EPISODE 1068

[INTRODUCTION]

[00:00:00] JM: A large software companies such as Dropbox is at a constant risk of security breaches. These security breaches can take the form of social engineering attacks, network breaches and other malicious adversarial behavior. This behavior can be surfaced by analyzing collections of log data. Log-based threat response is not a new technique, but how should these logs be analyzed? Grapl is a system for modeling log data as a graph and analyzing that graph for threats based on how nodes in the graph have interacted. By building a graph from log data, Grapl can classify interaction patterns that correspond to threats.

Colin O'Brien is the creator of Grapl and he joins the show to discuss security as well as threat detection and response.

[SPONSOR MESSAGE]

[00:00:55] JM: If you listen to this show, you are probably a software engineer or a data scientist. If you want to develop skills to build machine learning models, check out Springboard. Springboard is an online education program that gives you hands-on experience with creating and deploying machine learning models into production, and every student who goes through Springboard is paired with a mentor, a machine learning expert who gives that student one-on-one mentorship support over video.

The Springboard program offers a job guarantee in its career tracks, meaning that you do not have to pay until you secure a job in machine learning. If you're curious about transitioning into machine learning, go to softwareengineeringdaily.com/springboard. Listeners can get $500 in scholarship if they use the code AI Springboard. This scholarship is for 20 students who enroll by going to softwareengineeringdaily.com/springboard and enter the code AI springboard. It takes about 10 minutes to apply. It's free and it's awarded on a first-come first-served basis. If you're interested in transitioning into machine learning, go to softwareengineeringdaily.com/springboard.

Anyone who is interested and likes the idea of building and deploying machine learning models, deep learning models, you might like Springboard. Go to softwareengineeringdaily.com/springboard, and thank you to Springboard for being a sponsor.

[INTERVIEW]

[00:02:34] JM: Colin O'Brien, welcome to the show.

[00:02:36] CO: Thanks for having me.

[00:02:38] JM: You worked in threat detection and response at Dropbox. What kinds of security threats were faced by Dropbox's infrastructure while you were there?

[00:02:48] CO: Yeah. I was at Dropbox for right about two years on their detection response team, and Dropbox is a fairly nice target for attackers in the sense that there's a lot of sensitive data. On the other hand, we do have a pretty large security team and a pretty significant investment in there. So we saw threats from kind of the actors that you might expect are interested in user data. That's going to be anything ranging from potentially government actors all the way to people looking for Bitcoin wallets or really just taking over accounts to sell them on the dark web. There is really quite a range of different types of attacks that we had to consider at any given time.

[00:03:32] JM: There are a lots of events that are happening across the Dropbox infrastructure. How were these events managed? How did the vast quantity of information that gets generated from user events get managed and stored and accessed?

[00:03:51] CO: Yeah. We had two main environments. That would be a corporate environment and our production environment, and we took two different strategies. So this was something that I actually worked on directly, which was trying to unify our interface to two different sets of data. One of our data stores was a more classic SIM. We would ingest logs from our corporate environment into there and try to parse them out as we could. Then the other one was actually sort of hooking into our production data pipeline. What the developers were already using, using

some internal systems that we had. This was a SQL database. Production logs would then go into there, and this led to some interesting issues as well as some great benefits as well.

For one thing, we got independent scale across our environments, which was nice, but the other thing is we had two different sort of interfaces and languages into very important data that often would have to be joined together or at least there was potential for that need. We spent a good amount of time trying to really work on that problem.

[00:04:55] JM: What was your system for analyzing all the events that could come up this vast quantity of events that is happening across the infrastructure? I know you had like a big Kafka cluster with all these things that are getting stored. Tell me a little bit about the analysis though of those events.

[00:05:15] CO: Sure. Yeah. Analysis could range from a couple of different places. We had analysis sort of just built into the SIM, right? We would run searches over the data looking for individual log events that seemed risky. A lot of this was searching for process commandline arguments, file names that we didn't expect, things like that, or just known tactics that attackers were leveraging. In the production side, it was a very similar case. Sort of manually writing these queries in whatever language these systems required. But in my time there, a colleague of mine and myself built out a system that would ingest the data from both of those data sources, both the SIM as well as our SQL data store through Kafka. It would allow us to write Python code that would sort of take input as events that we had deemed sort of special in those other systems, and then the Python code, extract further information. Try to leverage a little more advanced logic or even hit other APIs. We really just wanted the flexibility of a programming language so that we could perform more advanced analysis.

[00:06:22] JM: Tell me more about detection and response infrastructure at Dropbox. What worked and what failed?

[00:06:30] CO: Sure. Yeah. At Dropbox scale, you definitely run into a lot of things that simply stop working or become prohibitively costly. We spent a lot of time just fighting the fact that most systems aren’t really built to deal with that level of data, and so they start running into problems of storage. At Dropbox, we wanted to make sure that we had a huge backlog of storage. If

something happened a year ago and we found out that day, we wanted to be able to look all the way a year back or further and really understand what was going on. This was a required capability, which means massive, massive retention volumes. Thankfully, Dropbox is a storage company. So we did have some pretty beefy hardware to throw at the problem, but I think for most companies, that's not a reasonable solution.

