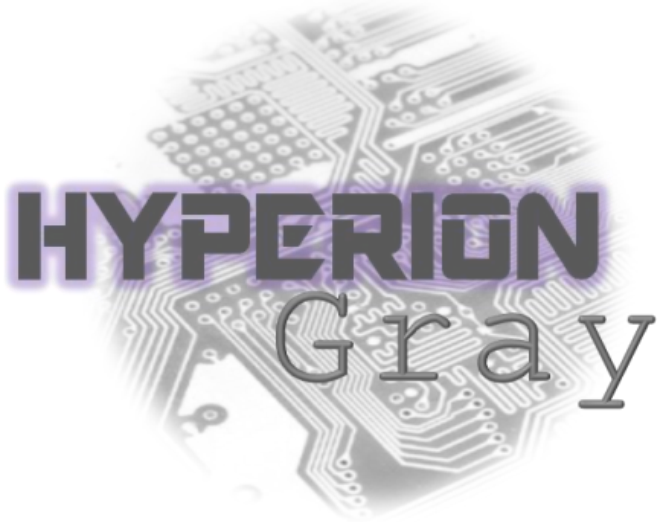
Conducting Massive Attacks With Open Source Distributed Computing

By Alejandro Caceres

(@DotSlashPunk)



DEF CON 21

How to (almost) get fired from your job

Step 1: Speak at a hacker con on your open source community-focused side project (PunkSPIDER)

- Combined distributed computing (my main area of research) with web application fuzzing
- Was pretty cool (if I do say so myself)

Step 2: Have a friend of a high-level executive at your company stumble upon talk at said con

Step 3: Have said friend confuse community-focused web app security side project for a "cyber weapon" and tell executive that you're building a cyber weapon in your spare time.

Step 4: ☹☹☹

Why did I just tell you that story?

- It was the inspiration for this talk – got me thinking about the following:
  - What would it take to build true distributed network attack tools?
  - Where can distributed computing help the most?
  - How can one simply and quickly build distributed attack tools to do whatever it is you're into
    - We won't judge - but don't do anything illegal. Seriously. Please? Ah whatever, you're not listening anyway.
- My goal is simply to explore some of the possible answers to these questions

Distributed Computing Today

- Great advances in distributed computing lately
  - Apache Hadoop
  - Google's MapReduce papers and implementation details
- We've seen some great stuff come out of this
  - Data Analytics
  - Super fast data processing (for faster analytics)
  - Counting things (analytics)
  - Analyzing things (analytics)
- You **might** notice a trend in the above uses of distributed computing or "big data" technologies if you're into buzzwords (looking at you Splunk, IBM, EMC, etc. etc. etc.)
  - Spoiler: we're mostly using it for data analytics
  - This bores me

Distributed Computing In the (distant) Future

- My main "thing" is using distributed computing / "big data" technologies for massive attacks
  - Most of my research thus far has been in application-level attacks
- *I want to dive into this area and see what's possible!*

High-level idea behind distributed attacks

- Much respect for the 1337 hackers out there, working on extremely complex low-level problems to break into things
- However, much of the time this isn't needed. Especially on the web application side, if you choose a big enough target (e.g. a country), you're going to break into things. Lots of things.
  - We've seen the awful state of web application security in our distributed fuzzer unleashed on the Internet. (http://punkspider.hyperiongray.com)
- <analogy> Try enough door knobs, and some of them will be open. In many environments, lots of them will be open. Or at least have a broken lock that you can kick in easily. </analogy>

Why *Distributed* Attacks?

- Often the time required to attack a target is way too long
  - Longer attack times may mean more chance of being detected and stopped
  - Extremely large beds of targets may be completely infeasible due to time restrictions and coordination issues
- E.g. PunkSPIDER – our target was the entire Internet
  - The Internet is a big place, it would take years to scan it properly, even just for high level vulnerabilities
- Coordination between computing resources
  - Without coordination between various computing resources, you may end up duplicating a lot of effort and the attack may be less effective

But distributed computing sounds hard...

