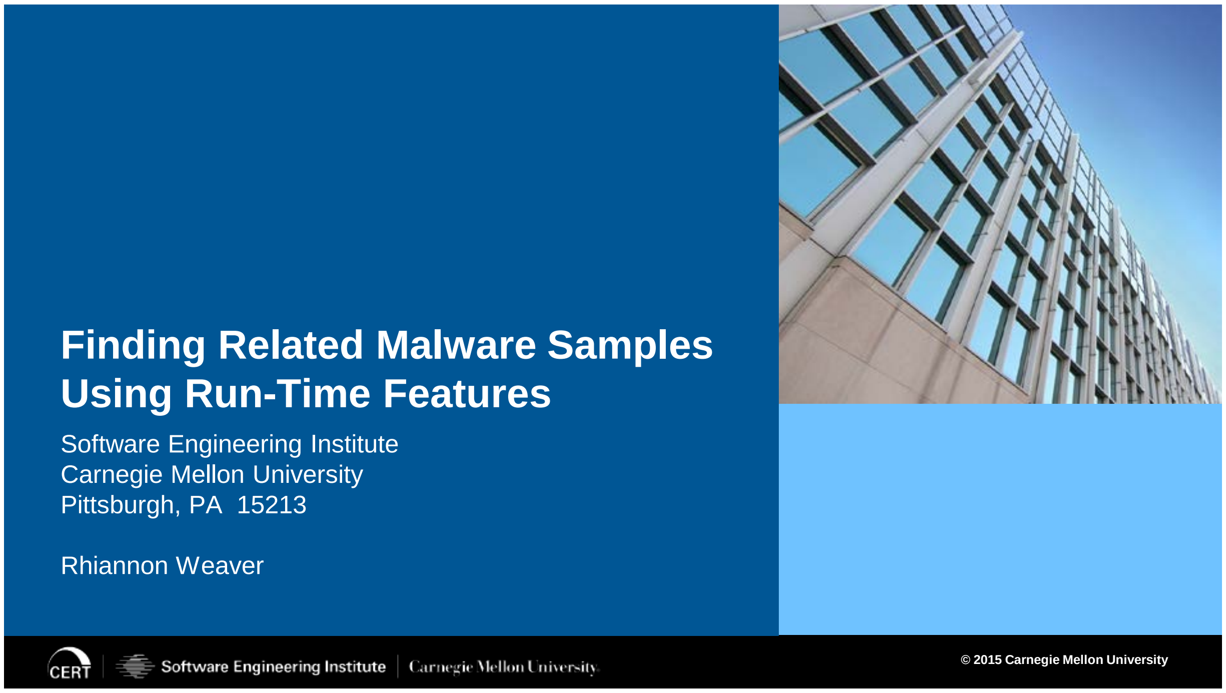
Finding Relationships Among Malware Samples Using Run-Time Features - Rhiannon Weaver

Table of Contents



Finding Related Malware Samples Using Run-Time Features

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Rhiannon Weaver

CERT | Software Engineering Institute | Carnegie Mellon University

© 2015 Carnegie Mellon University

**003 Announcer: We're going to move onto our next topic, which is going to be "Finding Relationships Among Malware Samples Using Run-Time Features" by Rhiannon Weaver. Rhiannon is a, Weaver, is a statistician who has been working in network security analysis since 2007. She received her BS in Computer Science and BS in Mathematics from Pennsylvania State University, and her MS and PhD in Statistics from Carnegie Mellon University. She's currently a member of the Technical Staff at CERT at the Software Engineering Institute.

And before we turn it over to Rhiannon, I just want to remind everybody, anybody that's logging in for the first time today to make sure to look at that Files tab at the bottom of the console where you could walk away with a PDF of each presentation here today, along with other cybersecurity-related work from SEI and CERT. And also, if you're just logging in and out from various presentations, make sure you fill out that survey as your feedback is always greatly appreciated. And lastly, for those of you on Twitter, be sure to follow @SEInews and follow the hashtag certcyber to follow along today. So now we'll turn it over to Rhiannon. Rhiannon, all yours. Welcome.

