CHAMPLAIN COLLEGE | LCDi Leahy Center for Digital Investigation

Hash Cracking with Hashtopus

175 Lakeside Ave, Room 300A
Phone: (802) 865-5744
Fax: (802) 865-6446
http://www.lcdi.champlain.edu

4/17/2015

Hunter Gregal
Tyler Peyton
James Parfet

Disclaimer:

Contents

Introduction

Background:

There has been extensive research in the fields of hash cracking and password recovery. While modern cryptographic hash functions are much stronger than they used to be, so are the computers that attempt to crack them. One specific example of research done in distributed hash cracking is Andrew Zonenberg's article on cross-platform GPU-accelerated password recovery (Zonenberg, 2009). In the case of using Hashtopus as a tool for distributing hash keyspaces to be cracked, very little research has been done. Hashtopus was first presented in the Hashcat forum by its developer "Curlyboi", where it was modified and updated based on user feedback. The project is currently maintained on the developer's Github page: https://github.com/curlyboi/hashtopus

Purpose and Scope:

Cracking hashes is a fundamental principle of information security and even digital forensics. Often, when one imagines a cyber-threat in action, they imagine passwords being cracked. Hash-cracking also has a place in the prevention of attacks on information (such as with password audits) as well as in the process of catching potential suspects in a wide variety of crimes involving digital devices. The goal of this project is to test different distributed hash cracking methods against various encryption algorithms. We will examine the best methods in terms of efficiency and speed, and identify each method's pros and cons. The selling point of this research will be having valid and accurate data that has been obtained in a legal and controlled environment.

Research Questions:

What solutions are available for cracking hashes across a network?

What is the most effective way to crack hashes?

Is Hashtopus combined with oclHashcat a viable solution for distributed hash cracking?

Terminology

- **Agent** – A computer running the Hashtopus agent software with oclHashcat
- **Brute-Force Attack (cracking method) –** Tries all combinations from a given keyspace. It is the most basic form of attack.
- **Dictionary Attack (cracking method) –** An attack based on using a predefined list of potential passwords called a wordlist and checking each entry against the hash in question. This can be very efficient if you have a powerful wordlist.
- **Graphics Processing Unit (GPU) –** A programmable logic chip that renders images, animations, and video for the computer's screen. **GPUs** are located on plug-in cards, in a chipset on the motherboard or in the same chip as the CPU. They are commonly used for hash cracking because of their parallel processing ability.
- **Hash cracking –** The process of attempting to determine the plaintext equivalent of a hash value.

Terminology cont'd on Page 3 →

- **Hash value** – Also called a *message digest* or simply a *hash*, is a fixed length number generated from a piece of data like a document or string of text. The formula used to produce a hash value performs it in such a way that it is extremely unlikely that some other text will produce the same hash value. The hash value can be substantially smaller or bigger than the original data. Table 1: Hash Values gives an example of very different outputs.

Table 1: Hash Values

Input	Hash Value (MD5)
hello	5d41402abc4b2a76b9719d911017c592
hollo	181d1f65fc3edfc75945b24f22cd7e22
Hello	8b1a9953c4611296a827abf8c47804d7
hell0	73b43f17232b391b9123adf40c1b65dd

- **Hashing** – Producing *hash values* for data integrity or for security purposes.
- **Hashlist –** A list of passwords stored in hashed form using MD5, NTLM, SHA-1, SHA-512, or other hashing algorithm.
- **Hashtopus** – A multiplatform client-server tool to distribute oclHashcat/cudaHashcat tasks between multiple computers (agents).
- **Keyspace** – In cryptography, an algorithm's **key space** refers to the set of all possible combinations that can be used to generate a key, and is one of the most important attributes that determines the strength of a cryptosystem.
- **MD5 –** An algorithm that is used to verify data integrity through the creation of a 128-bit message digest from data input (which may be a message of any length) that is claimed to be as unique to that specific data as a fingerprint is to the specific individual.
- **NTLM –** The successor to the authentication protocol in Microsoft LAN Manager (LANMAN), an older Microsoft product, and attempts to provide backwards compatibility with LANMAN.
- **oclHashcat** – A GPU-based multi-hash cracker using a brute-force attack (implemented as mask attack), combination attack, dictionary attack, hybrid attack, and rule-based attack.
- **SHA-1 –** A 160-bit cryptographic hash function which resembles the earlier MD5 algorithm. It was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm. Cryptographic weaknesses were discovered in SHA-1, and the standard was no longer approved for most cryptographic uses after 2010.
- **SHA-2 –** A family of two similar hash functions, with different block sizes, known as *SHA-256* and *SHA-512*. They differ in the word size; SHA-256 uses 32-bit words and SHA-512 uses 64-bit words. There are also truncated versions of each standard, known as *SHA-224, SHA-384, SHA-512/224 and SHA-512/256*. These were also designed by the NSA.
- **Task –** An object within Hashtopus that describes all the parameters for a certain attack, including what flags to use, what files to attach, and what chunk of the keyspace to distribute.

Methodology and Methods

First, an in depth inventory will be made of all hardware being used for cracking the hashes, as well as the server hardware being used to host the distribution server.

With a complete inventory taken and documented, Hashtopus will be installed and configured on the server following the documentation in the manual hosted by the developer at http://hashtopus.nech.me/manual.html.

