CrackLord: Maximizing Computing Resources

BlackHat USA 2015

Michael McAtee, Manager, Crowe Horwath LLP
Lucas Morris, Senior Manager, Crowe Horwath LLP

Abstract

As IT security professionals, we have the need to crack various sizes of passwords on a regular basis. Furthermore, new methods, such as rainbow tables and general purpose graphics processor computing, have drastically changed the speed at which we can reverse the hashed or encrypted passwords that we recover during our testing.

As part of our password cracking efforts, we often had to deal with limited physical resources with multiple teams working to share the same systems. To aid in the management of resources available to our teams, we have put together an interface that wraps the various resources we utilize. Originally developed to manage password cracking, the system has been expanded to help manage available tasks which utilize heavy amounts of various resources, such as time, GPU, CPU or network.

Introduction

Passwords have become a ubiquitous method of providing security to data and access throughout the digital age. From almost the beginning, our technology has used various strings of characters or words to validate access. As the primary method of credential validation, passwords are involved in almost every test, audit, or improvement performed in technology. Furthermore, as the complexity of technology systems grows, the volume of data that must be processed during security projects has also increased. This has resulted in increased time and effort to process and parse these data-sets.

Thankfully, several changes have occurred over the last few years in the approach to cracking passwords. The accelerator pipeline in modern Graphics Processing Units (GPUs) can be harnessed for its massive computational power to create a stream processing paradigm for password hashes. As shown in the following chart, the number of guesses available for a common hashing algorithm, such as NTLM, is orders of a magnitude larger on a GPU in comparison to the same algorithm on a traditional processor:

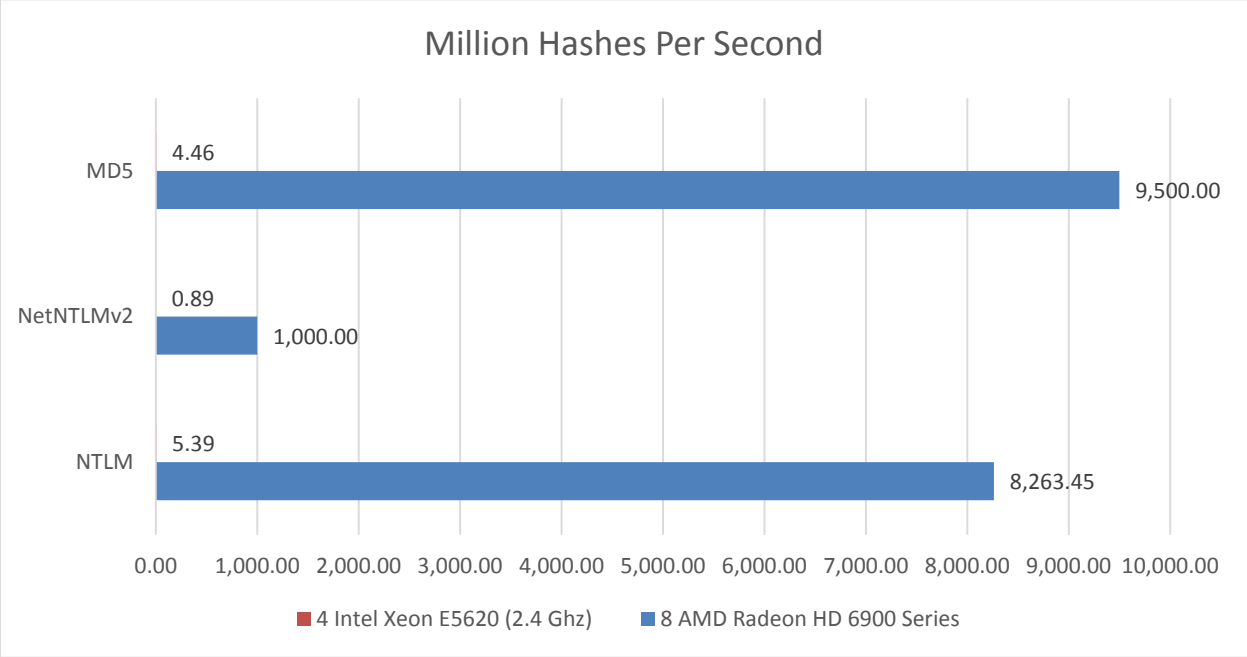

Figure 1: GPU versus CPU password cracking speeds on sample Crowe cracking hardware

Many organizations and individuals have built massive GPU password cracking systems and clusters as part of their security services. Furthermore, cloud based services, such as Amazon Web Services GPU instances, have also placed high performance cracking into the realm of affordability for anyone who may need access to it. Although the current tools do an amazing job providing heavy utilization for individual hardware, they have not kept pace with the need for sharing these resources amongst a team or group.

