Case Study | 2016 | RELEASE 1.0

Cascades:

The Anonymous Hack of HBGary
(Part B)

This case study was created as part of the Open Technology Institute's (OTI) effort to create a curriculum focused on how digital technology is transforming public policy and governance. It is intended for use in a classroom setting.

CASCADES

On Saturday, February 5, 2011, Aaron Barr, the CEO of HBGary Federal, a company that sold IT security services to the federal government, seemed to be in an excellent position.[1] The previous day, the *Financial Times* had run a story about his use of social media to track down the leaders of Anonymous, a powerful network of hacktivists that had recently come under scrutiny from law enforcement officials for its involvement in (among other things) attacks targeting the websites of financial firms that had stopped processing donations to WikiLeaks.[2] The FBI had reached out to Barr about his research, and the parties planned to meet on Monday. Barr thought the meeting, the publicity (the *Daily Kos* had also picked up the story), and a presentation he planned to give about his research at an upcoming technology security conference would give his cash-strapped firm a much-needed boost. "[The] story," he had written in an e-mail to colleagues on Friday, "is really taking shape."[3]

At the same time, a collection of Anonymous hacktivists was initiating an attack against HBGary Inc., HBGary Federal, and Barr himself.[4] The effort had begun the previous day when four hacktivists had connected in a chat room. They had read the coverage in *The Financial Times*, and they were concerned. For one thing, they had questions about the accuracy of Barr's claims: he had said that he had identified Anonymous's leaders, which did not seem plausible because Anonymous depicted itself as "non-hierarchical."[5] What's more, Barr had specific names, which meant that some innocent people might be falsely accused.[6] Finally, the collective was on alert following the arrest in late January of five Anonymous members in England and the execution of 40 search warrants targeting Anonymous members in the United States. As Joseph Menn, the reporter who authored *The Financial Times* story, explained, they were trying to figure out whether Barr "actually knew anything or not." Or as Gabriella Coleman, a cultural anthropologist at McGill University and the author of a recent book on Anonymous, said, the hacktivists were "ready to rumble."[7, 8]

By Sunday evening, the hacktivists had gained access to tens of thousands of HBGary's e-mails and documents; defaced the company's website; accessed Barr's Twitter account; and gained access to rootkit.com, a website created by HBGary, Inc. founder Greg Hoglund.[9] All of this begged a very simple question: How did a company that was supposed to be at the forefront of technology security get hacked so easily?

Three Dominos

The hacktivists began by gathering data on Barr and trying to find a loophole in HBGary Federal's defenses. They soon found one in the form of an SQL injection.[10]

Domino One – SQL Injection[11]

SQL is an acronym for the Standard Query Language. This is a language that is used to extract information from databases. A database is itself a collection of information, organized around a record. Each record holds the same kind of information about some entity. Think of a record

as a row in a spreadsheet, where the individual columns store the same kind of data about the entity represented by the row. For a personnel record, each row might represent a single employee, with columns for employee number, name, address, date of hire, and the like. The same information is gathered for all employees. The collection of all of the employee records would be a single database table, and the full database might have a number of different tables. For example, along with the employee table, there could be a table that contained various job levels and associated pay rates, or a table that contained employee numbers, healthcare options, and other benefits.

The standard query language allows extraction and manipulation of data from a database. Using SQL, one could ask for all of the employee records for the company, or all of those employees that are under 20 years of age. The query for the first of these might look something like

> Select * from personnel_table

while the second might look like

> Select * from personnel_table where DOB > 1/1/1995

Both of these say to return all of the information for the selected records from the personnel_table, but the second also says to make sure that another condition is satisfied, namely that the record returned will have a value in the DOB field (assuming there is such a field that stores the date

the statements be separated by some character (like ";") that shows where one statement ends and the other begins.

This ability to string multiple SQL statements together is the root cause of an SQL injection attack. There are applications that will take user input, construct an SQL query out of the input, and submit the query. For example, an employee telephone directory might ask the user for a name of the person whose directory information is desired, and construct a query like

> Select name, number from personnel_file where name contains XXX

where "XXX" is whatever the user entered.