Presenter: Hi. Thanks, Shane. So today I just, I'm going to be talking, as Shane said, about finding related malware samples through using run-time features. A little bit about me. I'm a statistician by trade. I came into network security because they had sort of hard problems and interesting data sets to work on. As Shane mentioned, my background is in statistics. So today I guess I'm going to talk to you a little bit about some of those interesting data sets and hard problems.

Establishing Trust in Software

Establishing Trust in Software

Who are you?
What do you want to do?

(Are you lying about either?
Can I still trust you next week?)

How does this break down in the Cyber landscape?

- Unique identifiers of individuals
- Granularity of “identity”
- Rate of change

http://en.wikipedia.org/wiki/Jamie_Madrox

http://marvel.com/universe/Rogue

http://marvel.wikia.com/Raven_Darkholme_(Earth-11326)

CERT | Software Engineering Institute | Carnegie Mellon University
CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University
4

**004 So since we're talking about aligning with the COI initiatives, I think where my research aligns mostly is with trust. And so when I was thinking about trust and software I wanted to start thinking about, "Well, how do we trust individuals?" And generally with trust, it's kind of two-pronged. One is, like, "Who are you intrinsically? What is your identity?" And based on that identity, I can establish whether or not I think you're trustworthy. But also there's a component in there of, like, "What do you want to do?" right? Like, I might trust you to deliver a pizza but I might not trust you to deliver a secret document or something like that. So there's these two layers. :"Who are you?" and sort of more of behavioral, "What do you want to do?" And then with people, it's kind of like make sure that you're

not lying about either and you kind of
have to also think about how people
change and how that ability--sorry.
How basically as I said here, can I
still trust you next week? So I sort of
have to reinvest in my trust models.

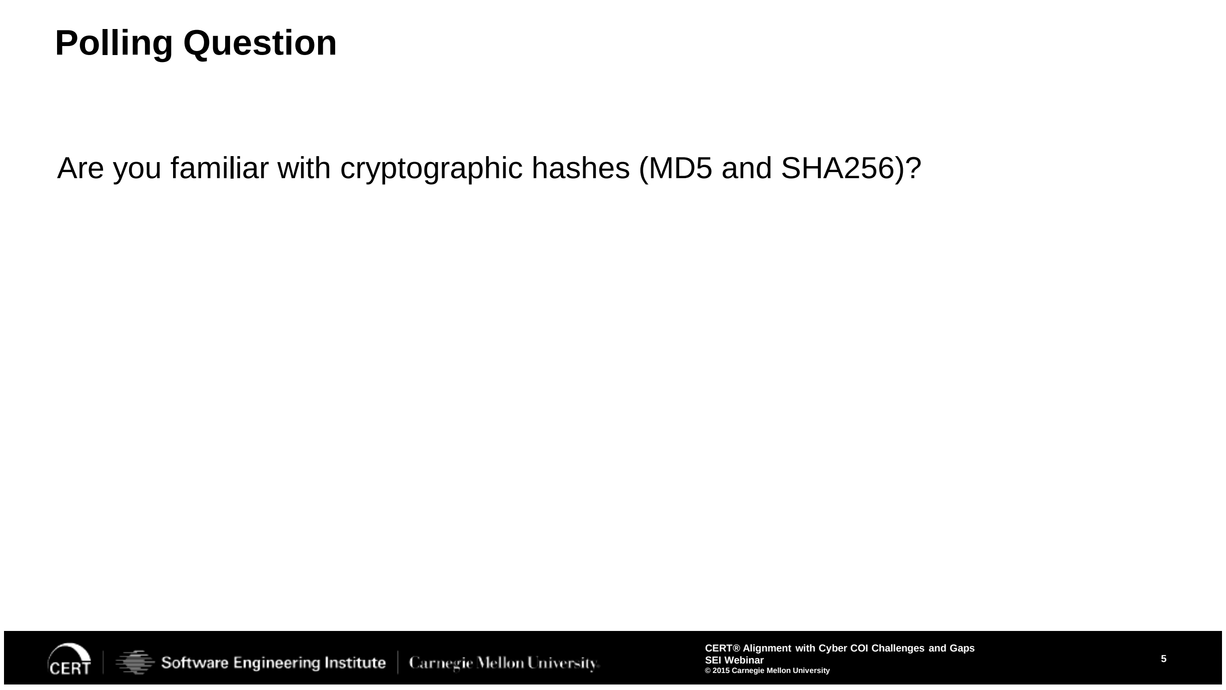
And then I was trying to think about
how does this break down in a cyber
landscape? And I have sort of three
points here, which is sort of unique
identifiers of individuals. It's kind of
harder to think about what makes an
individual, when you're thinking
about a cyberspace, because there's
nothing physical there, and sort of
the idea of the granularity that you
have of what identity really means.
And then of course in the cyber
landscape you also have a much
faster rate of change. And so if you
were thinking about, this is why I put
sort of my three individuals here, are
the kinds of individuals that you
might see in the cyber landscape.
And they are sort of very interesting
individuals from comic books. One is
the Multiple Man up at the top. So
he can basically just make copies of
himself all over the place. And
they're all sort of him, but, you know,
they're in different environments and
sometimes they can change. So if I
trust one of them, do I have to trust
all of them?