Another thing at Dropbox, thankfully, we were in a position to sort of build some of that infrastructure out. Our SQL database was managed by another team. They did a great job. Really scaling that thing out, because it had to handle developer logs as well as our security relevant log. I think it took a really significant engineering effort across a number of teams to build out that infrastructure, and security was sort of just one consumer trying to pull a lever here or there to get what we needed out of that system.

[00:07:53] JM: How do you differentiate false positives from actual threats? There's a lot of events that are going be happening. You're looking at different user behaviors. It will be easy to mistake somebody posting a PDF about Bitcoin for something that's somebody putting together a scam. Give me some description for how you look at false positives and how you find actual threats.

[00:08:24] CO: Yeah. Yeah, that's definitely one of the most significant challenges I think any security team is going to face, and I think that if you're working at a company where you have a lot of software developers, you're really going to hit that problem hard. Software developers and attackers share most of their behaviors. It's really only an intent where they differ. A software developer is going to be going into sensitive environments to debug something. They're going to run network scanners. They're going to do things that we would expect an attacker to do. So you kind of run into this issue where you start to see the same user over and over performing a suspicious action and you run into this alert fatigue problem where do you really want to investigate this SRE performing a network scan for the hundredth time this month?

What we tried to do their was leverage contexting as much as possible to make that triaging process really, really quick, because if you're going to force people to investigate the same alert for the same person a hundred times in a month, you're going to run into people just closing the tickets immediately, even if they don't mean to. They're just not going to pay the same level of

attention to it. If you can put the right amount of context right in front of them, they can start to make a decision about what's different. They can see things like maybe they ran the network scan, but they had previously done something else suspicious.

We tried to sort of mash as much context into our tickets as possible, but certainly false positives were something that we kept strict metrics around. We spent a lot of time in our development process when building out our detections. Just making sure if this detection had been in place over the last X number of months, how much would it have caused us pain? We would create dashboards to track those false positive rates. It's something we definitely struggled with quite a lot, and I think the real downside to false positives was that once in a while something that would've been a really good alert, right? Something that would have caught an attacker just went off too often for us to actually consider that something we wanted to respond to every time. II don't think we ever really solved that problem. It ended up sort of being Band-Aids here and there, trying to incrementally improve it and to some extent just elbow grease and brute force to respond to it whenever we could.

[00:10:51] JM: The attackers that you're talking about, these are people who are just – They're like scanning the network traffic to understand what servers dropbox.com is hitting and then they're using typical techniques to figure out what your network infrastructure looks like a little bit and then they're just poking around and prodding and trying to find little vulnerabilities and you are trying to detect those kinds of people who are surveying your infrastructure?

[00:11:24] CO: Yeah, absolutely. I think probably the attacker that most companies including Dropbox are most concerned with is really very simple. It's an attacker who's going to try to get into your environment through something very, very basic like phishing, right? Sending you an invoice or sending you a Word document that has a malicious macro and then some unsuspecting person downloads and executes that, and all of a sudden the attacker has a foothold on to your environment.

From there what most attackers are going to do is start scanning around the environment, looking for authentication services. Trying to move around and gain as much information, as many capabilities as they can. In the worst case, they would then pivot into a production environment, which is where the sensitive data is stored and that's where we really want to

make sure we prevent attackers from performing that movement. We certainly focus on all areas of the attack chain, but our number one concern would be an attacker going from our or corporate or even an external environment into our production environment.

[00:12:31] JM: Okay. You're talking about the use case where there is a social engineering attack, like a PDF with a virus has made its way into Dropbox corporate and somebody has downloaded it. The attacker has gotten into the network. I mean, this is kind of like the zero trust idea. You assume that somebody is inside your network and you're trying to run this logging and threat detection over internal users.

[00:13:05] CO: That's correct. Yeah. We want to make sure that we can see things like what processes are executing on a user's environment? What domains are being looked up? So that if a user ever is compromised, we can trace back what's happened. We can find out where the attacker is currently and really scope out everywhere where the attacker has been. If they're still in the environment, if they're still present are moving around, that's where we would really want to engage probably multiple teams to start responding, remove the attacker and then perform a sort of postmortem procedure. Yeah.

[00:13:39] JM: Okay. Tell you more about the types of data that comes into making these threat detection and response workflows. So are we talking about raw logging data? What are the data types that we can use to make decisions?

[00:14:01] CO: Yeah, that's exactly right. So these are going to be really low-level logs a lot of the time. This instrumentation can come from anywhere. A lot of services, like for example Slack or Gmail, you can audit logs for things like emails coming in or users logging in from their source IPs and we'll collect information like that. We're also going to want to collect things like which processes have executed. That's going to have information about the file that executed, the process ID, the timestamps involved. These are all really very low level. So there's some work on the backend to try to tie these things together. As an example, you might have one log source which would be giving you things like process executions and you might have a completely different log source giving you network instrumentation, right? Not even on the asset. Not on the device, but determining what ports are being used. Giving you network flow data, and then maybe a third one that allows you to sort of tie processes to the ports they were

talking about. It takes a lot of effort on the backend to sort of merge those things together so that you can start asking questions like, “Well, which processes were talking on the network at this time and to what other systems?”