- It's not! Huge advances in recent years make it really easy to get up and running
- In this talk we'll focus on Apache Hadoop, one of the best, and simplest, implementations of distributed computing

Hadoop and Me (and You)

- I ~~really like~~ love Hadoop
- Hadoop is an implementation of the MapReduce distributed computing concept
  - You write a Map function that gets distributed across the cluster – it takes in several key-value pairs as inputs and emits several key-value pairs as outputs
  - You write a reduce function. A partitioner sorts the output from the map function by its keys – each set of key-value pairs with common keys are sent through the map function, which emits a final set of key-value pairs. This final set should be the solution to the original problem you were trying to solve
- If you're confused, it's actually pretty simple in practice. It's also awesome, and easy to implement.

Using MapReduce – PunkSCAN Example

- The classic example for MapReduce is a "word count" example. It counts words real fast, cool huh? False. This is uber boring.
  - I even tried adding animated .gif flames and spinning .gif skulls to my word count job and it was still way too boring to show you
- Let's take a better example
  - You have a list of a ton of websites, you want to see if they have obvious vulnerabilities
    - In this case, lets assume we just have the list of sites
    - In PunkSPIDER our list comes from automated crawling of the Internet using a distributed crawler

Using MapReduce PunkSCAN Example (cont.)

```
class PunkFuzzDistributed(MRJob):

    def mapper(self, key, url):
        '''Yield domain as the key, and parameter to be fuzzed as the value'''

        #takes in <None, url> as the <key, value> of the mapper input

        self.set_status(u'building PunkFuzz object')
        mapper_punk_fuzz = punk_fuzz.PunkFuzz(self)
        parsed_url = urlparse(url)
        domain = parsed_url.scheme + "://" + parsed_url.netloc + "/"

        if mapper_punk_fuzz.check_if_param(parsed_url):

            self.set_status(u'checking if URL has param')
            parsed_url_query = parsed_url.query
            url_q_dic = parse_qs(parsed_url_query)

            for query_param, query_val in url_q_dic.iteritems():

                self.set_status(u'looping through params and vals')

                #and now we fuzz
                mapper_punk_fuzz.punk_set_target(url, query_param)
                vuln_list = mapper_punk_fuzz.fuzz()

                #output vuln_list and domain for each url and query param pair
                yield domain, vuln_list
```

Using MapReduce PunkSCAN Example (cont.)

```
def reducer(self, domain, vuln_lists):

    full_vuln_list = []

    #iterate over all lists of vulnerabilities corresponding to a single domain
    for vuln_list in vuln_lists:

        full_vuln_list = full_vuln_list + vuln_list

    self.set_status(u'Indexing')
    #win
    mapreduce_indexer.PunkMapReduceIndexer(domain, full_vuln_list, reducer_instance = self).add_vuln_info()

    yield domain, full_vuln_list
```

Demo Time!

- Let's see PunkSCAN in action
- This is live production data being indexed to PunkSPIDER!

My Love Affair With MapReduce

- If you're astute you noticed a few things in my example
  - It's written in Python
  - It's only a few lines of code
- Some additional stuff I can tell you
  - As far as fuzzing goes, what I showed you is the only part of PunkSCAN that is "distributed computing-focused" code (the rest is a pretty standard fuzzer that I wrote and other basic python code)
  - It works REALLY well – we've scanned over 1.5 million domains using this code and found hundreds of thousands of vulnerabilities. It's really stable and very very fast
  - More nodes means faster fuzzing – simple as that

What is a Hadoop and Where Can I Get One?

- Apache Hadoop is a free and open source implementation of distributed computing with MapReduce
- It's very easy to set up on pretty much any Linux distro (I recommend trying it out on Kali, it works great!)
- A small cluster in the cloud can be built within a couple of hours
- Alternately you can build your own off of really old hardware
- Various other options – Amazon's EMR provides a Hadoop-like environment on demand
  - They don't like you hacking on Amazon's EMR
  - I got kicked off of AWS so take my advice on this with a grain of salt

Use Cases

- Now that we have the basics out of the way – it's time to talk about what we can do with this
- Three Examples we will be covering
  - Distributed recon
  - Distributed attack
  - Distributed password cracking

Use Case 1: Distributed Recon

- Why distribute recon?
  - Greatly speed up repetitive tasks
  - Wonderful for finding a massive number of low hanging fruit
  - Can make deep recon across a massive number of targets (e.g. an entire country's IP ranges) feasible in a short period of time)

