Automating Intuition:
Applying Machine Learning to Outstanding Network Asset Detection
What is Gold Nuggeting?

Enterprise networks house thousands of devices (IoT devices, servers, laptops, etc.), some of which present particularly ripe targets for bad actors. Over the course of years of experience, the best pen testers can find these priority assets, the ones best suited to compromise or collect valuable information, and from which to launch a successful attack.

Delve - founded by former pen testers and AI researchers - has developed an open-source tool that encapsulates the accumulated knowledge of experienced pen testers, and combines that expertise with advanced machine learning to automate the process by which accomplished pen testers find the most interesting assets...the “Gold Nuggets.”

In this white paper, we’ll detail the concept of “Gold Nuggeting,” or how to dig for outstanding network devices using machine learning. We will present the problem of information overload when conducting offensive security assessments and how intuition in context - gained through years of experience - sets experts apart from novices. We will then show how, using a modern, yet simple, machine learning toolset, one can hope to overcome the problem of expertise scarcity, and help automate the critical Gold Nuggeting process.

Intuition Scarcity

When first evaluating the result of network scans, typically an Nmap report, the experienced intruder will quickly get a “feel” for the type of network he is currently in. During this critical early stage of an attack, the intruder is attempting to understand the relationship between the network devices he is looking at, or the underlying structure in the data, all of which is highly context-dependent. It is only when he understands the overall context of the network that he can start digging for gold.

For experienced hackers, red flags will quickly emerge. Select assets will stand out, for example, a linux server in a range of Windows workstations, a machine with multiple http servers on seemingly arbitrary ports, an unusual hostname scheme, or a list of exposed services that indicates administrative purpose to a machine.

The human challenge, of course, is that a typical enterprise network will host thousands of endpoints, far too many for a few security team members to constantly track and evaluate for their “attractiveness” to a potential intruder.
Yet intuition, acquired through years of experience, is what sets experts apart from novices (for both security professionals and their bad actor adversaries). Intuition is the ability to look at a large amount of information, quickly spot interesting items, and dismiss the rest.

So, what if we could substitute the process of manually sifting through mountains of entries in a network security assessment to extract the most valuable targets...the gold nuggets?

As it turns out, we can.

In the paragraphs below, we’ll discuss how machine learning, and more specifically unsupervised anomaly detection, is perfectly suited to empower security teams to automate the process of Gold Nuggeting, or outstanding asset detection.

**Elements of Machine Learning**

**The Goal of Machine Learning**

The goal of machine learning is to automatically learn facts and inferences about a given population given only the data representation of a sample of this population.

The hypothesis is that by optimizing a given task on the training data, the algorithm will be able to generalize this task on the rest of the population. This implies that the new, unseen data is generated by a process similar to the data obtained for training. For example, we can believe that if an algorithm is good at recognizing cats from dogs in a picture, it has “learned something” about cats and dogs in general.

In our case, the population in question is the actual, physical network and all its connected devices. The sample is the subset of hosts and ports that have answered back to the Nmap probe. The data representation is the extracted Nmap report, containing structured information about the network.

The task is to dig for outstanding devices.
Automating Intuition: Applying Machine Learning to Outstanding Network Asset Detection

Implementing Machine Learning

Since machine learning algorithms are just fancy mathematical chains of operations, what they need as input is a numerical data representation of the network. Moreover, for efficiency reasons, a numerical representation that focuses on specific relevant features of the data is preferred over a raw, unstructured blob of numbers.

A machine learning researcher will use her ML and other mathematical flair, along with the domain expertise of security researchers and penetration testers, to craft a clever numerical representation of the network. This representation will focus on specific characteristics of the network devices that will isolate important assets. We’ll call these features “elements of intuition”.

The subtle point here is that although a great variety of features can be evaluated at any given time, we don’t know in advance which one will make the asset stand out from the rest. The context will dictate the separation of relevant elements of intuition from irrelevant ones.

This is a desirable property of machine learning.

Unsupervised Learning

Unsupervised learning is a class of machine learning algorithms that specifically use the underlying structure of the data to solve a task. This is in contrast to supervised learning, where the model is trained to optimize on the prediction given a supplied label for each example, say tagged faces in photos or pictures of cats.

Some archetypal unsupervised learning tasks include data clustering, graph community detection, pattern recognition, and anomaly detection. The last one, and more specifically the Isolation Forest algorithm, is the base of our Gold Nuggeting methodology.

Isolation Forest for Outstanding Asset Detection

There are many techniques to detect anomalies in numerical data. Most of them work by defining a distance between points, and then computing the data points that are farthest from the others. This works well with small amounts of data but gets very ineffective when the number of points and the number of features grows high. We call this property the curse of dimensionality.

Isolation Forest is a very efficient and clever solution to this problem. The algorithm works by building a collection of isolation trees. An isolation tree is built by randomly selecting a feature from the feature space (elements of intuition), then randomly selecting a split value between the minimum and maximum value of this feature.
The process is repeated until every point in the description space is completely isolated from the other ones. Intuitively, if a data point is isolated more quickly than the rest, it should be considered as "outstanding", our gold nuggets. The figure below shows the process of randomly splitting features and the construction of the associated isolation tree.