[00:15:15] JM: Okay. Now, why are graphs useful for modeling threats?

[00:15:23] CO: Yeah. In my opinion, the current state-of-the-art is all about events, right? These events are very low-level. They are sort of time series oriented, right? You think of things as sort of individual isolated events that go through time, but that's not really a good model to work with. Almost any investigation is going to require joining that data together, right? You want to be able to not just say, Well, this process was suspicious.” You need to start asking questions like, “Well, and then what did it do? What child processes did it execute? Was there an SSH connection into our production environment? How can I trace further from there?”

All of these little pivot points where we’re from one piece of information to another, this is what graphs are really, really good for and it's what time series databases and to some extent SQL databases just aren’t really built for in the same way. Performing complex joins in your average SIM is borderline unsupported.

As one example, Elasticsearch doesn't have a SQL-like join. It's a primitive that doesn't exist because they've explicitly said they don't believe it's possible to do it in an efficient way in a system like Elasticsearch. So they provide sort of custom queries that act a little bit like joins, but even those they actually recommend that you don't use if you care about query performance.

Where other systems fall apart in terms of joining data together, graphs, that’s like their bread-and-butter, right? That's the thing that they do better than anything else. This actually goes a lot further than even just investigations in my opinion. Detections, that's where you want to talk about attacker behaviors in your environment. To me, detection logic that's constrained to events ends up being not as high efficacy, right?

You're going to have detections that look for things like process names or command line arguments or filenames, because that's the information that an event has. The problem with that

information is that attackers can control it. An attacker controls what process name they leverage. They can change files. You're working with a lot of attacker-controlled information there.

What attackers have a much harder time controlling are their fundamental behaviors. Things like attackers scan the network, attackers create files and execute them, and these are all composite behaviors. You can't easily represent any of those things without looking at many, many events and joining them together. Once again, I think a graph puts the data into a format where describing attacker behavior is just much simpler.

[00:18:06] JM: Now, we would want to model attacks as graphs or threats as a graph, but the data we have is raw log data. How could we build a graph from raw log data?

[00:18:23] CO: Yeah. It's a pretty difficult problem to solve, and I think this is really where most of what Grapl spends its time on is going to be spent. You have to do a lot of data cleaning logs are so low-level. They come in so many different forms and types. Some of them will have one set of information. Others will have another set. Finding out a way to merge that data together in a way that's both efficient and provides an abstraction that you want to work with is really difficult.

One of the processes that Grapl leverages to do this is an identification process, right? What it focuses on is figuring out what the underlying entity that a log is talking about really is. As an example, process execution log might contain a process ID, right? But PIDs are not a great identifier. They collide over time, right? A PID can be reused and they’re not unique to an asset. Grapl uses a number of algorithms and even falls back to some really strong heuristics to determine what that underlying process really is, and then it merges graph data into a node based on that identity.

This means that when you look at a node, you know that it contains all of the information of that entity, which is very different from log-based systems where to understand something, you have to look across tens, hundreds or even thousands of logs, and that's just not really the case with Grapl. All the data gets merged into sort of one central node for a given entity.

[SPONSOR MESSAGE]

[00:20:05] JM: GitLab Commit is coming to London. GitLab Commit is GitLab's community event. GitLab is changing how people think about tools and engineering best practices, and GitLab Commit is a place for people to learn about the newest practices in DevOps and how tools and processes come together to improve the software development lifecycle.

GitLab Commit is the official conference for GitLab and it's coming to London, October 9th at The Brewery. If you can make it to London on October 9th, mark your calendar for a GitLab Commit. Go to softwareengineeringdaily.com/commit and sign up with code COMMITSED to save 30% on conference passes. If you're working in DevOps and you can make it to London, it's a great opportunity to take a day away from the office. Your company will probably pay for it. I'm going to go to a GitLab Commit at some point. It won't be this one in London unfortunately, but I will definitely go because I'm excited about the GitLab ecosystem. It's quite an interesting ecosystem and seeing developed has been cool over the course of Software Engineering Daily's lifetime.

At GitLab Commit, there speakers from VMware, Porsche, and GitLab itself. If you're in London on October 9th, you can check it out. I hope you do check it out, and thanks to GitLab for being a sponsor.

[INTERVIEW CONTINUED]

[00:21:39] JM: Let's take a step back. If we're talking about looking at data from logs or if we're talking about collecting log data and building a graph out of that data, what are the machines that we are logging? Are we talking about client machines? You mentioned basically the social engineering attack where I download a PDF and all of a sudden my computer is now maliciously taken over and other external users can access that machine, or are we talking about like AWS instances, like my EC2 instance getting hacked?

[00:22:21] CO: Yeah. We're talking about all of it really. Attackers are going to want to go all over your network. Attackers don't really know where they're going most of the time. They might have an idea of what they want, but they don't know how to get there. It's a very exploratory

process to move throughout a network. You might start off in an exposed EC2 instance, but you want to get into a different production environment that's maybe AWS-based. Lots of companies are multi-cloud at this point. So the attacker is going to have to sort of pivot around, picking up clues on how developers works. They'll look in your batch history and they'll try to figure out how do people move around this environment. How do I get where I need to go?