Next, we will begin adding agents to our hash cracking cluster. We will conduct tests with 1 agent computer, 5 agents, 10, and then 15; all attempting to crack the same hash or hash list. This will be done to test the correlation between the number of agents used and the efficiency of the cracking process. Following each test, we will record the hash or hash list being tested, the number of agents used, the speed of the attack, and the total elapsed time. In addition to fitting each agent computer with the Hashtopus agent file, a dedicated user account will be created with a script to run Hashtopus at login in order to streamline the process of connecting agents to the primary server.

The following cracking methods will be used to test the efficiency of the distributed cracking server:

Brute force attacks against 4 pre-determined hashed words on the MD5, NTLM, SHA-1, and SHA-512 hashing algorithms.

Dictionary attacks against a pre-determined list of hashed words on the MD5, NTLM, SHA-1, and SHA-512 hashing algorithms. The dictionary used will be a custom-made wordlist created for this project. The breakdown of ours is shown in Table 2: Wordlist. It is a combination of colors, names, and passwords from the Xato leak with usernames stripped and appended to the bottom of the list.

Table 2: Wordlist

Wordlist Name (file)	Author	Source (link)
10k most common	Mark Burnett	Source
Xato Leak	Mark Burnett	Source
other-names	Bob Baldwin (augmented by Matt Bishop & Daniel Klein)	Source
English_words	N/A	Source
colors-crayola	N/A	Source
TechnicalManualWords	N/A	Source
Given-Names	N/A	Source
colleges	N/A	Source
UltimatePasswordsList	N/A	Source

Equipment Used

For this project, both server hardware and workstation hardware will be used. See Table 3: Server Hardware for an outline of the server and its hardware specifications.

Table 3: Server Hardware

OS	Processor	Memory	Storage
Debian	1 core of 8 x 2.4GHz ESXi Server	4.2GB	HD1 16GB (OS) HD2 100GB (Database)

Aside from the server hosting the distribution software, agent workstations will be used to crack hashes and handle the computational load distributed by the server. Table 4: Agents gives an outline of each agent and its hardware specifications.

Table 4: Agents

Name	OS	Storage (MB)	Memory (RAM)	Processor*	GPU**	GPU Memory	GPU Clock
Agent-01	Win 7	953767	32 GB	Intel Core i7 950 @ 3.07GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-02	Win 7	953640	32 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-03	Win 7	953640	32 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-04	Win 7	953640	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-05	Win 7	953640	32 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-06	Win 7	953640	32 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-07	Win 7	953640	32 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-08	Win 7	953640	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-09	Win 7	953767	32 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-10	Win 7	953767	32 GB	Intel Core i7-2700K @ 3.50GHz	GTX 550 Ti	1024 MB	4.1 Gbps
Agent-11	Win 7	953640	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 650 Ti	1024 MB	5.4 Gbps
Agent-12	Win 7	953767	32 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-13	Win 7	953640	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 650 Ti	1024 MB	5.4 Gbps
Agent-14	Win 7	953640	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 650 Ti	1024 MB	5.4 Gbps
Agent-15	Win 7	953767	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-16	Win 7	953767	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 650 Ti	1024 MB	5.4 Gbps
Agent-17	Win 7	476710	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-18	Win 7	953639	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 660 Ti	2048 MB	6.0 Gbps
Agent-19	Win 7	476711	16 GB	Intel Core i7-3770K @ 3.50GHz	GTX 650 Ti	1024 MB	5.4 Gbps
Agent-20	Win 7	953767	8 GB	Intel Core i7 950 @ 3.07GHz	GTX 460 SE	1024 MB	1.7 Gbps

* = Max clock speed for processors corresponds with number after the '@' symbol in the "Processor" field.
** = All GPUs are NVIDIA GeForce brand.

Data Collection

Data collection will involve running tests against MD5 hashes, SHA1 hashes, NTLM hashes, and SHA256 hashes. For each hash we will use dictionary attacks and brute-force attacks. Finally, for each attack method we will attempt distributed attacks using 1 agent, 5 agents, 10 agents, and 15 agents; using digression based on estimated attack times.

Analysis

In this experiment, brute-force and dictionary attacks will be used against hashed words. These attacks will be carried out using a distributed client-server model utilizing the Hashtopus and oclHashcat programs. Each attack will be done using an increasing number of agents, which will help balance the processing load of the attack. It is our hypothesis that, with each additional agent aiding in the processing of the attack, the time to complete the attack will decrease. This is assuming that each additional agent will provide another Graphical Processing Unit (GPU) which will increase the speed at which hashes can be processed.

