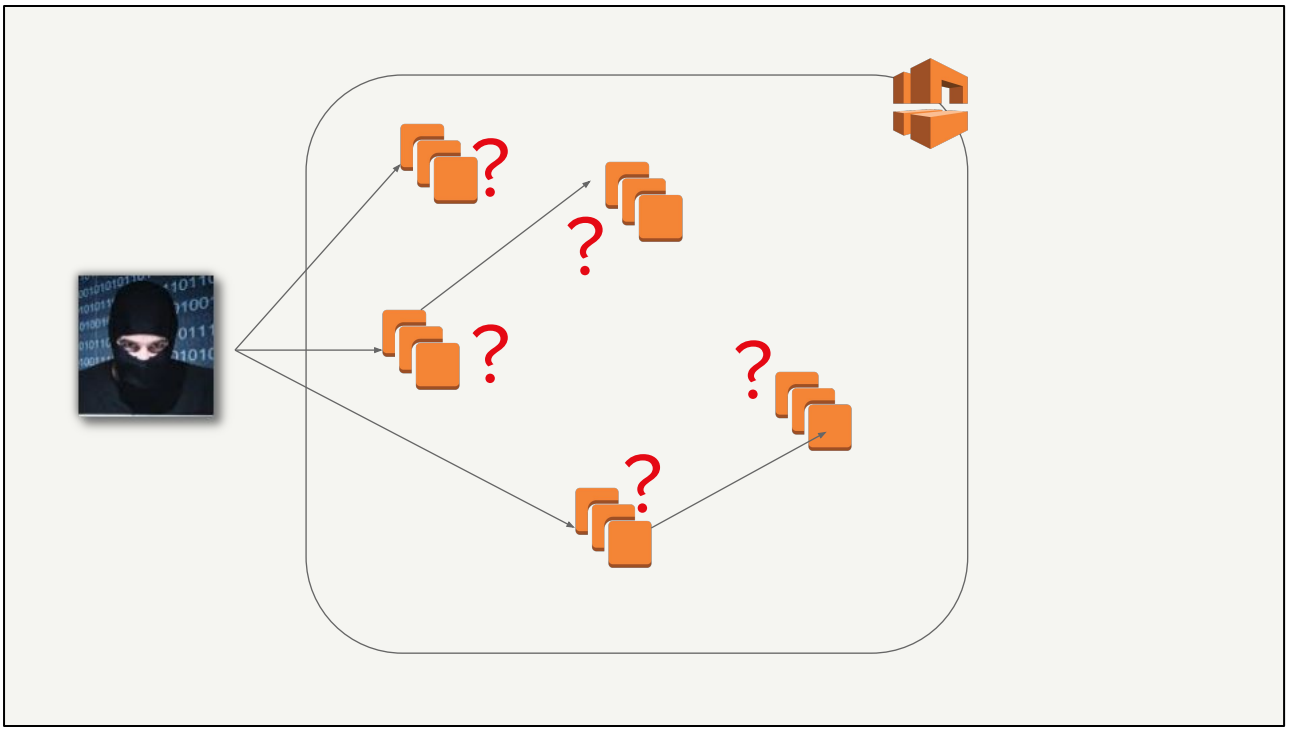
Diffy.

A DIFFERENCING ENGINE
FOR DIGITAL FORENSICS

NETFLIX

- Digital Forensics and Incident Response (DFIR) teams work in a variety of environments to quickly address threats to the enterprise. When operating in a cloud environment, our ability to work at scale, with imperative speed, becomes critical. Can we still operate? Do we have what we need?