Whether it's starting from a corporate machine like a MacBook or a Windows laptop or it's starting from an exposed service, they basically are just going to try to move as far into your network as possible grabbing whatever they can. The scope here is really very wide. It's truly any system that you can get your hands on monitoring for.

[00:23:24] JM: Okay. The logs that were going to transform into a graph, what format do those logs need to be in?

[00:23:36] CO: Sure. It kind of depends. Individual log sources in Grapl's case have parsers for them. As an example, out of the box, you can use Sysmon. This is a Windows instrumentation tool that gives you things like process executions, file creations, host networking and that's sort of thing. It's really powerful, and if you're deploying a Windows fleet, I think Sysmon is really one of the best tools that you can deploy there and it's free as well.

In the case of AWS, we will in the very near future, ingest CloudTrail, GuardDuty and AWS config. Those are going to give you information about API calls happening in your AWS environment, the state of EC2 instances. In the case of GuardDuty, that's actually AWS' own detection logic that they output events for. If AWS builds a new signature, those are going to come out through CloudTrail. Grapl can ingest those and create a graph that represents what the EC2 instance involved was and then you can pivot for what the asset was and the process of off of it and that sort of thing. But it is the case that Grapl needs to know how to parse the data. There is a generic format that you can sort of map to. If you provide your logs in a JSON format that Grapl already knows about, then it can just ingest them right off the bat.

[00:24:58] JM: You mentioned Sysmon. Can you describe in more detail what Sysmon is?

[00:25:03] CO: Yeah, sure. Sysmon is this awesome tool. It's built at Microsoft by Mark Russinovich, who I believe now is the CTO of Azure. Sysmon hooks into your Windows system and it outputs logs based on all sorts of events happening. One of the things that make Sysmon so great is that for a process creation log, just as an example, it's going to do some heavy lifting for you on the client side. It'll determine a canonical identity for that process. We don't have to rely on the PID if you're using Sysmon. It's going to pull in the parent process information. It does some of these joins of information upfront, pulling in things like files, and that's really important if you're using a log-based system because you can't do those joins easily on the backend. Sysmon tries to sort of solve that problem by just joining everything upfront. It does make the logs pretty large because they contain a lot of redundant information that that might be in other logs already, but it makes it a really powerful free tool that anyone can go use.

[00:26:11] JM: And each log entry has a PID and a PPID. Is that correct for a Sysmon log?

[00:26:19] CO: Yes. I believe every single Sysmon log is going to contain information about the process that performed the action, and at least in the majority of cases, possibly all of them. It's also going to pull in the parent process ID and possibly even the parent process name or other information about it.

[00:26:40] JM: Okay. That gives us a trace of the parent process calling other children processes. We could think of that as a graph. You could think – I have a parent process and that calls some child process. That child process calls another process. We can get a graph of some of data. That's one way we can semantically think about a graph being formed. Give me a bit more description for what the nodes and edges are in this graph and how it gets formed.

[00:27:17] CO: Yeah. Really, almost any law log kind of has a graph hidden within it, right? In the case of almost any process execution log, there're always a couple of different entities in there. One is represented by the parent process ID. One is represented by the child process ID. You might have a file path in there, right? That's three nodes right off the bat that we would want to represent in Grapl. If I were to ingest a log like that using Grapl, what I would end up generating is a graph that has a parent process node. At that point, the only information that would be in there is the PID and the time that we last saw it based on that event. We would have the child process.

Again, we would have the child process ID. We would know that the child process was created at that time stamp, and then if there's file information like where the child was executed from, we're going to have a third node, which represents that file, and then edges between them. The parent process would have a children edge to the child. The child process would have a binary file edge, which would point to the file it was executed from. This is really how all of these graphs get generated.

In the case of AWS logs like CloudTrail, they contain information like a source IP address. They contain a destination service. They have request details. Again, we would likely have about three nodes given that information. One for the IP address, one for the request metadata in the API call that was actually made, and then one for the destination service. Whether that's a listening port, whether it's something like AWS S3 or EC2, that would be our graph. The really powerful thing with Grapl is that these two different graphs might end up overlapping. They would end up connecting even though they're from different data sources. If a child process performed in AWS API call, we would see that the child process connected to the source IP address for that API call and we would get a graph that just combines things automatically.

[00:29:23] JM: You've mentioned Grapl a few times. That's the impetus for this show. Explain what Grapl is and how you started it.

[00:29:31] CO: Yeah. Grapl is really the implementation of this approach, this graph-oriented approach. Grapl is a graph-based detection response service that you can run in your AWS environment. I build Grapl over the last couple of years to just solve a lot of the problems that I was seeing or I was experiencing in the industry.