The graph on the left shows two generic features, the tree on the right is built using the random splits from the algorithm corresponding to vertical and horizontal lines on the left graph. We put the colored data points along the tree as they are being isolated by the process.

The randomization in the algorithm is extremely important as it ensures fairness in the process. Then, the entire choose-split-repeat step is itself repeated a large number of times as we build the forest. The average number of splits, or path length, required to isolate a point is computed for every data point. The smaller this number, the more "outstanding" the datapoint.

Batea in Production
Internal Representation

Batea works by taking an XML version of an Nmap report, parsing it into a collection of Python structured objects, one for every host. Batea then applies specific transformations to obtain numerical features. These features are just aggregated characteristics or statistics that represent each host, for example, the number of open ports, the complexity of the hostname, the IP address octet, etc. These numbers combined are vectors, or rows, forming the numerical representation of the network devices. Ultimately, the collection of these vectors into a matrix represents the whole network.

<table>
<thead>
<tr>
<th>ip0</th>
<th>ip1</th>
<th>ip2</th>
<th>ip3</th>
<th>service_count</th>
<th>port_entropy</th>
<th>os_windows</th>
<th>os_linux</th>
<th>os_other</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>39</td>
<td>2.25</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>2</td>
<td>22</td>
<td>10</td>
<td>1.67</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>6</td>
<td>1.02</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>51</td>
<td>1</td>
<td>0.6</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>2</td>
<td>45</td>
<td>4</td>
<td>0.78</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>4</td>
<td>0.96</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>2</td>
<td>217</td>
<td>20</td>
<td>2.04</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

An elementary matrix representation of a network

Feature Engineering

Almost every machine learning algorithm takes as input a matrix of numbers, having each object represented as a row, and each column acting as a descriptive variable of these objects. Hence, the crafty part of Batea, namely the design of elements of intuition, asks that we transform the internal Python objects into numbers that have the three following desirable properties:

- **Ordinality**: We want these characteristics to induce an order on the objects, so that we can compare any network device as being smaller or bigger, with respect to this feature, than any other one.

- **Variance**: We want these numbers to express a lot of variability, so that the spread of values implies the possibility of isolating outstanding network devices.

- **Relevance**: The numerical features need to represent a characteristic of the device that makes it interesting for an intruder. It should say something about its position in the network, its importance, criticality, complexity, misconfiguration potential, etc.
For example, the figure below shows the combination of two features computed from the Nmap report. On the Y axis, we have the port number entropy. This quantity represents how “unusual” the combination of ports is on the asset, while on the X axis, the number of recognizable services that have been found on the device. From the combination of these two features - or, again, elements of intuition - we can run the Isolation Forest Algorithm, and obtain a normalized anomaly score, the scale of which is shown on the right side of the graph. Here, because of the normalization process, a higher anomaly score means a more outstanding device.

**Numerical Representation of Devices**

![Numerical Representation of Devices](image)

**Gold Nuggets**

Given that we have taken meaningful elements of intuition all at once, the fact that the Isolation Forest algorithm always takes the whole dataset into consideration ensures that the network context is embedded in the anomaly score used to predict Gold Nuggets.

For a given example network, the distribution of the anomaly score is shown on the graph below. The pattern of “herd” on the left and a few isolated points on the right clearly show that there are only a few outstanding assets. These are the Gold Nuggets that we’re looking for. Batea will sort and output them for you. In this case, it is a Linux mail server in a Windows workstation enterprise subnetwork.
Model Persistence

Batea also allows for model persistence. That is, one can save models trained on previous data and later use these models to predict Gold Nuggets on unseen data. Although it is possible (and default behavior) to train the model and predict on the same network data, the ability to combine multiple networks together in order to obtain a baseline for “normality” is what makes this tool so valuable.

Next Steps

As we have seen, Batea allows a pen tester, or anyone really, to use the underlying context of their network, along with elements of intuition, to dig for Gold Nuggets with very minimal effort. Moreover, Batea is designed to be a living community project. Whether it is by adding specific elements of intuition, exchanging trained models and training data, or by extending its functionalities, this framework only makes sense if it is useful in real assessments.

Feel free to give us comments, improvement ideas. Fork it or make pull requests at https://github.com/delvelabs/batea
You can also try it live at https://batea.delvesecurity.com/

About Delve

Delve is transforming the way enterprises conduct vulnerability management. By leveraging the power of Artificial Intelligence and a view of each vulnerability in the context of its individual characteristics within the organization, the asset's importance, and external factors, Delve delivers a meaningful, prioritized list of vulnerabilities that enables IT and Security teams to reduce vulnerability risk while minimizing remediation activity. Delve was founded in 2014 and has offices in Montreal, New York City, and San Francisco.