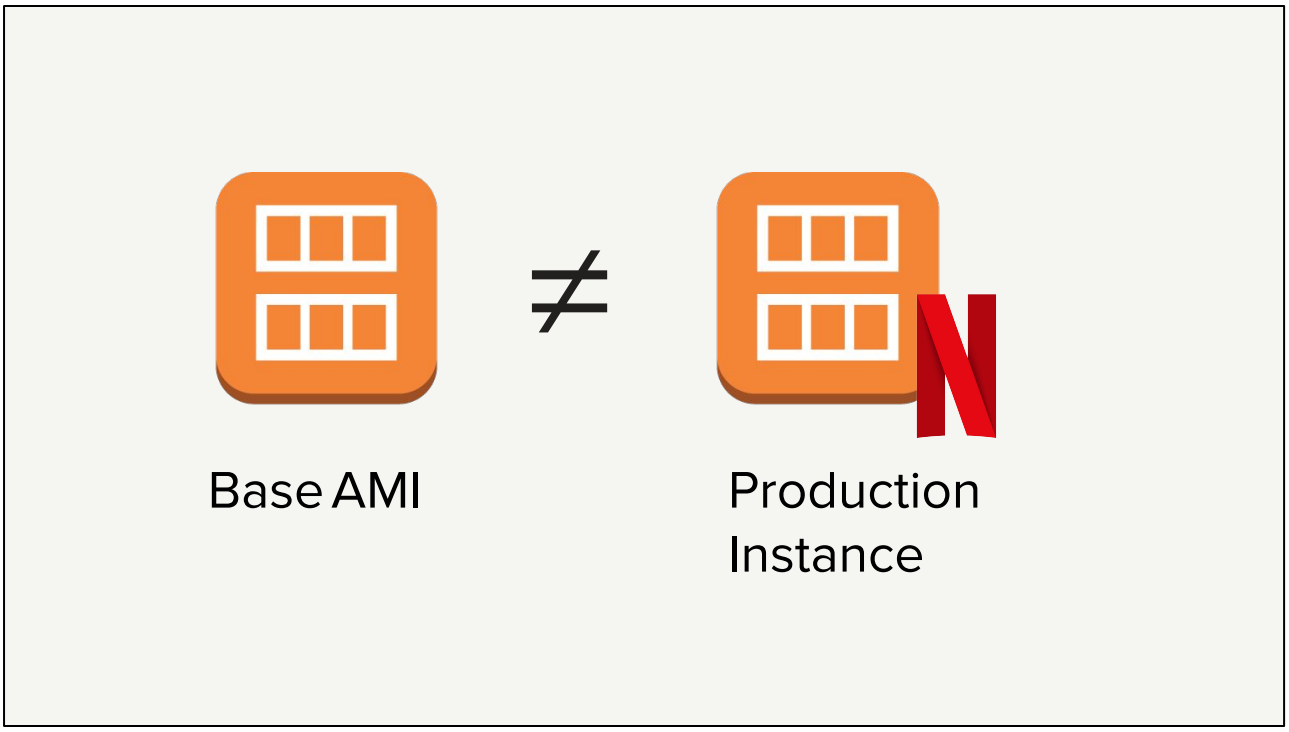
- When moving through systems, attackers may leave artifacts -- signs of their presence -- behind. As an incident responder, if you've found one or two of these on disk or in memory, how do you *know* you've found all the instances touched by the attackers? Usually this is an iterative process; after finding the signs, you'll search for more on *other* instances, then use what you find there to search again, until it seems like you've got them all. For DFIR teams, quickly and accurately "scoping a compromise" is critical, because when it's time to eradicate the attackers, it ensures you'll really kick them out.

Lack of Normal:

“Normal” varies.

- **First challenge: A lack of “normal.”**
- We operate on AWS. Other than our base Amazon Machine Image (our Base AMI), we didn’t have an idea of what was “normal” from a security perspective.

- At Netflix, to become an application instance, as in most cloud architectures supporting CI/CD -- continuous integration or continuous delivery -- processes, the Base AMI goes through a baking, “aminator” and deployment process to become ready to serve traffic.
- This process includes the installation of software from internal code or software artifact repositories, and the startup of new running services or new open ports.
- In some unusual cases it may involve downloading additional packages from the Internet.