Results

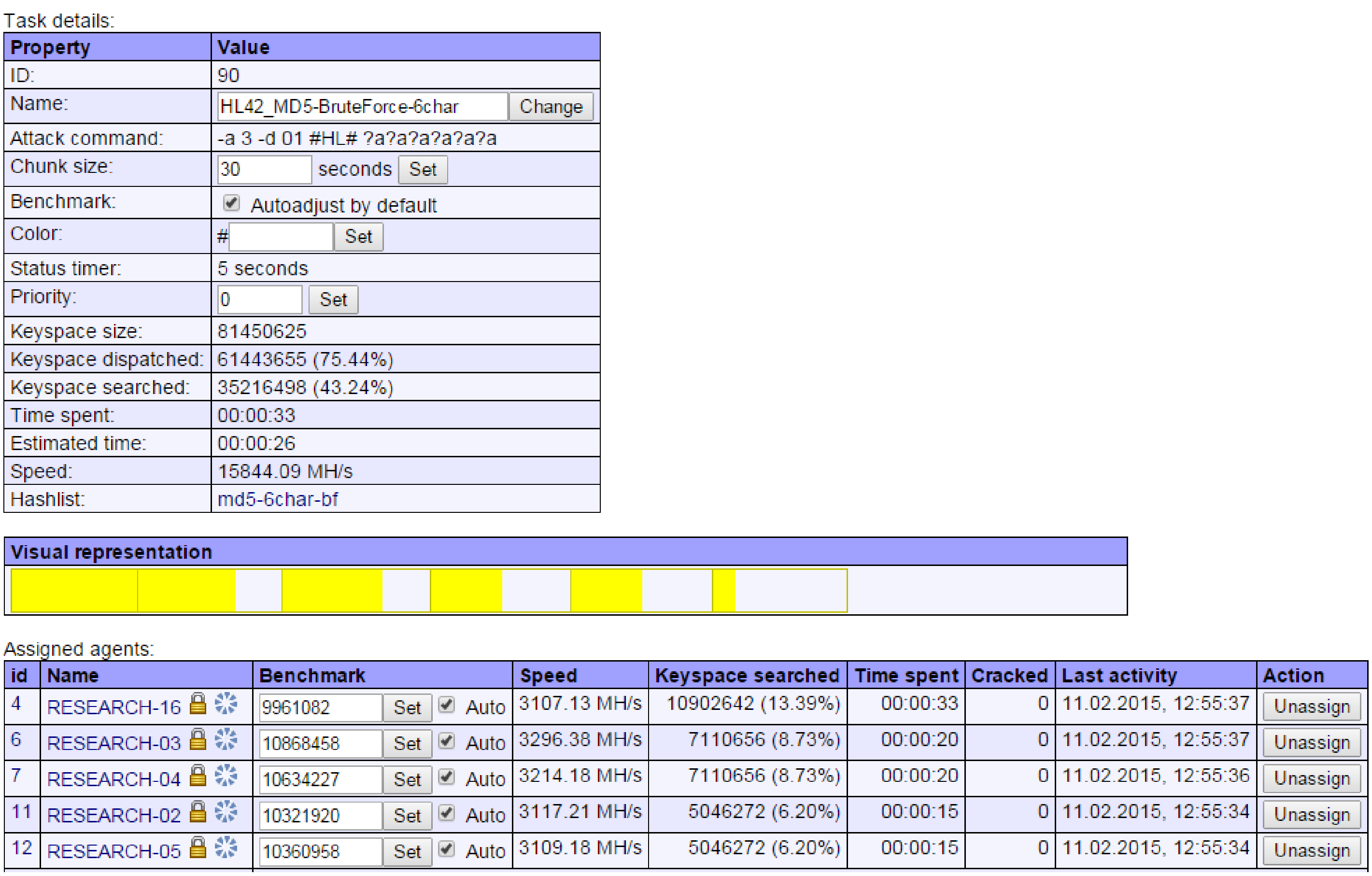
Brute-Force Attacks

The following tables represent the results obtained from Brute-Force attacks utilizing Hashtopus in conjunction with oclHashcat. Each table will contain the hash value being attacked, the plain text equivalent, the oclHashcat Flags used, the number of agents working on the task, the total speed that the agents can guess hashes in the keyspace (represented in million hashes per second), and the total elapsed time of the attack. Note that some tests were ignored due to extremely fast speeds, or due to ETA time constraints. Total Elapsed Times in red represent estimated times, and were not completed.

Table 5: MD5

Hash Value	Plain Text	oclHashcat Flags	Agents	Total Speed	Elapsed Time
a5c11712675394d71e 322d82abefb348	C4@mp	-a 3 -d -01 #HL# ?a?a?a?a?a	1	3200 MHs	00:00:01
			5	--	--
			10	--	--
			15	--	--
3696c7dc85658e9130 c4963623e02640	C4@mp!	-a 3 -d -01 #HL# ?a?a?a?a?a?a	1	3128 MHs	00:02:43
			5	15844 MHs	00:00:40
			10	25216 MHs	00:00:29
			15	16000 MHs	00:00:30
aee9fdd8b37c598159a 1821cc2db6777	#C4@mp!	-a 3 -d -01 #HL# ?a?a?a?a?a?a?a	1	--	--
			5	15935 MHs	00:47:31
			10	25327 MHs	00:35:00
			15	40000 MHs	00:21:50
723ed84d54e2c937ab dae835b80e0921	#C4@mp!0	-a 3 -d -01 #HL# ?a?a?a?a?a?a?a?a	1	--	--
			5	--	--
			10	32000 MHs	2d 09:27:09
			15	40000 MHs	2d 06:11:12

Figure 1: Hashtopus Interface – MD5 Test



Task details:

Property	Value
ID:	90
Name:	HL42_MD5-BruteForce-6char Change
Attack command:	-a 3 -d 01 #HL# ?a?a?a?a?a?a
Chunk size:	30 seconds Set
Benchmark:	Autoadjust by default
Color:	# Set
Status timer:	5 seconds
Priority:	0 Set
Keyspace size:	81450625
Keyspace dispatched:	61443655 (75.44%)
Keyspace searched:	35216498 (43.24%)
Time spent:	00:00:33
Estimated time:	00:00:26
Speed:	15844.09 MH/s
Hashlist:	md5-6char-bf