The second one at the top is Rogue.
She's, she has good intentions, but
any time she touches someone she
can basically absorb their life force.
If she absorbs too much, she almost
becomes them. So she's super powerful,
but she's also super vulnerable.

And then the last one is Mystique, who is the shape shifter. She has bad intentions most of the time, but she can basically change her form to look like someone who has good intentions. And the interesting thing about this is that these skills are inherent to these individuals. And they're subject more to the limits of human creativity than they are to limits in the human world. So if we're going to build trust models in the cyber world, we have to think about individuals who have these kinds of traits. And also these kinds of abilities and vulnerabilities.

So that brings us to talking about malware and what this really looks like when we're studying things like malicious files.

Polling Question



Polling Question

Are you familiar with cryptographic hashes (MD5 and SHA256)?

Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

5

**005 Announcer: Okay. So we

got our first polling question today from Rhiannon, and I know I'm going to butcher this, and she told me the right titles yesterday, but I'm going to try. Are you familiar with cryptographic hashes MD5 and SHA256? Okay.

Presenter: MD5 and SHA256.

Announcer: So let me launch that question. We'll give you about 15 or 20 seconds to vote, and then we'll come back to the results, so we can keep going on, Rhiannon.

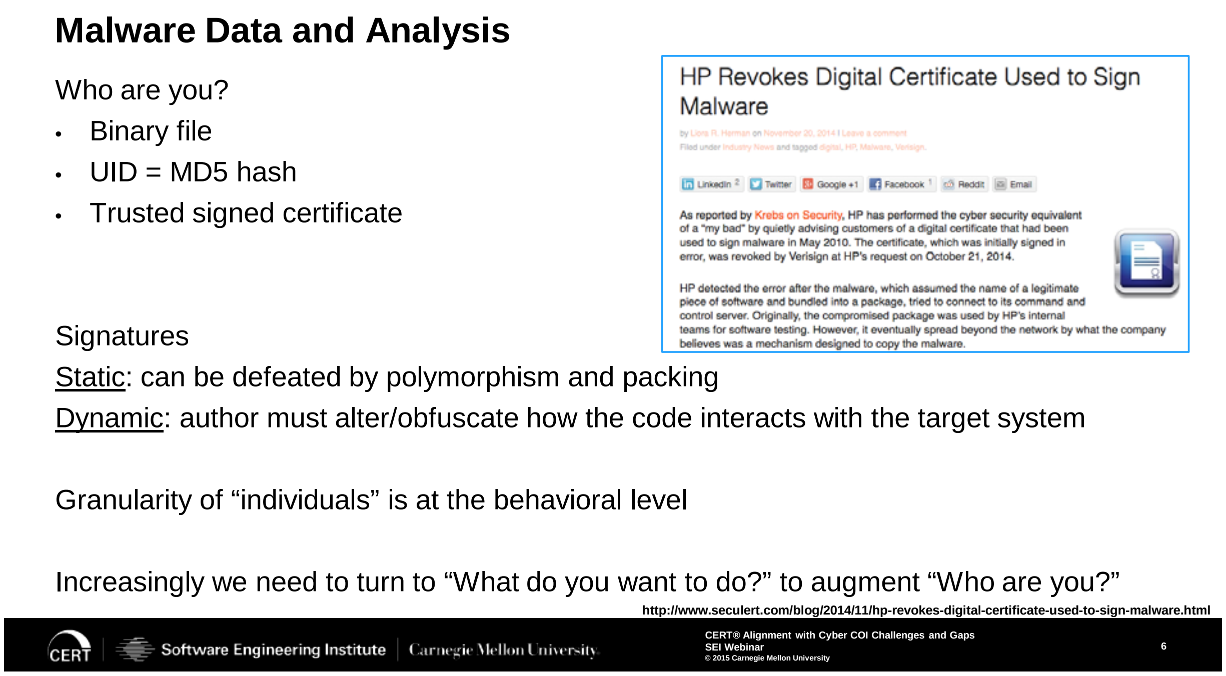
Let me preview here. This is going to give us an idea of what level detail she needs to go into. So about 65 percent are familiar, 35 percent, not. So if you can give us a quick overview maybe what they are?