If the application does not check the user input, however, the user can input not just the name of the person being looked for, but the name, a separator, and then any SQL command that the user wants to run. So if the phone number was held in the same table as other employee data, and the input was not checked, a user could look for the name

> Waldo; Select name, salary from personnel_file

and get not only the name and phone number for all employees with the name Waldo, but the name and salary for all employees. Other commands could tell what other tables are stored in the database. In short, the ability to add other queries opens systems that are victims of this attack to leaking all of the information contained in the database.

In the case of HBGary, the content management open to this sort of attack. While most of the content that was contained in the database was harmless, one table contained the names of the users of the system and the hashes of their passwords. This was the first domino to fall in the attack.

Domino Two – Passwords

When Anonymous was able to extract the password table from the HBGary content management system via an SQL injection attack, they did not get the names and passwords of those who used those systems. No modern system will store passwords in the clear (that is, in a fashion that could be read by a human); instead, these systems will store a cryptographic hash of the password that corresponds to the user names.

Cryptographic hashing is a mechanism that transforms a set of characters into a different set of characters in a fashion that is easy to do, but very difficult to reverse. A good hashing function will take an input (say, a string of characters) and produce a number of a particular length. How long the input string is has no effect on how long the number produced is. If the hash function is a secure one, there will be no relation between the numbers produced by two strings by the hash function. Change one character in the input string, and the resulting hash will differ from the original as much as if you used a completely different string.

Functions like this are examples of one-way functions. Given a particular password, the hash of the password is reasonably easy to calculate. But given the hash, it is very hard to calculate the password that produced that hash, even if you know the function that was used to produce the hash. A password file will store your user id and the hash of a password you have chosen.

When you are asked for your password, the password that you enter is hashed, and the hash is sent to the password checker. If the hash matches the stored hash, you are allowed in; otherwise you entered the wrong password.

This is an elegant and efficient solution for storing passwords. Computers can compare the hash of passwords more rapidly than they could compare the passwords themselves. Storage costs are known, since the size of a password hash is always the same. Since the password itself is never transmitted, the overall system is more secure.

The only way to find the password that corresponds to a hash is to use brute force—that is, to try every combination of characters that might comprise the password, run the hash function on that combination of characters, and see if the resulting hash is the same as the one that you have found. If you start with the assumption that the password is an English word, then you will have between 170,000 (the number of entries in the Oxford English Dictionary) and 1,000,000 words (the rough estimate of English words from the Global Language Monitor) to run through the hash function.[12] If the hash function is complicated and takes one second to produce a hash from a string, then it will take between 28 and 277 hours to build a table of all of the hashes for words in English. But if your hash function is faster (say, being able to produce 1,000 hashes in a second), then it will take somewhere between 15 minutes and three hours to build such a table. In fact, with modern chips, building a table that maps all of the MD5 hashes of words in the Oxford English

But that assumes that you have to build such a table yourself. These days, there are lots of these tables available on the Internet. Known as rainbow tables, these mappings from hashes to passwords are shared by hacktivists (both those who test vulnerabilities as a hobby and those who do so for a living) with others. There are even websites (such as one attributed to HaQ4U) that allow you to enter an MD5 hash and will return the plain text that results in that hash.[13, 14]

There are a number of ways to avoid having the hashes of a password cracked by rainbow tables. One way is to have a password that is not a common word (which is why many organizations will not allow dictionary words as passwords). Another is a technique called salting; this involves adding data that is an additional input into the hash function, so that the hash depends on both the value of this data and the value of the password. Either of these will make the number of possible password combinations much larger than the number of words in the dictionary, and will foil the de-cyphering of the password by the use of rainbow tables.

Unfortunately, the password table exposed by the SQL injection of the HBGary web site used a fairly simple hashing function, did not use salting on that function, and some of the passwords were simple dictionary words. One of those whose password could be easily cracked because of this was the CEO of HBGary Federal.