Visual representation

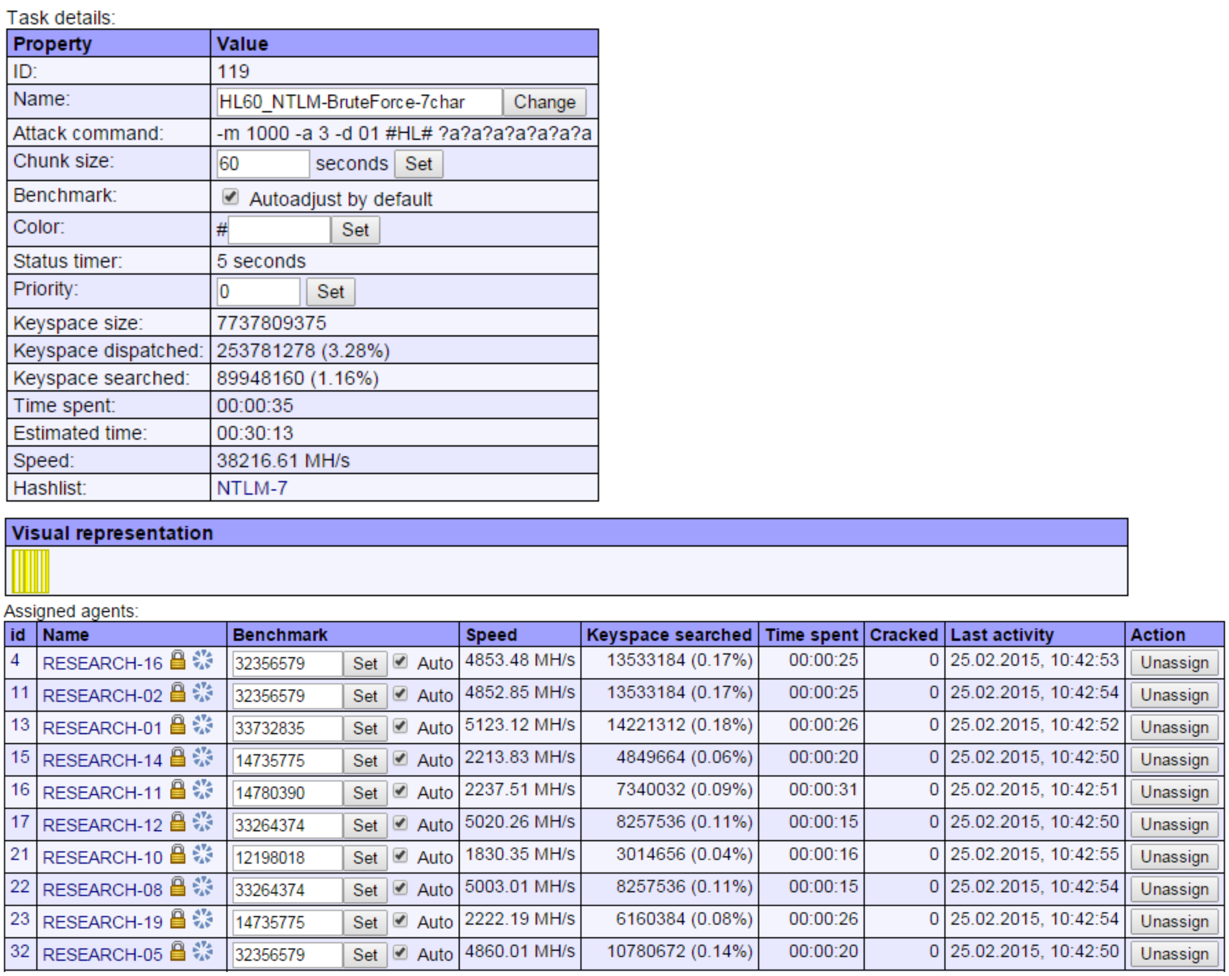
Assigned agents:

id	Name	Benchmark	Speed	Keyspace searched	Time spent	Cracked	Last activity	Action
4	RESEARCH-16	9961082 Set Auto	3107.13 MH/s	10902642 (13.39%)	00:00:33	0	11.02.2015, 12:55:37	Unassign
6	RESEARCH-03	10868458 Set Auto	3296.38 MH/s	7110656 (8.73%)	00:00:20	0	11.02.2015, 12:55:37	Unassign
7	RESEARCH-04	10634227 Set Auto	3214.18 MH/s	7110656 (8.73%)	00:00:20	0	11.02.2015, 12:55:36	Unassign
11	RESEARCH-02	10321920 Set Auto	3117.21 MH/s	5046272 (6.20%)	00:00:15	0	11.02.2015, 12:55:34	Unassign
12	RESEARCH-05	10360958 Set Auto	3109.18 MH/s	5046272 (6.20%)	00:00:15	0	11.02.2015, 12:55:34	Unassign