Presenter: Sure. So cryptographic hashes are a way of uniquely identifying pieces of information. So if--they're mathematical formulas, so if you have two pieces of information, binary files, and they have the same hash, you know you're looking at the same thing. The important thing about hashes currently is that they're sort of the lingua franca of how people share information about malware through ops and blogs and stuff like that. They'll share hashes of files. And then also they're not particularly what we would call distance preserving. So a hash, if you change one little thing in a file, you know, one or two bits, the hashes could be completely different. And generally you see them written as hexadecimal code, so--or a list of hexadecimal digits. The MD5 has 32 of these, and the SHA256 has 64. So that's basically the important things about cryptographic hashes.

Announcer: Okay. Great.

Presenter: All right.

Malware Data and Analysis



Malware Data and Analysis

Who are you?

- Binary file
- UID = MD5 hash
- Trusted signed certificate

HP Revokes Digital Certificate Used to Sign Malware

by Liora R. Herman on November 20, 2014 | Leave a comment

Filed under Industry News and tagged digital, HP, Malware, Verisign.

As reported by Krebs on Security, HP has performed the cyber security equivalent of a "my bad" by quietly advising customers of a digital certificate that had been used to sign malware in May 2010. The certificate, which was initially signed in error, was revoked by Verisign at HP's request on October 21, 2014.

HP detected the error after the malware, which assumed the name of a legitimate piece of software and bundled into a package, tried to connect to its command and control server. Originally, the compromised package was used by HP's internal teams for software testing. However, it eventually spread beyond the network by what the company believes was a mechanism designed to copy the malware.

Signatures

Static: can be defeated by polymorphism and packing

Dynamic: author must alter/obfuscate how the code interacts with the target system

Granularity of "individuals" is at the behavioral level

Increasingly we need to turn to "What do you want to do?" to augment "Who are you?"

http://www.seculert.com/blog/2014/11/hp-revokes-digital-certificate-used-to-sign-malware.html

CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

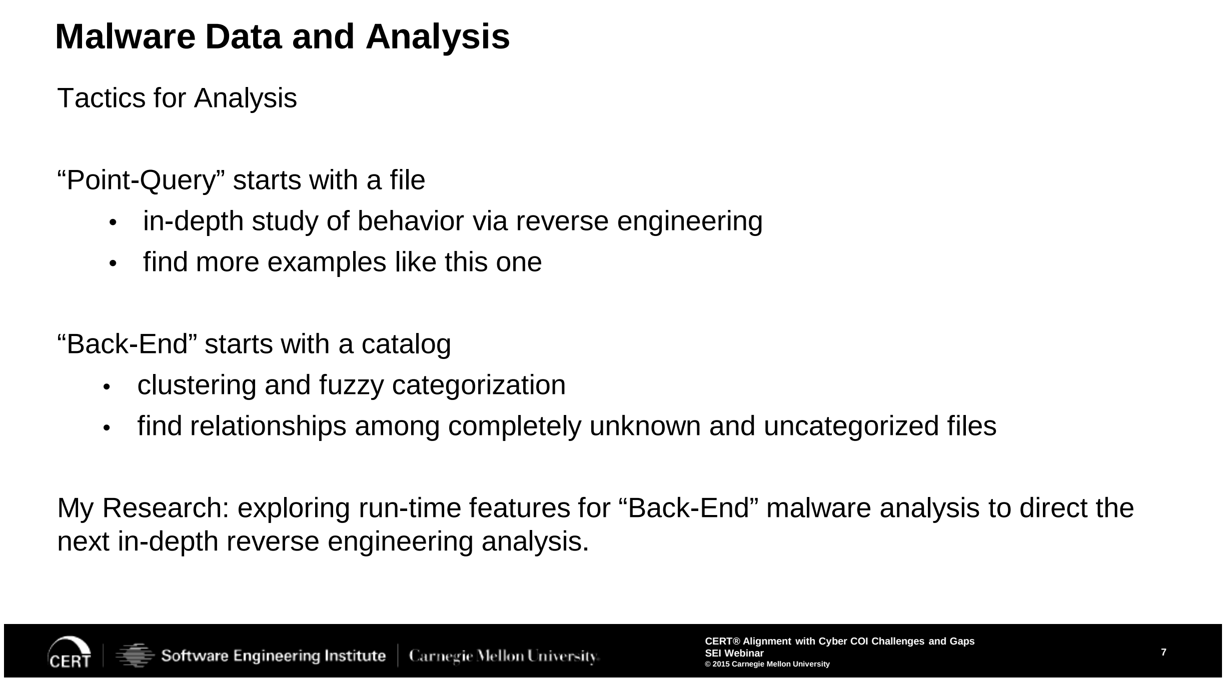
6

**006 So talking about malware data and analysis. Back to our questions of identity and individuals. Who are you when we want to talk about malware data? A malware, basically you have a binary file. Strings of code, ones and zeros. And they identify these uniquely by this MD5 hash. Or a SHA256 hash, for example. So our unique identifier for the individual is this hash. And that's how we know that we're looking at the same type of--the same information, for example. The other thing that comes up with trust is also for software that you can do trusted science certificates. And the difficulty with these is that it's very, you know, you have to maintain that chain of command for the--or chain of trust for certificates. So, you know, the BIOS signs the operating system and the operating system signs the et cetera. Oh, all the way up to who trusted this file? The difficult thing with that is that there have been cases where digital certificates of trust have been used to sign malware. They have to revoke them, and these certificates can also be stolen. So the question there is, you know, it's basically a very large way of doing whitelisting. But I still think we still need to do some sort of blacklisting in order to augment this and understand what the features are, why we trust these pieces of software and which ones we don't.

So when we talk about trying to understand identity, we also talk about developing signatures for different types of files. So static signatures are signatures that are built based on that binary file, that static file that's just a string of ones and zeroes. They can be pretty easily defeated by things like polymorphism, for example. As I said, when you change one little thing in the file, right, the hashes will be completely different. And so you've got two different individuals, but they're essentially the same thing. Also what's called packing, which is basically ways of taking a file and compressing it in ways that make it difficult to unpack it and learn what its features are.

There's also dynamic signatures, which are based more on features that you get when you actually look at this file in the run-time environment, so... And this is getting more toward the, "What are you trying to do?" aspect of trust than the "Who are you?" in terms of it's taking, you know, basically putting in a little DMZ and running it for a little bit and seeing, "Okay. Is it actually doing what it's saying it's doing?" And what makes it more difficult for malware authors, in that case, is that they have to alter or obfuscate how the code interacts with the target system, in that case. So granularity of individuals, when you're talking about cyber and you're talking about software, is more intertwined with the behavioral levels than it would be in the physical world. And so increasingly we need to turn to more of this, "What do you want to do?" in order to augment, "Who are you?"