The Final Dominos – Repeated Passwords and Human Error

Barr and Ted Vera, HBGary Federal's COO, used the same simple password for multiple accounts (e.g., Twitter, LinkedIn, and e-mail). What's more, Barr was the administrator for the firm's e-mail. This enabled the hacktivists to get into Hoglund's e-mail account, which led to the root password to rootkit.com (a site that Hoglund had founded) and Jussi, a security specialist at Nokia who managed access to the site. In e-mail correspondence with Jussi, the hacktivists posed as Hoglund and pretended not to know his username or his password, but the hacktivists convinced Jussi to give them the information. To Peter Bright, a reporter who authored an in-depth piece on how Anonymous hacked HBGary, this was yet another example of the company's poor defenses.[15]

But according to Amadeus Stevenson, the Chief Technology Officer for Decoded, a company that provides digital literacy programs for non-technical staff in organizations, this also illustrates the importance of human and cultural tendencies in IT security. The hacktivists used a number of tricks—including mimicking Hoglund's writing style; providing bona fide information they had found in the email archives to gain credibility; and emphasizing that they were in a rush, which made it seem to the employee that his boss needed something quickly and therefore placed pressure on the employee. The end result was that Jussi gave the hacktivists something he simply should not have: access to the Rootkit server. As Stevenson explained, they used some sophisticated psychological techniques to convince him to do so. "We all want to help somebody who's mistaken," he elaborated, "... There's something about the way human brains are wired - we want to help someone when we know the answer and they don't ... especially when it's our boss. They [hacktivists] used that leverage to exert extra pressure to get a better result."[16]

And the results were indeed calamitous. By the evening of Sunday, February 6, the hacktivists had taken over Barr's Twitter account (from which they posted profane material); posted all of Barr's e-mails online; and defaced the HBGary Federal homepage by placing on it a message announcing their presence. It read:

> This domain has been seized by Anonymous under section #14 of the Rules of the Internet. Greetings HBGary (a computer 'security' company). You recent claims of 'infiltrating' Anonymous amuse us, and so do your attempts at using Anonymous as a means to garner press attention for yourself. How's this for attention? You've tried to bite at the Anonymous hand, and now the Anonymous hand is bitch-slapping you in the face.[17, 18]

Index

1. For additional details, see Parmy Olson, *We Are Anonymous: Inside the Hacker World of LulzSec, Anonymous, and the Global Cyber Insurgency*, Little Brown and Company: New York, 2012, p. 14.

2. Joseph Menn, "Cyberactivists Warned of Arrest," *The Financial Times*, February 5, 2011, available at http://www.ft.com/intl/cms/s/0/87dc140e-3099-11e0-9de3-00144feabdc0.html#axzz3ksQm5IRg (accessed on December 1, 2015).

3. Nate Anderson, "How One Man Tracked Down Anonymous—and Paid A Heavy Price," *ArsTechnica*, February 9, 2011, available at http://arstechnica.com/tech-policy/2011/02/how-one-security-firm-tracked-anonymousand-paid-a-heavy-price/ (accessed on December 1, 2015); and Olson, *We Are Anonymous*, p. 9.

4. HBGary Federal was a spinoff of HBGary, Inc. The latter owned 10% of the former. "HBGary Launches HBGary Federal," *Forensic Focus*, December 9, 2009, available at http://www.forensicfocus.com/index.php?name=News&file=article&sid=1314 (accessed on December 6, 2015); and Olson, *We Are Anonymous*, p. 5.

5. David Kushner, "The Masked Avengers," *The New Yorker*, September 8, 2014, available at http://www.newyorker.com/magazine/2014/09/08/masked-avengers (accessed on September 30, 2015).