Table 6: NTLM

Hash Value	Plain Text	oclHashcat Flags	Agents	Total Speed	Elapsed Time
5334d05aa04ceead7e8 336da47f2adee	C4@mp	-m 1000 -a 3 -d 01 #HL# ?a?a?a?a?a	1	5000 MHs	00:00:04
			5	--	--
			10	--	--
			15	--	--
73be8d998d36e717c6 be396c2703277c	C4@mp!	-m 1000 -a 3 -d 01 #HL# ?a?a?a?a?a?a	1	5000 MHs	00:00:03
			5	--	--
			10	--	--
			15	--	--
28f28ca8be1bb08528b b78ac9cbaa157	#C4@mp!	-m 1000 -a 3 -d 01 #HL# ?a?a?a?a?a?a?a	1	5100 MHs	00:04:30
			5	24844 MHs	00:01:27
			10	38000 MHs	00:01:20
			15	--	--
26b6472b5c891a1bd2f 1ed0475b6cde3	#C4@mp!0	-m 1000 -a 3 -d 01 #HL# ?a?a?a?a?a?a?a?a	1	--	--
			5	--	--
			10	35000 MHs	2d 04:15:00
			15	--	--

Figure 2: Hashtopus Interface – NTLM Test


Task details:

Property	Value
ID:	119
Name:	HL60_NTLM-BruteForce-7char Change
Attack command:	-m 1000 -a 3 -d 01 #HL# ?a?a?a?a?a?a?a
Chunk size:	60 seconds Set
Benchmark:	Autoadjust by default
Color:	# Set
Status timer:	5 seconds
Priority:	0 Set
Keyspace size:	7737809375
Keyspace dispatched:	253781278 (3.28%)
Keyspace searched:	89948160 (1.16%)
Time spent:	00:00:35
Estimated time:	00:30:13
Speed:	38216.61 MH/s
Hashlist:	NTLM-7

Visual representation

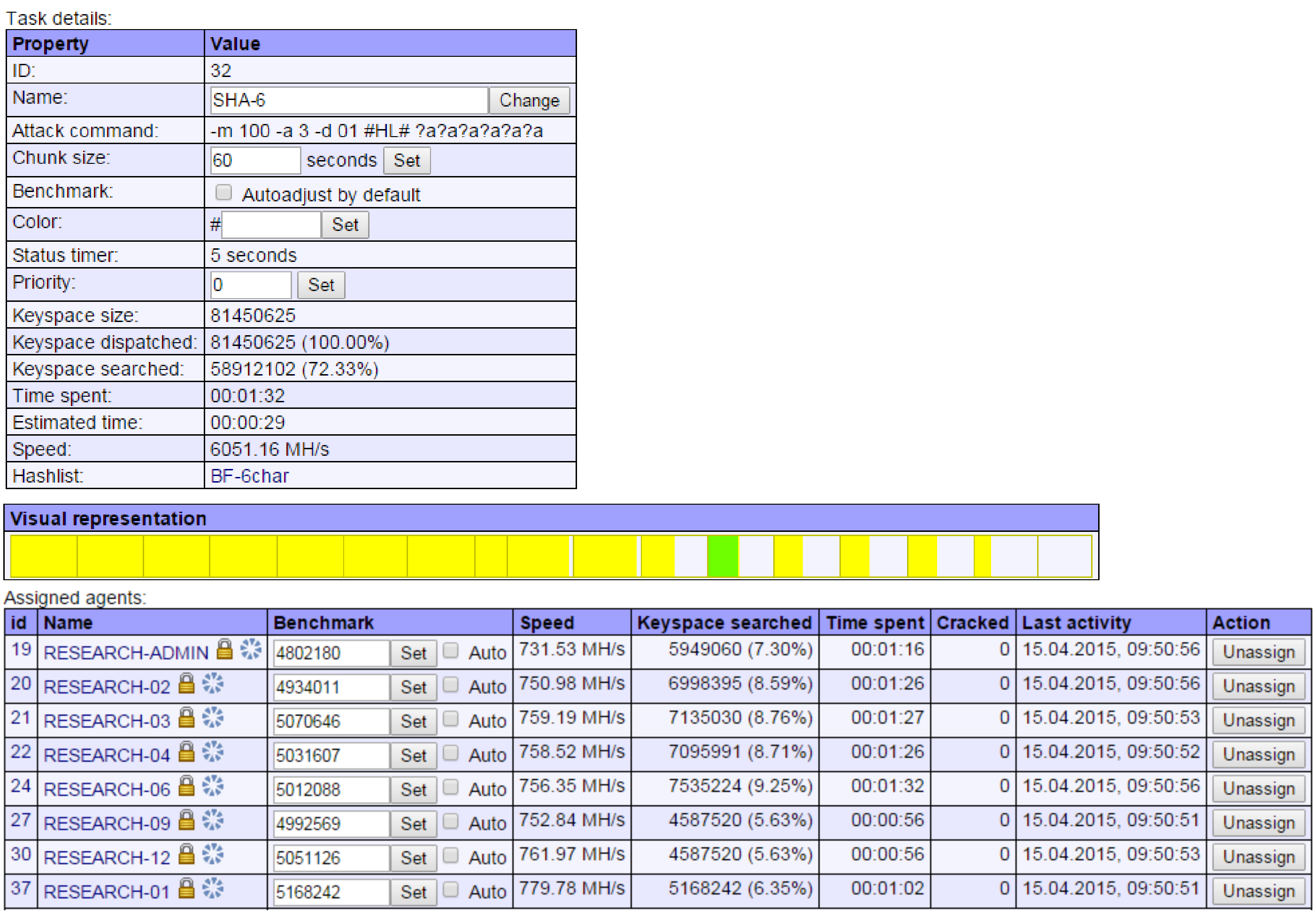
Assigned agents:

id	Name	Benchmark	Speed	Keyspace searched	Time spent	Cracked	Last activity	Action
4	RESEARCH-16	32356579 Set Auto	4853.48 MH/s	13533184 (0.17%)	00:00:25	0	25.02.2015, 10:42:53	Unassign
11	RESEARCH-02	32356579 Set Auto	4852.85 MH/s	13533184 (0.17%)	00:00:25	0	25.02.2015, 10:42:54	Unassign
13	RESEARCH-01	33732835 Set Auto	5123.12 MH/s	14221312 (0.18%)	00:00:26	0	25.02.2015, 10:42:52	Unassign
15	RESEARCH-14	14735775 Set Auto	2213.83 MH/s	4849664 (0.06%)	00:00:20	0	25.02.2015, 10:42:50	Unassign
16	RESEARCH-11	14780390 Set Auto	2237.51 MH/s	7340032 (0.09%)	00:00:31	0	25.02.2015, 10:42:51	Unassign
17	RESEARCH-12	33264374 Set Auto	5020.26 MH/s	8257536 (0.11%)	00:00:15	0	25.02.2015, 10:42:50	Unassign
21	RESEARCH-10	12198018 Set Auto	1830.35 MH/s	3014656 (0.04%)	00:00:16	0	25.02.2015, 10:42:55	Unassign
22	RESEARCH-08	33264374 Set Auto	5003.01 MH/s	8257536 (0.11%)	00:00:15	0	25.02.2015, 10:42:54	Unassign
23	RESEARCH-19	14735775 Set Auto	2222.19 MH/s	6160384 (0.08%)	00:00:26	0	25.02.2015, 10:42:54	Unassign
32	RESEARCH-05	32356579 Set Auto	4860.01 MH/s	10780672 (0.14%)	00:00:20	0	25.02.2015, 10:42:50	Unassign

Table 7: SHA-1

Hash	Plain Text	oclHashcat Flags	Agents	Total Speed	Elapsed Time
9427c670fdf02a45e4c51c 0abc65cf51e7c28e7f	C4@mp	-m 100 -a 3 -d 01 #HL# ?a?a?a?a?a	1	740 MHs	00:00:02
			5	--	--
			10	--	--
			15	--	--
b7609ae22bd12cb29c313 9b0e4e19334142231dd	C4@mp!	-m 100 -a 3 -d 01 #HL# ?a?a?a?a?a?a	1	740 MHs	00:11:21
			5	3740 MHs	00:03:48
			10	7150 MHs	00:01:32
			15	--	--
5b56b28006c2b676e9107 e9f0a7f6ad2eb57f949	#C4@mp!	-m 100 -a 3 -d 01 #HL# ?a?a?a?a?a?a?a	1	676 MHs	1d 04:31:57
			5	3650 MHs	05:13:54
			10	7100 MHs	02:40:00
			15	9220 MHs	02:00:00
307a8ba374d757b9f4469 bc00623a0d9ee76520a	#C4@mp!0	-m 100 -a 3 -d 01 #HL# ?a?a?a?a?a?a?a?a	1	--	--
			5	--	--
			10	6700 MHs	9d 23:24:29
			15	--	--

Figure 3: Hashtopus Interface – SHA-1 Test


Task details:

Property	Value
ID:	32
Name:	SHA-6 Change
Attack command:	-m 100 -a 3 -d 01 #HL# ?a?a?a?a?a?a
Chunk size:	60 seconds Set
Benchmark:	Autoadjust by default
Color:	# Set
Status timer:	5 seconds
Priority:	0 Set
Keyspace size:	81450625
Keyspace dispatched:	81450625 (100.00%)
Keyspace searched:	58912102 (72.33%)
Time spent:	00:01:32
Estimated time:	00:00:29
Speed:	6051.16 MH/s
Hashlist:	BF-6char