Malware Data and Analysis



Malware Data and Analysis

Tactics for Analysis

“Point-Query” starts with a file

- in-depth study of behavior via reverse engineering
- find more examples like this one

“Back-End” starts with a catalog

- clustering and fuzzy categorization
- find relationships among completely unknown and uncategorized files

My Research: exploring run-time features for “Back-End” malware analysis to direct the next in-depth reverse engineering analysis.

CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

7

**007 So for malware data and analysis, I sort of break it down into two different types of tactics. And I call them point-query and back-end. And sometimes they intertwine a little bit, but in general a point-query starts with a file. I get something, you know, "We found this on the computer," or something like that. What is this thing actually trying to do, and finding more examples like it. So it starts with a file and then it expands out and tries to find more things. And then another one would be sort of more of a back-end analysis, which might start with, for example, a very large catalog of suspicious files and say, "Which ones of these are related?" So you would do more sort of clustering and fuzzy categorization and then try and find relationships among these unknown and uncategorized files. And where my research falls at the moment is sort of exploring these dynamic run-time features in the larger back-end analysis to help direct where the more focused inquiries can go.

So with that I'm going to talk a little bit about--

The CERT Artifact Catalog and Run-Time Environment

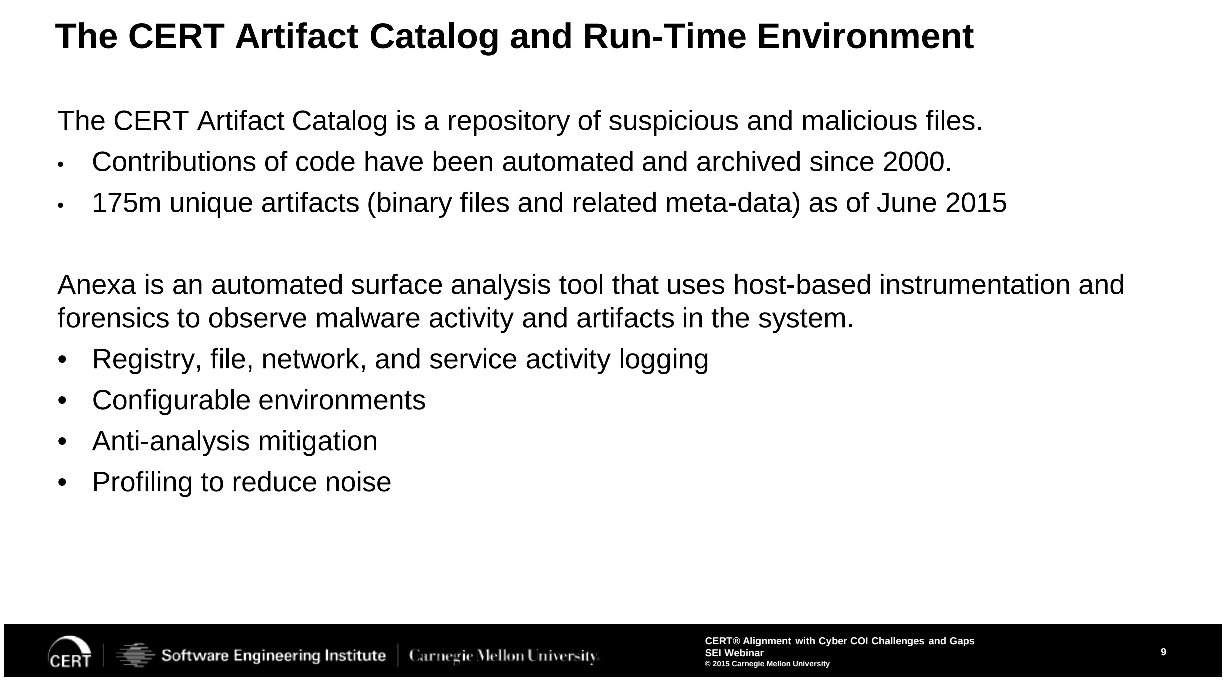
The CERT Artifact Catalog and Run-Time Environment

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

8

**008 --our artifact catalog and our run-time environment. So let's see.

The CERT Artifact Catalog and Run-Time Environment



The CERT Artifact Catalog and Run-Time Environment

The CERT Artifact Catalog is a repository of suspicious and malicious files.