I started off my career at Rapid7 working on the inside IDR product. That's their detection response manage service. I worked on the data science team there and then later the engineering team, and I saw a lot of problems with how a lot of these commercial products are being built to handle data. Logs are so low-level. They have so many problems when you're working with them. I felt that something that could really clean that data up just at minimum providing more enriched logs or logs that have a little extra information in them. Right off the bat, that would be a really important step forward.

What I found when I started working at Dropbox was the main operation that I was doing was joining data together, and that was really difficult. It made things very slow when we were querying and it just wasn't very easy to write those joins. I decided that what Grapl should really make it easy first and foremost is joining data together. It should be very easy to ask a question about what a process did or what's connected to a network address, things like that.

Over about that at this point, nearly 3 years I'd say, I just sort of built out this system to solve those problems. Grapl takes a couple of interesting approaches like gives you Python, which allows you to write much more powerful attack signatures. I've personally found that domain specific languages are a little bit harder work with. Yeah, it really just tries to solve the problems that I've been seeing for so long.

[00:31:28] JM: Give a little bit more description of the usage of Grapl. Let's say I've got a huge media company, Software Engineering Daily. I've got 10 Windows users who are journalists and five Linux users who they're using Ubuntu and they are developers. We got a 15-person organization. We got a bunch AWS infrastructure. All these machines are generating logs. Where are those logs going and how can we turn them into a graph with Grapl?

[00:32:06] CO: Sure. Yeah. Setting up Grapl is pretty straightforward. You can set it up in an AWS account that you own, and Grapl is open source. So you would go check out the GitHub repo and run through that the set up process, which is fairly straightforward. I'd say roughly 15 minutes total. At that point, Grapl is up and running. It's ready to ingest whatever logs you send it as long as it has a parser. At that point, it's sort of a matter of getting the logs to Gralp.

Grapl's ingestion point is an AWS S3 bucket. That's their storage system. It's globally available. It's authenticated. It makes for really nice ingestion point. What you would do is sort of picking what services you wanted to use to send that log to us. So we're working on a reference pipeline right now actually based on one forwarding agent. You would send those logs up to the S3 bucket.

From there, you could write your detection logic. You would go to a Jupyter Notebook that's a Python environment that allows you to write Python, work with Grapl through that Python code,

investigate the data, deploy your attack signatures, and that's basically it. Grapl is going to handle taking those logs and turning them into graphs. It's very operationally system. It's built primarily on serverless software. So you should be able to just focus on building out your signatures.

[00:33:32] JM: The graph itself, so logs are getting parsed into a sub-graph representation. The sub graph is getting pulled into a master graph. Describe the relationship between these sub-graphs and the master graph in more detail.

[00:33:51] CO: Sure. Yeah. This is where the identity aspect of Grapl really plays a role. When Graple gets a sub-graph that's been generated from a log or multiple logs, it's going to try to figure out what the entities that those nodes represent are and they all have an identity, right? This is just an index of what the thing is. It's going to find in the master graph whether that node exists already or not. Because we may have seen that entity before in the case of a parent process, right? We must've seen that process be created beforehand or it's possibly performed other actions. We'll look it up. We'll try to find it. If we don't, we create the node with the metadata that we have. Otherwise we'll just update it, right? The graph is sort of constantly growing, expanding, being updated as new information grows, or rather flows into the master graph.

[00:34:46] JM: Okay. Can you give a little bit more detail on what the nodes and edges would be in this graph?

[00:34:52] CO: Sure. Nodes are going to be things like a process, right? A process would have properties on it. This is going to be things like a process name, the time it was created, a determination time if the process has been terminated, and then there are going to be edges that hang off of that node. Things like a process has children, and so you'll have multiple edges to all of the other process nodes that have a connection to it. If any of those process executions had a parent PID relating to our process node, they would get tied together. You end up with a very highly connected graph primarily representing two things. So you have entities, like nodes, and then the edges are sort of your behaviors. They're the actions across your environment. It's sort of like a noun and verb dichotomy there.

[00:35:43] JM: The nodes or processes. The edges are actions you said or like verbs?

[00:35:50] CO: Yeah. Yeah. It's not just processes. As example, what Grapl comes with out of the box, you have process nodes, you have file nodes, IP addresses, connections. Connections themselves are nodes. They hold data. There are assets, and then there's a plug-in system where you can add your own nodes. Then the edges are really any relationship. Usually some sort of an action that connects the two. A process might have a connection to a IP address, and that would be represented as a process node within outbound edge to a connection node, which would hold metadata about the connection. Then the connection node would have an outbound edge to the destination IP address node. This process might also have a binary file edge to the file that it was executed from. That binary file might have a creator edge for the process that had actually created the file and so on.

[00:36:49] JM: Okay. If I have this graph of IP addresses, processes, files, the different entities that exist across my infrastructure, it seems like that could be really, really big.

[00:37:04] CO: Yeah, absolutely. It's definitely – Companies send up terabytes of events on a daily basis. I've heard of companies sending out tens or even hundreds of terabytes of data every single day and storing all of this is absolutely one of the big problems to solve. Grapl has a couple of really nice advantages here. If you're in a log-based system, you end up having to store every single event in full, and even if that information isn't new, you end up storing it.