Visual representation

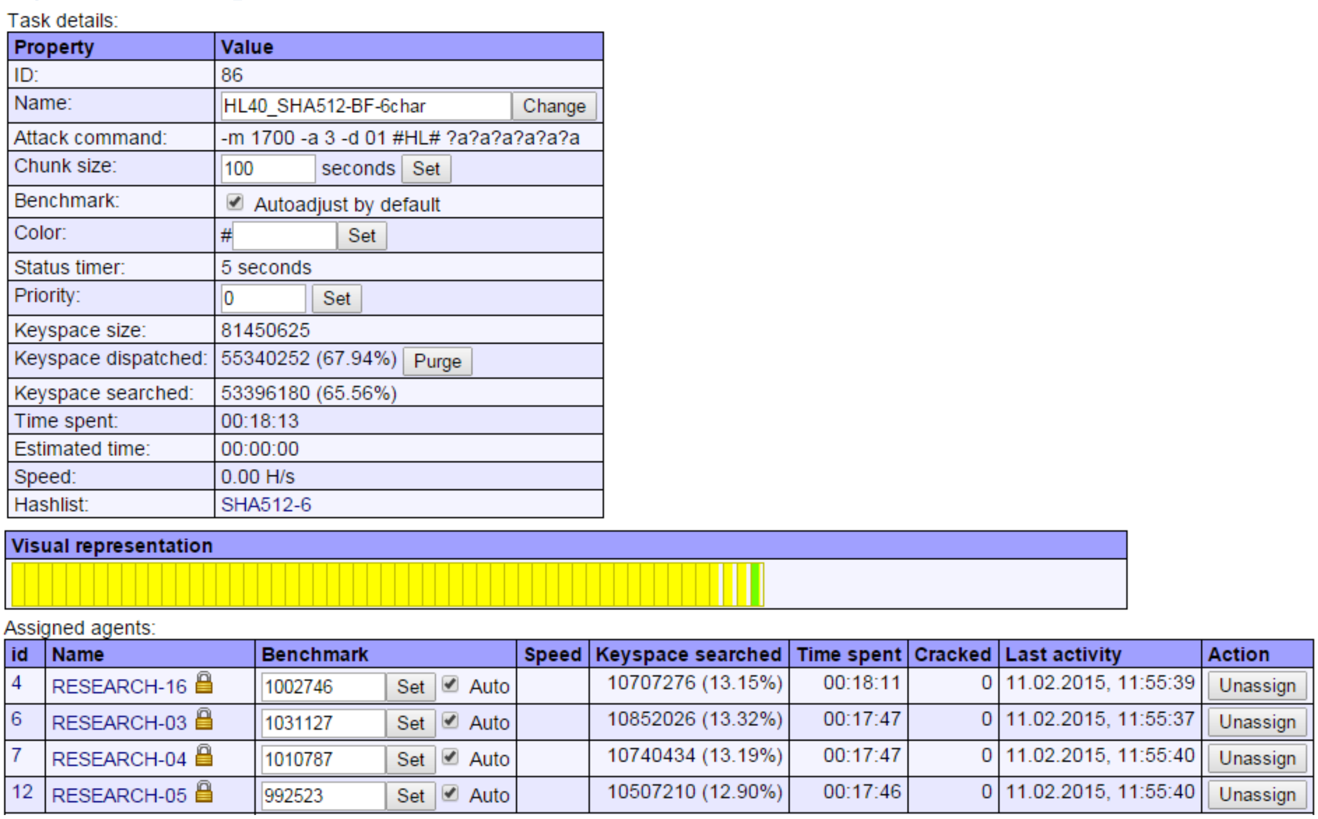
Assigned agents:

id	Name	Benchmark	Speed	Keyspace searched	Time spent	Cracked	Last activity	Action
19	RESEARCH-ADMIN	4802180 Set Auto	731.53 MH/s	5949060 (7.30%)	00:01:16	0	15.04.2015, 09:50:56	Unassign
20	RESEARCH-02	4934011 Set Auto	750.98 MH/s	6998395 (8.59%)	00:01:26	0	15.04.2015, 09:50:56	Unassign
21	RESEARCH-03	5070646 Set Auto	759.19 MH/s	7135030 (8.76%)	00:01:27	0	15.04.2015, 09:50:53	Unassign
22	RESEARCH-04	5031607 Set Auto	758.52 MH/s	7095991 (8.71%)	00:01:26	0	15.04.2015, 09:50:52	Unassign
24	RESEARCH-06	5012088 Set Auto	756.35 MH/s	7535224 (9.25%)	00:01:32	0	15.04.2015, 09:50:56	Unassign
27	RESEARCH-09	4992569 Set Auto	752.84 MH/s	4587520 (5.63%)	00:00:56	0	15.04.2015, 09:50:51	Unassign
30	RESEARCH-12	5051126 Set Auto	761.97 MH/s	4587520 (5.63%)	00:00:56	0	15.04.2015, 09:50:53	Unassign
37	RESEARCH-01	5168242 Set Auto	779.78 MH/s	5168242 (6.35%)	00:01:02	0	15.04.2015, 09:50:51	Unassign

Table 8: SHA-512

Hash	Plain Text	oclHashcat Flags	Agents	Total Speed	Elapsed Time
10dfb6aa8830a407588f50d37cc340d47436d4bfa27795333ecc4b6ff08fcbb3ca685fce25c529ea4be2ec19723046a0f921b768133a0c2acc14a3997d6dbe90	C4@mp	-m 1700 -a 3 -d 01 #HL# ?a?a?a?a?a	1	92000 kHs	00:00:06
			5	--	--
			10	--	--
			15	--	--
8aa560fda8753952616d59cc11e7c440e61d9cd793e38aace64097a9f278a398812e1450797aca51bce90c4b7bad658ad5cbccac1fe5034e74da54aba145ce33	C4@mp!	-m 1700 -a 3 -d 01 #HL# ?a?a?a?a?a?a	1	92000 kHs	02:15:00
			5	460 MHs	00:18:13
			10	710 MHs	00:12:47
			15	975 MHs	00:08:32
40f799ca5f5742379d40f5ef972ace54377962c5096f0d7b234078b90bc8f8ed6580db991a1a8213a88ce38cfa6d736c4c8b8fe5917da3664b878659dc222d3c	#C4@mp!	-m 1700 -a 3 -d 01 #HL# ?a?a?a?a?a?a?a	1	--	--
			5	--	--
			10	500 MHs	2d 04:52:26
			15	--	--
1e03ee9ac55cff933504fafce6532f4adc6d75d1ddb427c8419256f98dbf5ee57b8e1952467dbe422ff19751141c1b8b2e7cb0035bbe77a53f527b2cfbe7d312	#C4@mp!0	-m 1700 -a 3 -d 01 #HL# ?a?a?a?a?a?a?a?a	1	--	--
			5	--	--
			10	790 MHs	97d 14:00:05
			15	--	--