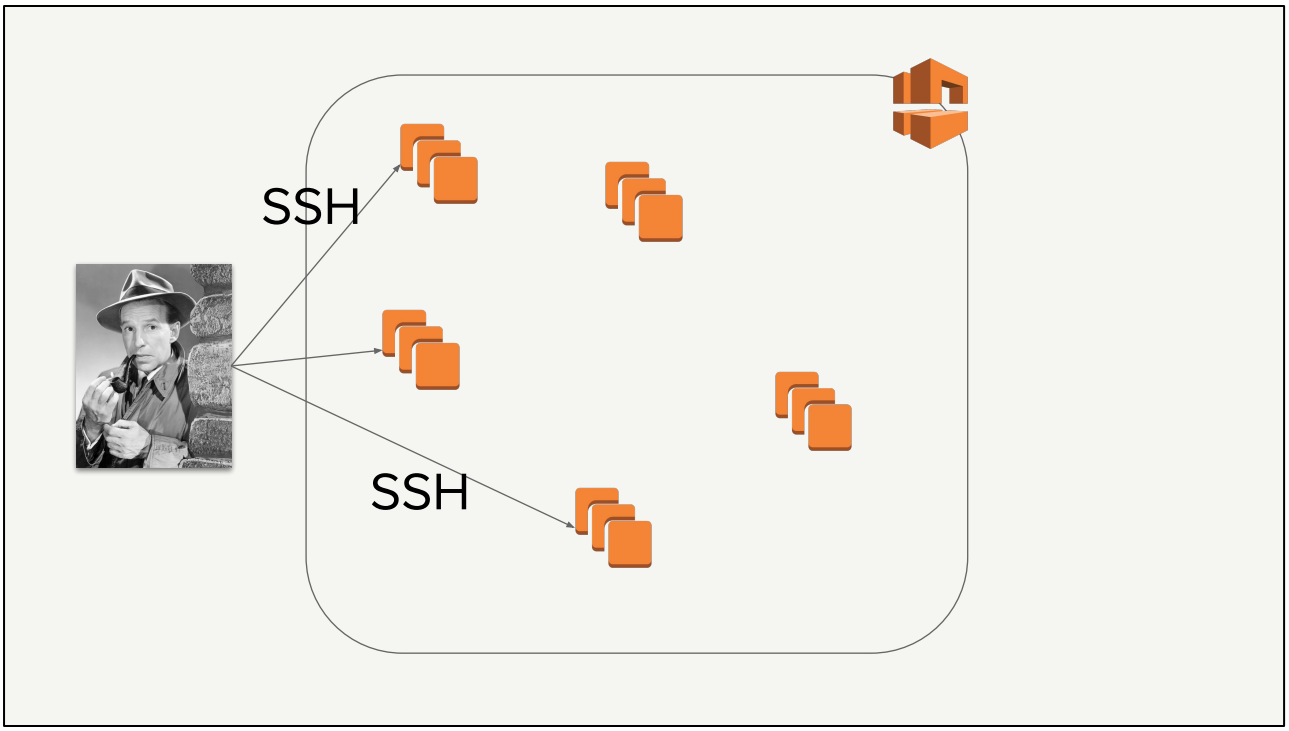


- In short, an application's running instance looks in many cases very different from the base AMI.
- We don't maintain a baseline of what this application's running instances looked like.
- So, we haven't known what "normal" was, for instances of that particular application.

Example IOCs:

- Log signature & response code
- File name, hash, and location

- Example indicators of compromise may include a log signature and response code, or a file name, hash and location.

