hashcat-6.1.1\z2.hash... Oversized line detected! Truncated 402236 bytes

We tried finding a solution (here and here), but the only kind of solution we found was this:

hashcat supports a data length of about 8 KB (compressed of course) for -m 13600 = Winzip

Unfortunately, for -m 13600 you need the whole data_buf (encrypted and compressed data) to verify if the password is correct.

(github)

We've been unable to launch the attack on the ZIP file with Hashcat.

Elcomsoft Distributed Password Recovery

Elcomsoft Distributed Password Recovery was able to open that ZIP archive and run the attack in a matter of seconds.

CASE STUDY 3: BREAKING VERACRYPT

In the third case study, we'll try breaking the password to a VeraCrypt container.

Hashcat

On paper, Hashcat had been offering support for VeraCrypt containers long before we did it in Elcomsoft Distributed Password Recovery, so we expect a well-ironed solution. We didn't have much trouble running the attack on the hash file extracted from the VeraCrypt container. In this regard, Hashcat is a fast and mature solution for breaking encrypted containers. What we did have a problem with was producing the required hash file to launch the attack.

If you are attacking an encrypted container (the file), all you need are the first 512 bytes of the file. In the case of full-disk encryption, you will also need to extract the 512 bytes; however, the extraction process is somewhat more complicated. The Hashcat Wiki has the following explanation:

1. *for a TrueCrypt boot volume (i.e. the computer starts with the TrueCrypt Boot Loader) you need to extract 512 bytes starting with offset 31744 (62 * 512 bytes). This is true for TrueCrypt 7.0 or later. For TrueCrypt versions before 7.0 there might be different offsets.*

   Explanation for this is that the volume header (which stores the hash info) is located at the last sector of the first track of the system drive. Since a track is usually 63 sectors long (1 sector is 512 bytes), the volume header is at sector 63 - 1 (62).

2. *if TrueCrypt uses a hidden partition, you need to skip the first 64K bytes (65536) and extract the next 512 bytes.*

   dd if=hashcat_ripemd160_AES_hidden.raw of=hashcat_ripemd160_AES_hidden.tc bs=1 skip=65536 count=512

3. *in all other cases (files, non-booting partitions) you need the first 512 Bytes of the file or partition.*

The same procedure should also work for VeraCrypt volumes (but you need to adapt the hash mode to -m 137XY - see the --help output for all the supported hash modes for VeraCrypt and the correct values for X and Y).

Additional troubles arise when you try to extract hashes from VeraCrypt volumes stored inside a forensic disk image. If this is the case, you may have to mount the disk image, calculate the address of the last sector of the first track on the disk, then extract the required binary data. We understand the theory but decided to skip this exercise due to lack of motivation.

Elcomsoft Distributed Password Recovery

It is exactly this kind of exercise we wanted to skip when developing Elcomsoft Distributed Password Recovery. In order to extract encryption metadata from one or several encrypted disks, you will need to open the physical device or forensic disk image in the included Elcomsoft Forensic Disk Decryptor (or boot the computer with Elcomsoft System Recovery). Simply ticking the required disk(s) in the GUI will do the trick.

CONCLUSION

As we already said in the beginning, Hashcat is a great tool and a strong competitor. Many of the differences are in the areas of usability and things actually working, let alone working out of the box. Where Elcomsoft Distributed Password Recovery just works the way you expect it to, you may have to spend a bit of time to make Hashcat work. These bits of time quickly accumulate, turning into multiple man-hours spent on setting up and configuring the distributed network, updating agents, distributing dictionaries across remote agents, finding and using hash extraction tools and overcoming the various obstacles, some of which we have described in this article.