Resource Intensity

Over the past several years the world of password cracking has exploded with new tools and techniques. These new techniques have made it easier than ever to reverse captured password hashes. Based on our experience, within the past few years passwords have often become the first step into compromising the entire network. New techniques such as LLMNR/NetBIOS response have reduced the efficacy of pass the hash, again increasing the necessity of actually cracking the hashes. With the addition of powerful techniques, from GPU cracking to rainbow tables, it is easier than ever to access the plaintext.

Many of the most popular password cracking solutions were built initially for a specific need used by one individual. This holds true for many of the staple programs used by the security industry. While these tools have helped increase the efficiency of the individual, they often cause contention when used as teams. This is especially true when dedicated systems are used to run specific tools, such as a custom built GPU password cracking system.

The most common way to manage a dedicated shared resource such as this has been shared accounts and, if the environment is UNIX, screens. The first problem faced with this approach is how to manage when the resource is under contention. Can the current job be paused? What is the priority of this job versus another? Did the individual who started the running job include the correct flags or commands to make this resume-able? Can I run more than one of these jobs at the same time? The second problem faced is fully utilizing this new resource. For instance, if there is a lower priority job that needs to be run for a longer period of time, it is difficult to know when this can be started and make sure that someone gracefully pauses it.

In addition to managing a shared resource with one tool, adding more tools that may or may not be able to run concurrently can make managing the resource that much harder. Often times a dedicated GPU password cracking system will also have a decent number of CPUs. This could lead to a CPU bound task such as John the Ripper being run simultaneously with a GPU task, such as oclHashcat. This increases the complexity of determining who is running what job, what can and cannot be paused, and who has the higher priority.

These are the problems we experience firsthand and set out to resolve.

CrackLord – Distributed Job Management

CrackLord attempts to provide a scalable, pluggable, and distributed job tracking and management system. Better said, CrackLord is a way to load balance the resources, such as GPUs and CPUs, from multiple hardware systems and operating systems (Microsoft Windows and Linux) into a single queuing service. At the outset of the project, there were three primary use cases that drove the development of the service:

- **Group Job Management** – Whether a group of students, penetration testers, auditors, administrators, or security aficionados, we knew that direct access to resources wasn't a tenable solution in the long run. With numerous servers in our lab, the headaches having users SSH into systems and start a GNU screen with their name while hoping someone didn't accidentally stop their job, were constant.
- **Centralized Access to Distributed Resources –** Having multiple GPU systems and several exceptionally powerful servers, we wanted to provide the ability to put jobs into a single place and have the workload automatically divided between the hardware. Although we could have created a single cluster, we preferred to have more jobs running instead.

- **Access to Cloud Based Resources** – Although not put in use in our environment, we also saw a need for organizations and groups that may not have access to GPU hardware that may still have a need to utilize their performance. As such, we wanted to include the ability to add cloud based hardware into the system itself.

CrackLord uses various components across two major services, the resource and queue daemons to accomplish this task. The following diagram shows a high level overview of the high level design of the system:

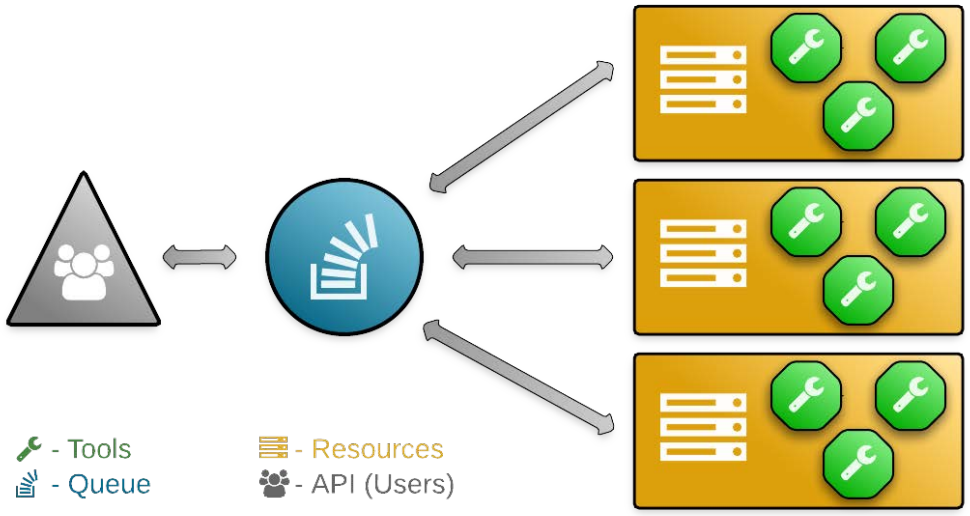

Figure 2: Overview of Design

Resource Daemon

The Resource is a service that runs on individual systems, providing access to their underlying hardware. Resources utilize various tools, such as Hashcat, John the Ripper, rcrack, NMap, NTDSExport, and others, to run jobs that use the local CPU, GPU, or network. There can be multiple resources in any system, with the queue automatically identify resources that are empty and don't have a currently running job.

Tool plugins are wrappers that work on a resource to run the work as required by tool. The implementation in the API, queue, resource, and default web GUI were designed in a generic manner so that almost any tool could be wrapped and included in CrackLord itself. The tool plugin will typically process the input from the user, store any necessary information, create working directory, and then run the tool in a shell on the resource server. It will endeavor to provide updates to the queue while running so a status can be provided to the end user. Once the process completes, the tool plugin will then process the output as well as do any local cleanup that is necessary.

Resources themselves are controlled by resource managers, which are another form of plugin. Managers provide an interface between the queue and whatever is controlling the resource. In the case of directly connected resources, the default plugin simply exposes a standard interface for the user to connect physical resources into the environment. The resource manager interface was created to allow hardware from several different providers to be used, from physical resources, to cloud resources such as Amazon Web Services GPU instances.

Queue Daemon

The Queue is a service that runs on a single system, providing an interface for users to submit jobs and manage connected resources. These jobs are then processed and sent to available Resources to perform the actual task. Users are able to create, pause, resume, and delete jobs in the Queue which will communicate with the Resource to handle the results. Finally, the system is designed to be extensible

providing standard interfaces and libraries allowing new tools, resource types, and management interfaces to be written and added as necessary.

The queue also has a standard API and GUI that come with the default queue daemon installation. The queue also includes a web server to host this GUI for the end user. Finally, the queue is also capable of handling authentication through both local users within a configuration file, or through Microsoft Active Directory.

System API

The API is hosted on the Queue at and must be accessed over HTTPS. All requests must include a token for authentication which is provided during the login process. In the case of GET requests, parameters will be expected within the query string.

Similar to the API itself, there are five object types exposed using a standard RESTful API:

- Tools
- Resource Managers
- Jobs
- Resources
- Queue

The following table shows the types of requests that are allowed in the API for each resource type:

	Query	Create	Read	Update	Delete
Tools	X		X		
ResourceManagers	X		X		
Jobs	X	X	X	X	X
Resources	X	X	X	X	X
Queue				X	

Figure 3: It should be noted, the queue object is only used for updating job order in the queue itself.

References

- Your Sexy Graphics Card Really Is A Great Way To Crack Passwords
  http://blog.scorpionsoft.com/blog/2009/04/your-sexy-graphics-card-really-is-a-great-way-to-crack-passwords.html

- 25-GPU cluster cracks every standard Windows password in <6 hours
  http://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/

- Hashcat
  http://hashcat.net

- John the Ripper
  http://www.openwall.com/john/

- Wikipedia
  https://en.wikipedia.org/wiki/Graphics_processing_unit

- Modern Password Cracking: A hands-on approach to creating an optimised and versatile attack. Chrysanthou Yiannis
  https://www.ma.rhul.ac.uk/static/techrep/2013/MA-2013-07.pdf

Additional Links

- Source Code
  https://github.com/jmmcatee/cracklord

- Detailed API Documentation
  https://github.com/jmmcatee/cracklord/wiki/API

Appendix: System Relationship Diagram