Figure 4: Hashtopus Interface – SHA-512 Test


Task details:

Property	Value
ID:	86
Name:	HL40_SHA512-BF-6char Change
Attack command:	-m 1700 -a 3 -d 01 #HL# ?a?a?a?a?a?a
Chunk size:	100 seconds Set
Benchmark:	Autoadjust by default
Color:	# Set
Status timer:	5 seconds
Priority:	0 Set
Keyspace size:	81450625
Keyspace dispatched:	55340252 (67.94%) Purge
Keyspace searched:	53396180 (65.56%)
Time spent:	00:18:13
Estimated time:	00:00:00
Speed:	0.00 H/s
Hashlist:	SHA512-6

Visual representation

Assigned agents:

id	Name	Benchmark	Speed	Keyspace searched	Time spent	Cracked	Last activity	Action
4	RESEARCH-16	1002746 Set Auto		10707276 (13.15%)	00:18:11	0	11.02.2015, 11:55:39	Unassign
6	RESEARCH-03	1031127 Set Auto		10852026 (13.32%)	00:17:47	0	11.02.2015, 11:55:37	Unassign
7	RESEARCH-04	1010787 Set Auto		10740434 (13.19%)	00:17:47	0	11.02.2015, 11:55:40	Unassign
12	RESEARCH-05	992523 Set Auto		10507210 (12.90%)	00:17:46	0	11.02.2015, 11:55:40	Unassign

Dictionary Attacks

Table 9: Dictionary Attacks represents the results obtained from dictionary attacks utilizing Hashtopus in conjunction with oclHashcat. The table will contain the type of hash list being attacked, the oclHashcat Flags used, the number of agents working on the task, the total speed that the agents can guess hashes in the keyspace (represented in million hashes per second), and the total elapsed time of the attack. Note that due to the speed of the dictionary attack, only one agent was required to reach reasonable speeds.

Table 9: Dictionary Attacks

Hash List	oclHashcat Flags	Agents	Total Speed	Elapsed Time	% Hashes Cracked / Total
MD5	-m 0 -a 0 -d 01 #HL# wordlist.txt	1	1422 H/s	00:01:42	28.85% (41863 / 145102)
NTLM	-m 1000 -a 0 -d 01 #HL# wordlist.txt	1	1450 H/s	00:01:40	28.85% (41863 / 145102)
SHA-1	-a 0 -d 01 #HL# wordlist.txt	1	1370 H/s	00:01:46	28.85% (41863 / 145102)
SHA-512	-a 0 -d 01 #HL# wordlist.txt	1	820 H/s	00:02:57	28.85% (41863 / 145102)

Conclusion

The primary goal of this project was to determine what solutions are available for cracking hashes while utilizing multiple machines as processing units. We discovered that many solutions for distributed hash cracking currently exist, all using different tools. For this project we decided to focus on the open-source tool called Hashtopus, stacked with oclHashcat as our cracking engine. We also wanted to try to isolate the most effective way to crack hashes. While we learned this is highly situational, it appears that it is generally most efficient to use a powerful dictionary-based attack. However, utilizing a dictionary for cracking hashes limits the size of the keyspace to the content of the wordlist. Brute-force attacks offer the entire keyspace, but require much more time and processing power. As we discovered in tests of longer passwords, these times can approach an unrealistic length. Finally, we determined that using Hashtopus as a distribution method of keyspaces, combined with oclHashcat, is an extremely effective method of cracking hashes and hash lists across a network utilizing many processing agents. According to our data, each additional agent working on the keyspace greatly increases the speed at which the hash can be guessed and decreases the overall time it takes to crack a hash.

Further Work

The continued maintenance and issues that arose with Hashtopus limited the variety of attack methods we used. By conducting tests with 15 machines, we reached a significant benchmark for computation power. By using a more efficient attack like a mask attack, versus a traditional brute-force attack, we could focus more power into cracking hashes. In addition, there are many other tools and options that can be tested aside from Hashtopus and oclHashcat. One tool known as CryptoHaze utilizes the concept of Rainbow Tables to distribute the keyspace across multiple agents. Further work could be done testing this alternative method of cracking hashes.

References

Curlyboi. Hashtopus - distributed solution. (n.d.). Retrieved April 16, 2015, from http://hashtopus.nech.me/

N.A. Hashcat advanced password recovery. (n.d.). Retrieved April 16, 2015, from http://hashcat.net/wiki/

N.A. Secure Hash Algorithm. (n.d.). Retrieved April 16, 2015, from http://en.wikipedia.org/wiki/Secure_Hash_Algorithm

Zonenberg, A. (2009). Distributed Hash Cracker: A Cross-Platform GPU-Accelerated Password Recovery System. Retrieved April 15, 2015, from http://colossus.cs.rpi.edu/~azonenberg/papers/cracker.pdf