As an example, if the process reads the same file a thousand times and it generates a thousand different logs all about reading that file, your storing things like the file name, the process name, the PID over and over and over again because the logs don't understand that that information isn't really unique. They can't coalesce it together. This makes storage sort of linear with the amount of data you send up.

Grapl understands the data. It understands what a PID is. It understands when that PDI is not unique, and because it's coalescing information like PIDs on to the node that underlies that information, if that same process reads a thousand – The same file a thousand times and it goes into Grapl, all that restoring is one process node and one file node with just the PID stored

once, the file name stored once, and then maybe a thousand edges which are relatively light. This gives Grapl a really nice sub-linear scaling attribute that log systems just don't have.

[00:38:43] JM: The graph is represented by a Dgraph? Dgraph is a graph database?

[00:38:51] CO: That's right. Yeah. Dgraph is this really great graph database. They feature horizontal scaling. Really great read and write performance, which is very uncommon in graph databases. They've got a solid query language. It's free open source Apache 2.0. I've been very happy with the choice of using Dgraph. They've been great on Slack for support, and so that is what makes up Grapl sort of data lake. It's where all of the master graph nodes and edges are stored.

[00:39:23] JM: How does Dgraph compare to other graph databases like Neo4j?

[00:39:28] CO: Sure. Yeah. Dgraph is written. I'm by no means an expert on graph databases, but from what I can tell, in terms of performance, Dgraph really handles writes a lot better. What makes Dgraph really special is a combination of having ACID transactions, because Grapl wants to ensure that data is always consistent. It has great horizontally scalable write throughput, which if you're ingesting terabytes of data and analyzing it in real-time, you need to have really, really strong write and read latency and throughput. The fact that it's horizontally scalable is a big differentiator.

A lot of graph databases are not horizontally scalable in terms of write load. You can add read replicas and that's it. That really set Dgraph apart from other graph databases. The fact that it's also open source, you can use Dgraph without any painful licensing restrictions. I have a cluster of Dgraph that I use and I don't have to worry about licensing issues because it's on multiple systems or anything like that. I'm going to have to pay money or be in a violation of some kind of license. It really just fit the bill in terms of what I needed at someone who is getting started with graph databases when I started building Grapl.

[00:40:46] JM: What role does time play in the construction of the graph? We've got nodes and edges, files that have some relation to a process, but that's a graph that's totally removed from the notion of time. It just shows relationships. Does time play a role?

[00:41:07] CO: Yeah, that's a really good question, and I think it's an important one because most people doing detection response have been very used to a very time-focused approach. They use time series databases. They query over searches that are time-oriented. Time plays a really big role in the existing way that we do things. Grapl doesn't have that same concept of ordered events with timestamps. It does encode time information.

As an example, a process has a property for its creation time for the last time we saw it for its termination time, but really the way Grapl deals with time is to more logically encode it, right? If you have a parent process and you have an edge to a child process, there is a logical causal relationship there. The parent must have existed before the child. We can use the metadata, like the child process is creation time and the parent process is last seen timestamp and stuff like that to sort of build a more detailed picture. But in my opinion, what you really want from time is more of a causality relationship. The actual details of this exact millisecond versus that exact millisecond are not as important.

Grapl does store those timestamps, but maybe not at the same granularity, whereas a log source, every timestamp is preserved. Grapl only stores what it thinks are sort of semantically interesting timestamps and then it relies on those relationships to encode causality.

[SPONSOR MESSAGE]

[00:42:53] JM: Today's sponsor is Datadog, a monitoring and analytics platform for cloud scale infrastructure and applications. Datadog provides seamless integrations with more than 400 technologies, including AWS, Postgres, MySQL and Docker so you can start collecting and visualizing performance metrics quickly. Distributed tracing and APM provide end-to-end visibility into requests wherever they go, across hosts, containers and service boundaries with rich dashboards, algorithmic alerts and collaboration tools, Datadog provides your team with the tools that they need to quickly troubleshoot and optimize modern applications. You can see it for yourself. You can start a 14-day free trial and Datadog will send you a free T-shirt if you just go to softwareengineeringdaily.com/datadog. You can start that free trial and get a free T-shirt. Go to softwareengineeringdaily.com/datadog.

[INTERVIEW CONTINUED]

[00:43:57] JM: Once I have this graph instantiated, I can execute an analyzer on Grapl. These analyzers can query the master graph for suspicious patterns. Describe what an analyzer does.

[00:44:14] CO: Yeah. Analyzers are what in other systems might be called alerts or saved searches. These are going to be snippets of Python that contain logic that matches an attacker's behaviors. These are your attack signatures. This could be anything like looking for – The canonical example I usually use is a parent process of something called svchost.exe that isn't on our white list. This is a really common attacker technique. They like to pretend that they're a built-in Windows service so that when you Google what's this suspicious SVC host? You'll get Google results saying, "Oh, that's part of Windows." They do that a lot, but we can create a relationship description of parent and child where we constrain the parent process to not be on our white list, and if anything returns from that, we deem that as suspicious.