6. Olson, *We Are Anonymous*, pp. 10-12; and Anderson, "How One Man...."

7. Menn, "Cyberactivists Warned...,"; Interview with Joseph Menn, by telephone, September 3, 2015. Hereafter cited as Menn interview. Unless noted, subsequent quotations from and attributions to Menn come from this interview and a follow-up interview conducted via e-mail on September 25, 2015; and Gabriella Coleman, *Hacker, Hoaxer, Whistleblower, Spy: The Many Faces of Anonymous*, Verso: London, 2014, Amazon Kindle Edition, Chapter 7.

8. In 2011, Menn, a noted technology industry author, was a reporter for *The Financial Times*; he is now a reporter for *Reuters*.

9. Olson, *We Are Anonymous*, pp. 3-4 and 19-23; and Greg Hoglund and James Butler, *Rootkits: Subverting The Windows* Kernel, Addison Wesley Upper Saddle River, 2006, p. xxi.

10. Olson, *We Are Anonymous*, p. 12.

11. The material under the sub-headings "Domino One – SQL Injection" and "Domino Two – Passwords" was developed by an academic partner with whom New America partnered on this project. Unless noted, it draws on this partner's technical expertise as well as Peter Bright, "Anonymous Speaks: The Inside Story of the HBGary Hack," *ArsTechnica*, February 15, 2011, available at http://arstechnica.com/tech-policy/2011/02/anonymous-speaks-the-inside-story-of-the-hbgary-hack/ (accessed on December 1, 2015).

12. The Oxford English Dictionary has 171,476 entries for "words in current use." "How Many Words Are There in the English Language?" *Oxford English Dictionary*, available at http://www.oxforddictionaries.com/us/words/how-many-words-are-there-in-the-english-language (accessed on September 30, 2015); and "Number of Words in the English Language: 1,035,877.3," Estimate as of January 1, 2016, Global Language Monitor, available at http://www.languagemonitor.com/global-english/no-of-words/ (accessed on January 29, 2016).

13. "Password Decoder," Provided by HAQ 4 U.com, available at http://haq4u.com (accessed on December 6, 2015).

14. For more information on the MD5 hash, see "What is the MD5 hash?" FastSum Integrity Control, available at http://www.fastsum.com/support/md5-checksum-utility-faq/md5-hash.php (accessed on January 1, 2016).

15. For additional details, see Bright, "Anonymous Speaks...."

16. Interview with Amadeus Stevenson, Chief Technology Officer, Decoded, by telephone, September 23, 2015.

17. Olson, *We Are Anonymous*, pp. 19-21.

18. Section 14 of the "Rules of the Internet" reads, "Do not argue with trolls—it means that they win." According to the website "Know Your Meme," the Rules of the Internet "is a list of protocols and conventions, originally written to serve as a guide for those who identified themselves with the Internet group Anonymous." A troll refers to "someone who intentionally disrupts online

communities." Quinn Norton, "How Anonymous Picks Targets, Launches Attacks, and Takes Powerful Organizations Down," *Wired*, July 3, 2012, available at http://www.wired.com/2012/07/ff_anonymous/ (accessed on December 2, 2015); "Rules of the Internet," knowyourmeme.com, last updated in April 2015, available at http://knowyourmeme.com/memes/rules-of-the-internet (accessed on December 2, 2015); and Mattathias Schwartz, "The Trolls Among Us," *The New York Times*, August 3, 2008, available at http://www.nytimes.com/2008/08/03/magazine/03trolls-t.html?pagewanted=all (accessed on December 2, 2015).

This report carries a Creative Commons license, which permits non-commercial re-use of New America content when proper attribution is provided. This means you are free to copy, display and distribute New America's work, or include our content in derivative works, under the following conditions:

- Attribution. You must clearly attribute the work to the New America Foundation, and provide a link back to www.Newamerica.net.

- Noncommercial. You may not use this work for commercial purposes without explicit prior permission from New America.

- Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

For the full legal code of this Creative Commons license, please visit creativecommons.org. If you have any questions about citing or reusing New America content, please contact us.

© 2016 New America