Use Case 1: Distributed Recon

- The best example is PunkSCAN
  - We use Hadoop Streaming, a Hadoop function that reads input and output from stdout, allowing you to write code in whatever language you want (this is why PunkSCAN was in Python)
- Heads up: Consider your problem – are you in need of CPU, Memory, or bandwidth?
  - If the former two are needed, any old cluster will do. If bandwidth, you need to carefully plan where your nodes are from a network standpoint
  - Always think before you code. You could waste time distributing something that might not help you that much to have distributed
  - In the case of PunkSCAN we did some pre-research to ensure that distributed fuzzing would help us (fuzzing is highly CPU and memory-intensive – bandwidth is a minor consideration – even for remote fuzzing)

How to get your own

- You can download PunkSCAN from BitBucket
  - We'll give you a link at the end of the talk
- You can write your own pretty easily:
  - Pick your favorite URL fuzzing library (there's a bunch out there)
  - Grab a library that will help you abstract the process of writing a mapper and reducer for Hadoop (we used the MRJob Python library in PunkSCAN)
  - Write a mapper and reducer leveraging the libraries
  - Run it across your cluster and watch it fly
- It really is that simple
  - Though admittedly testing and debugging is a pain

Use Case 2: Distributed Attacks

- Why distribute exploitation?
  - It's fun
  - You can conduct large-scale automated attacks in a short period of time – owning massive targets in a short time (such as entire countries)
- We'll be looking at the example of automated SQL Injection attacks by distributing everyone's favorite automated SQLi tool, SQLMap

Use Case 2: Distributed Attacks

- The basics
  - We use SQLMap's code as a "library" of sorts
  - We pick an abstraction library for writing a MapReduce job
    - In this case we picked the MRJob Python library
    - We write a mapper
    - We write a reducer
    - We run the job
- You may already notice a pattern – it's all about writing a MapReduce job
  - To see our detailed Mapper and Reducer, please visit www.hyperiongray.com and check out our code downloads section

Use Case 2: Distributed Attacks

- Demo (against our cloud test environment)

Use Case 2: Distributed Attacks

- Notice the simplicity of the code and the few lines of code/customization required to run this
- In the end, we end up with a bunch of stolen databases in Hadoops HDFS
  - HDFS is a central file system that Hadoop creates – it is accessible via any of the nodes
  - How much easier can it get? We don't even need to worry about which node we're on to store or retrieve data
- Now that we have all of these stolen databases, now what?

Use Case 3: Distributing Post-Exploitation Activities

- Why distribute?
- Attacking a *lot* of targets at once will leave the attacker with a ton of extracted data
- Password hashes to crack, data to analyze and parse
- From the vulnerabilities we've seen in PunkSPIDER this could be a LOT of data especially for password cracking – we need a better solution than single node cracking
- Why not repurpose old, commodity hardware to build your own cracking cluster?

Use Case 3: Distributing Post-Exploitation Activities

- Admittedly, this is one of the more complex tasks
  - We went with Java instead of Python (for performance)
  - Partitioning the job is non-trivial
- Luckily, you can just download our cracker PunkCRACK, free and open source, and use it and not worry too much about the internals
- However, for those of you more curious folks, you can see our detailed Mapper and Reducer at www.hyperiongray.com in the code downloads section.

Use Case 3: Distributing Post-Exploitation Activities

- Demo (again, against our own test data in our own environment)

Bringing It All Together

- We've thoroughly enjoyed proving the concept here, but what does this mean for you?
  - Leveraging distributed computing from an offensive perspective gives you the power to run massive attack scenarios – this lets you build custom tools to do that using open source technology and commodity hardware
  - Imagine "pen testing" an entire country – it's entirely feasible with the tools and concepts I've presented
- We think the security implications of this concept are broad – if we can feasibly simulate a massive attack scenario, then we can better study this and prepare for it.

Wrap-up

- Follow me on Twitter: @DotSlashPunk
  - I'll answer your questions if you are following me (personal questions answered on a case-by-case basis…)
- See more about us and more details on this presentation at http://www.hyperiongray.com
- See Check out PunkSPIDER at http://punkspider.hyperiongray.com

Thanks

- Thanks to:
  - Tomas
  - Mark
  - The SQLMap project (and everyone involved)
  - The Apache Software Foundation (and the Nutch and Hadoop community)
  - And of course THANKS to all of you for coming to my talk!
  - DEF CON 21 and everyone involved