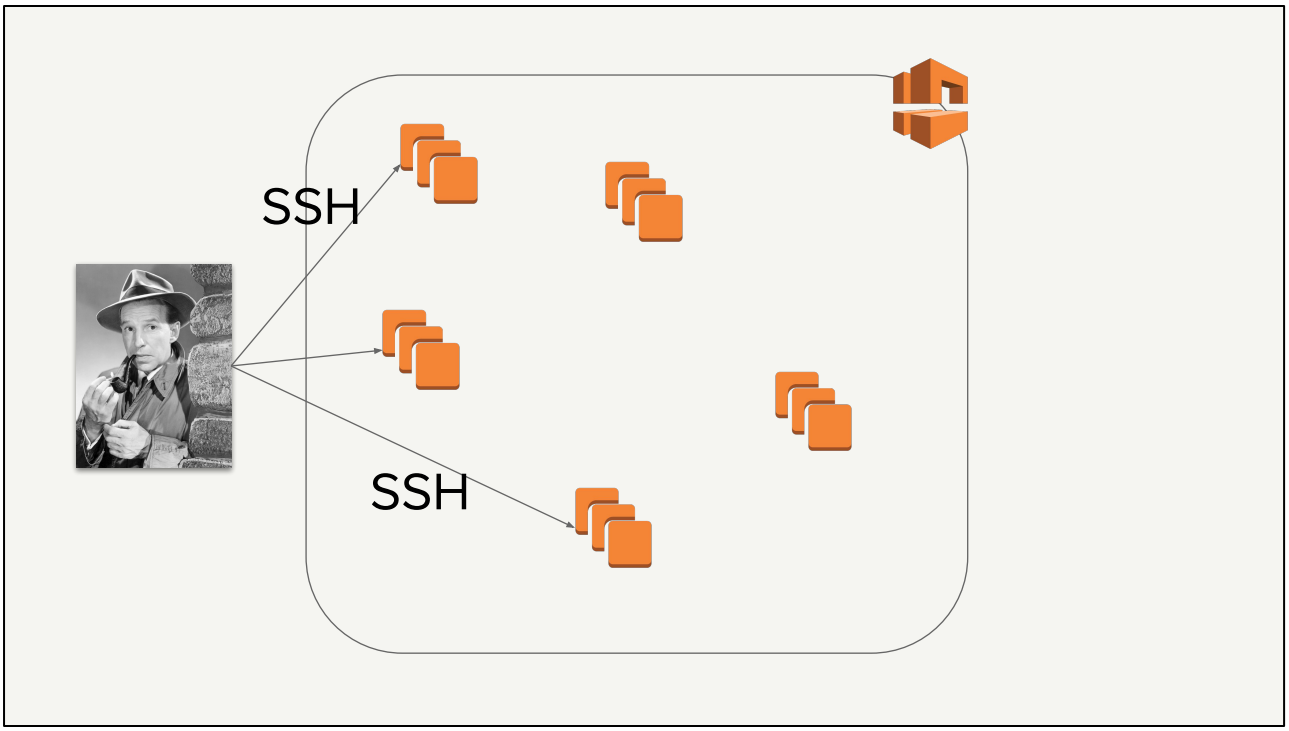


- We've scripted solutions to find such things using SSH, but we've also wanted to create an easier, more repeatable way to address the issue.

The Need for Speed:

Clusters roll.

- **Second challenge: Need for speed.** Our instances turn over quite quickly. We do thousands of pushes a day, and clusters are rolling all the time.
    - You don’t want application owners to have to concern themselves with whether they can roll a cluster or not.
    - Preserving the whole cluster inline will quickly become inefficient.
    - You don’t want to have to do deep-dive forensics on every single host. You want to identify those that need this deep-dive as quickly as possible.



- We do have a way to script our searching, or run live response, across many instances… usually requiring SSH.

Could we... baseline?

- So in the face of these two challenges, the Lack of Normal and the Need for Speed... what if we had a baseline?
  - When a particular application is showing signs of compromise...
  - What if we had on hand, a recent baseline of a freshly launched instance, with all installed software, configuration and running services? We could quickly compare that to the instances of our suspicious application.
  - Those that varied from the baseline in interesting ways would be worthy of further investigation.

- Osquery could be a great solution for this.
- However, we do not have the osqueryd daemon running in production, and can't take advantage of osquery's differential queries.
- We can still use the interactive query shell ("osqueryi"), and we've gotten that broadly deployed.

- Diffy quickly identifies which instance characteristics differ in interesting, security-significant ways from an established baseline.
- Or, which instance characteristics show up through... cluster analysis (that is, differ among a clustering of those characteristics across all of an application's instances).
- We do this using one or both of two methods: a "functional baseline," and a clustering method.
- The baselines, and the observations during an incident, are all collected with osquery.
- We can still operate if we don't collect a pre-incident baseline, using the clustering method. More on that later.

Functional baseline

- OSQuery binary installed
- Queries run, baseline obtained
- Results to ElasticSearch

- How does Diffy build the functional baseline?
- On a newly deployed instance, it installs the osquery binary, runs queries and retains the results.

How does Diffy build the functional baseline?

- On the left is our AWS virtual machine.

How does Diffy build the functional baseline?