Analyzers can also be a lot more general. Something that Grapl allows is much more structural detection logic. Rather than tying ourselves to process names, we could, for example, just look for all unique parent-child process combinations, because we don't know that attackers are going to use one process name or one file name, but we generally expect attackers to execute processes. If we're just tracking all of the suspicious processes across our environments, that's going to be something that an attacker has a really hard time avoiding. All that logic to do that is going to live in your analyzers. These analyzers execute against the graph database as soon as it updates. Every single time a node is updated, Grapl will handle executing your analyzer against that graph database. It's a real-time detection. I it's built to be very, very efficient. You don't have to worry about search windows.

Speaking of time, one of the problems I've certainly had is writing detection logic about relationships where you don't care about time. In a system like a SIM, if a parent process executes a child process hours apart, my search window to search for that relationship has to be over multiple hours and attackers take advantage of this all the time. They execute payloads far in the future. They'll just wait and then execute it a month later, and your signature, if it cares about a parent-child relationship, it's not going to catch something like that. That's another

problem that happens when you over rely on time, and Grapl gives you more of a structural approach there.

[00:46:49] JM: This analyzer, if it finds a suspicious pattern in the graph, what happens next?

[00:46:57] CO: Right. This is where Grapl doesn't have the same concept of an alert. Speaking of false positives earlier, Grapl doesn't really want you to have to deal with a binary sense of good and evil. If you had a unique parent-child process, that very well might be benign, but it is risky. It's a suspicious event. Rather than saying this is bad or this is good, we assign a risk score to it.

Something like a unique parent-child process might have a really low risk score. We want to track it, but we don't want to be waking anyone up in the middle of the night over it. We might give it a risk score of say five. But something like that SVC host with not whitelisted parent, that's almost certainly bad. We've put time into that whitelist. It's very unexpected and it's almost certainly malware. That might have a risk score of something like 100 or higher for that matter.

That allows you to avoid this world of constantly maintaining white list, constantly dealing with false positives, and more importantly we can write detection logic that focuses on attacker behavior without worrying about it creating tons of false positives. We really want to just track that behavior fundamentally.

[00:48:16] JM: How many suspicious patterns might I keep in my set of analyzers? How did these analyzers like run? Do I have this graph and then get a set of analyzers and I run them nightly as a batch job or am I running them continuously as long as some amount of data gets added to the graph? How do the analyzers get used in practice?

[00:48:46] CO: Sure. Yeah. The analyzers are going to execute in real-time. Every analyzer takes in a node that has recently been updated and it gives back to Grapl a query saying does this query match that node? Every time a node updates, which will happen, for example, if process terminating would be considered an update, a process writing to a file would be an update for both that process and that file. Both the process and the file would be scanned by all of your analyzers. It wouldn't be crazy to have hundreds of analyzers. I could even imagine

someone having thousands of analyzers. It's certainly not unheard. Because Grapl allows you to express more behaviors, I can certainly see a customer wanting to have a very large suite of this detection logic. Because it's happening in real-time though, we can keep the performance really, really strong. We don't have to search over hours and hours of data. We're just going to perform a series of indexed operations saying this latest node that updated, is that going to match our pattern? That's really, really efficient. There is no search Windows. Actually, in much of the time, it's going to be either constant time, cashed or we can even locally run the query to determine that it's impossible to match that node.

[00:50:16] JM: Describe the usage of Grapl in a little more detail. Are there any particular companies or users that you could give as use cases?

[00:50:28] CO: Sure. I don't think at this point I could talk about companies that are using Grapl, but what I can say is that we've definitely handed Grapl out to a couple of interested parties, colleagues of mine. There's a version of Grapl that I'll be releasing likely today or in the next couple of days that can run on your laptop without requiring that AWS environment. I'm really hoping to see a lot of people just download it and start playing around with it.

The feedback so far, I think people really resonate with the idea of a graph-based approach. I don't think that it's a – It's not a very hard sell at this point. I think the industry is starting to move in that direction anyways, and what I found is that even though it's a very different approach from what people are used to, it matches their model for how they work anyways. Everyone sort of building these graphs mentally as their working with their time series data, and Grapl sort of just removes that step for you. It does that work on your behalf. I think a lot of people are just – It's a bit of a breath of fresh air to not have to sort of hold that graph in your head and just see it on the screen in front of you.

[00:51:40] JM: Was there something like Grapl at Dropbox?

[00:51:46] CO: We never had a graph-based system at Dropbox. I think if we had, I honestly might not have ever ended up building Grapl. Grapl was very much built because I did not see anything out there that was solving the problem this way and every day I would go in and something would be painful about my job and I decided that I wanted to really fix that. This was

a range of things, but I think it's very frustrating to want to express an attack, and rather than having the hard part be researching attacker behavior, finding out what the right attack is, the hard part was expressing the attack. That's not where I think detection engineer as an incident responders should be spending their time. They should becoming experts on attacker behavior. They shouldn't be fighting their tools, fighting unclean data, dealing with poor performance and things like that. I did not see anything solving that problem and that just sort of fueled me to go out and do it myself.

[00:52:48] JM: Just to revisit the architecture. The large graph that you make, the master graph for a given company, does this master graph encompass all the AWS infrastructure and all the client devices or is it just like a master graph for the server infrastructure? Is it everything?