- Contributions of code have been automated and archived since 2000.
- 175m unique artifacts (binary files and related meta-data) as of June 2015

Anexa is an automated surface analysis tool that uses host-based instrumentation and forensics to observe malware activity and artifacts in the system.

- Registry, file, network, and service activity logging
- Configurable environments
- Anti-analysis mitigation
- Profiling to reduce noise

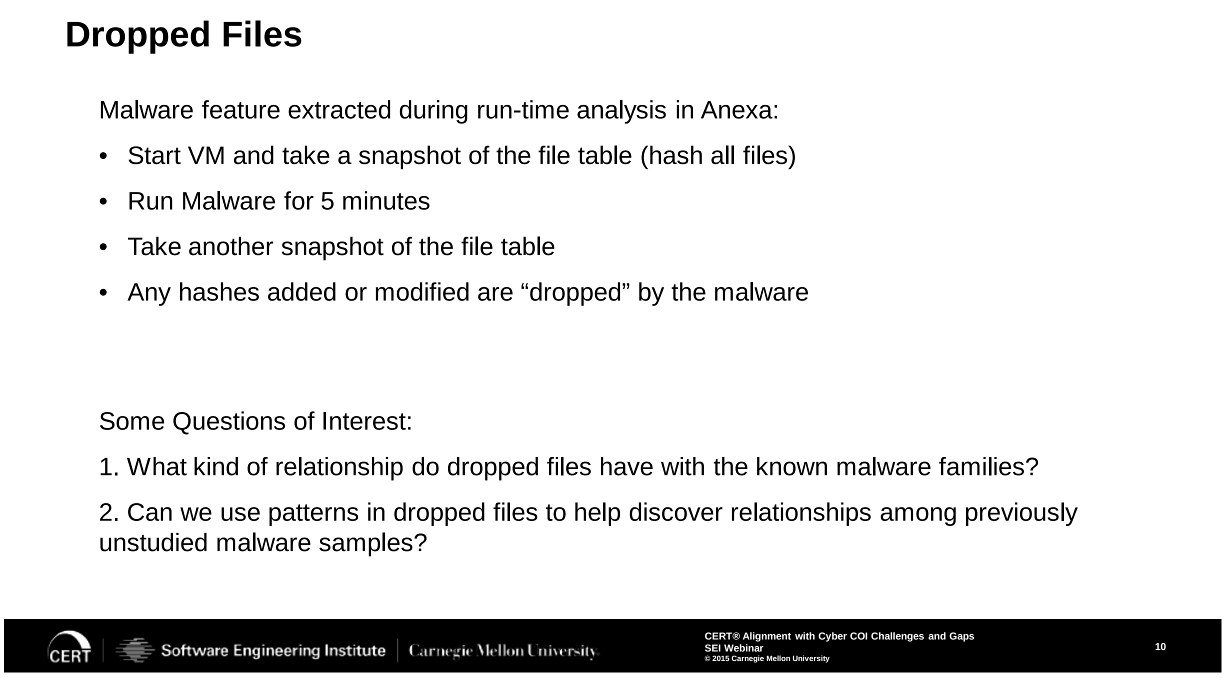
CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

9

**009 Here we go. The CERT artifact catalog and run-time environment. So the artifact catalog has--we've been compiling suspicious code. It's not necessarily all malicious, but it's all suspicious. Automated since about 2000, so that's about 15 years. And there are, as of this year, June, about 175 million unique artifacts in the catalog. And that's unique by these types of hashes, for example. So what happens is when you get, you might get a lot of variance of similar files or you might get a lot that are either one version is packed and one's not, so you get a lot of replication and you get a lot of uncertainty in these large catalogs. And we've been doing static analysis on malware in that sort of point-query side of things for a while. And what I've been trying to do is we have a run-time environment that we can take and run the files through and get some more dynamic features. And I'm trying to marry these two things together. So the--our automated run-time environment is called Anexa. It uses host-based instrumentation and forensics to observe malware activity and artifacts in the system. So it does things like cataloging registry keys, files that are touched, network and service touch points. It has configurable environments. It has a little bit of anti-analysis mitigation and what we call profiling to reduce noise. So trying to focus on the specifics of that file and not necessarily more system-type features.