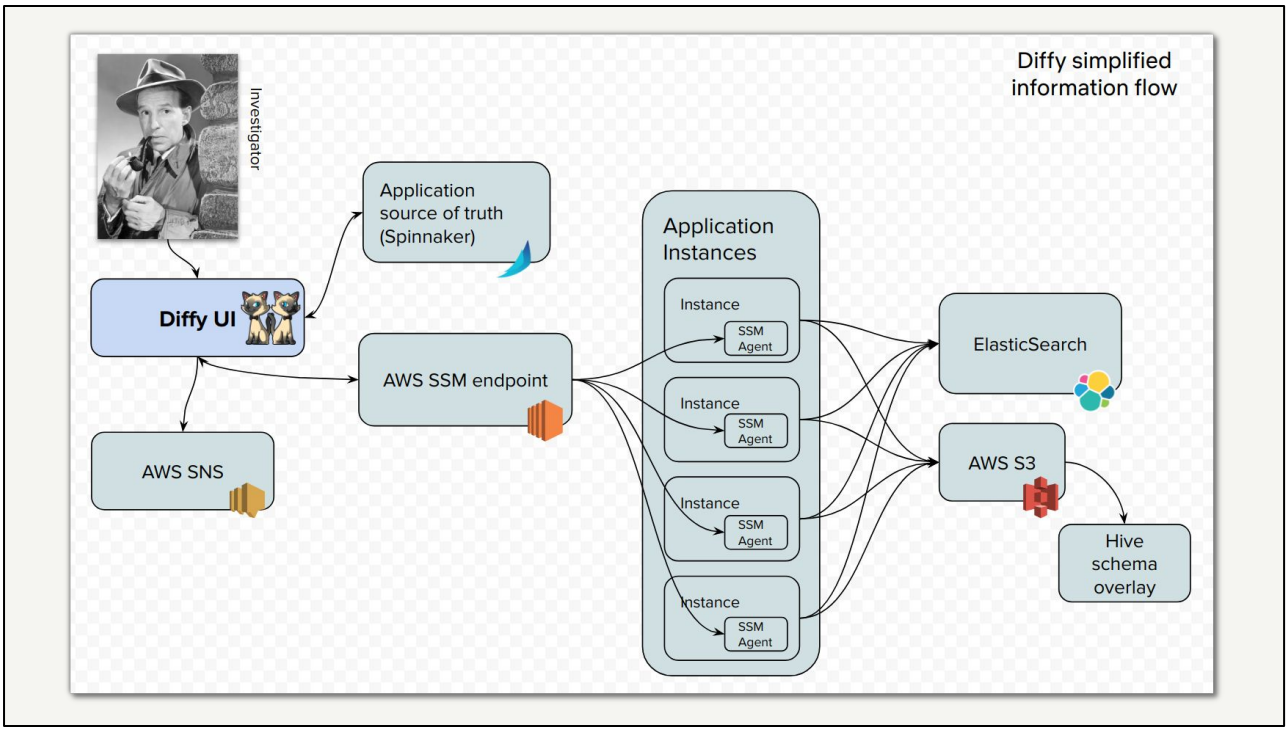
- To build the functional baseline, Diffy installs osquery...

How does Diffy build the functional baseline?

- ...requests data collection...

How does Diffy build the functional baseline?

- ...and sends the result to ElasticSearch.



- Here's an idea of the information flow.

Clustering method

- OSQuery binary installed
- Queries run, observations obtained
- Results to ElasticSearch
- Clustering algorithm runs

- For the clustering method, no pre-incident baseline need be collected.
- During an incident, osquery table output is collected from all instances in an application group, and a clustering algorithm is used to identify dissimilar elements in system state.

Standing out from the pack

- Unexpected listening port?
- Missing iptables rule?

- How does one stand out?
- You'll stand out if you have an unexpected listening port, or a missing iptables rule, for example.

The future

- Differencing engine as a service
- Take better advantage of OSQuery
- Validating Diffy

- We continue to develop this as a service. Ultimately application owners could trigger their own automated DFIR processes for our follow-up.
- We could take better advantage of osquery.
- We will be validating Diffy with attack platforms, as well.

Thank you.

Forest Monsen
fmonsen@netflix.com
https://github.com/Netflix-Skunkworks/diffy

NETFLIX

- Thanks! Download our code from https://github.com/Netflix-Skunkworks/diffy .