[00:53:13] CO: Yeah. The master graph is going to have everything, and what that allows is for us to, without even trying, interleave data together. As I sort of explained before with the example of process networking data overlapping with AWS, VPC flow logs or networking data from another source, because they're all stored in this one single distributed graph database, we don't have to manually join stuff together or figure out what one piece of data looks like. They all share this common schema, this common environment. So they're connected together just for free, and that's a really powerful capability that Grapl provides.

[00:53:58] JM: In summary, what problem does Grapl solve the was unsolved before?

[00:54:04] CO: I think the primary problem Grapl was solving without any of these stuff like Python or the implementation details that sort of make it possible is really allowing detection engineers and incident responders to think more in graphs, to take a tool that matches what attackers are doing that matches how you think about solving problems. Grapl doesn't make you fight to put your data together. It doesn't make you fight to connect composite behaviors. It really lets you focus on what I think are the most important parts of detection and response, which is understanding and expressing what attackers are doing. I think everything else is just to support that one goal.

[00:54:49] JM: The detection and response domain that Grapl is focused on, what are the other tools that people are using for detection and response in place of Grapl today?

[00:55:03] CO: Sure. The market is generally dominated by the SIM. That the security event and information or incident management software. This will be things like Elastic alert or ElastAlert. This'll be Splunk. There're systems like Xbeam. Really, there's a whole suite of detection response products out there right now and they all have their own spin on sort of the same thing. As far as I know, almost all of them are really based on logs. Some of them will do enrichment of those logs, but everything is sort of taking the same fundamental approach, which is ingest log data. Give you some way of querying that data, and that's about it, right? It's almost certainly that you work with the logs almost directly. I don't know of anything right now that gives you detection logic using a graph-based approach. I have seen graph start to get a lot more popular on the response side. We see graphs in tools like VirusTotals. There's VirusTotal graph which lets you look at a binary that's been uploaded to VirusTotal and see things like the IP addresses connected to it or things like that. There is Graph History, which is being used a lot to do graph visualizations and some people are leveraging that for incident response as well. But from a detection perspective, Grapl really unifies both the detection and response with that graph-based approach.

[00:56:37] JM: Okay. Well, Collin, what else do you think could be applied to this kind of graph of infrastructure that you can build with these logs? Are there any other applications that you see as being productive in addition to the detection response?

[00:56:58] CO: Yeah, absolutely. I see a lot of applications for other security teams or even just operations teams. Ultimately, what Grapl is doing is giving you data in a format that lets you really work with it much more easily. That's a problem that a lot of people have. Infrastructure security teams want to ask questions like which infrastructure is connected to a port that isn't supposed to be open and have we seen any connections to it? That's a very common like infrastructure security thing to monitor for. Abuse teams might want to see connections to their product from the front-facing, like their edge services based on Tor exit nodes or things like that. Operations teams which aren't really dealing with security might want to understand the processes on a system. Maybe one of them is using more networking than usual or its CPU is going off where there's of lots of process restarts. Ultimately, Grapl I think has a really strong story in the future to tell for solving those problems even if right now we're really focused on

detection response. But yeah, I see a lot of different places were Grapl could solve hard data problems for instant responders of all kind.

[00:58:15] JM: Just tell me a little bit more about the typical – What are a Grapl production deployment would look like? What are the AWS services that if I were to deploy an instance of Grapl myself that would underpin this piece of software who is actually in production?

[00:58:36] CO: Sure. Yeah. Grapl is built almost exclusively on managed and serverless software. Every processing unit in Grapl, the parsers, the process that merges the data into the graph, the process that executes your analyzers, all of that runs on AWS Lambda. This has a lot of really nice operational wins for us. You're not caring around lots of state. You get horizontal scalability. You don't have to worry about patching those services because AWS is doing that for you.

There is DynamoDB, which we use for identification. This gives us really, really awesome scalable performance for the identification algorithm. Dgraph, which is the only service in Grapl that is not AWS managed runs on AWS ECS. That's their Docker container orchestration. That, again, like gives you horizontal scaling, low operational costs. We've got an event bus system which is built on S3 events, which go to SQS. Every AWS Lambda in Grapl consumes from an SQS queue, and those queues are populated by events from S3.

Again, we've got this really nice horizontally scalable architecture. We have built-in retry logic, back-offs, things like that. A lot of work has gone into ensuring that Grapl is both extremely scalable and efficient while also staying very low cost operationally. I've seen a lot of security teams spend really millions of dollars even just on operational overhead. This could be even on AWS. They're running EC2 instances that they need to patch. They have to do debugging on big monolithic servers. They run into scaling limits. I have seen that happen so many times that going into Grapl, it was basically just a hard requirement to build as much of it into the system as being resilient and scalable.

[01:00:36] JM: Okay. Well, Colin, it's been real pleasure talking to you, and I wish you the best of luck with Grapl.

[01:00:41] CO: Thanks very much for having me on today. This was a pleasure.

[END OF INTERVIEW]

[01:00:52] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[END]