Dropped Files



Dropped Files

Malware feature extracted during run-time analysis in Anexa:

- Start VM and take a snapshot of the file table (hash all files)
- Run Malware for 5 minutes
- Take another snapshot of the file table
- Any hashes added or modified are “dropped” by the malware

Some Questions of Interest:

1. What kind of relationship do dropped files have with the known malware families?
2. Can we use patterns in dropped files to help discover relationships among previously unstudied malware samples?

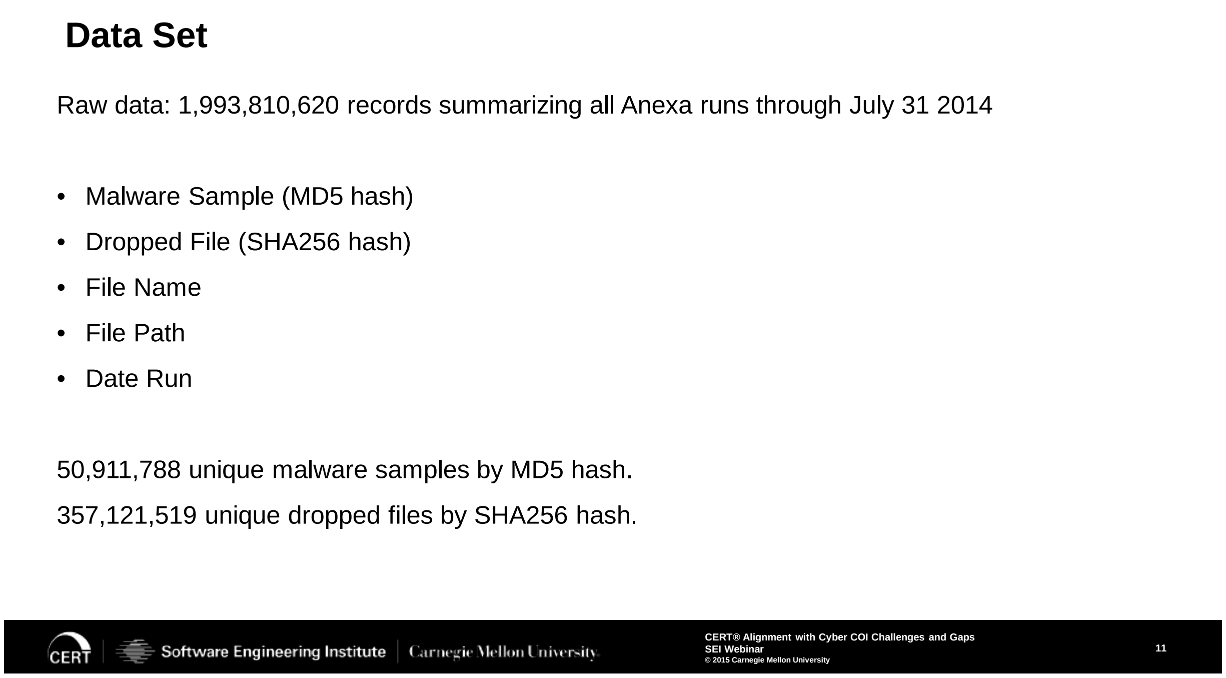
CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

10

**010 The particular feature that I was studying is what we call a dropped file. So this is a feature extracted during run-time analysis in Anexa, and basically what we do is we start up the Anexa environment, the virtual machine, and we take a snapshot of the file table, which basically is a--we get a hash of every file that's in that table. We run the malware for five minutes, and then we take another snapshot of the file table. Any hashes that are added or modified in that, in--sorry--between those two snapshots we consider dropped by the malware. So we call them dropped files. And some questions of interest that we've been looking at. One, what kind of relationship do these files have with malware families that we know about? And two, can we use patterns in these files to help discover relationships amongst unknowns and previously unstudied samples?

Data Set



Data Set

Raw data: 1,993,810,620 records summarizing all Anexa runs through July 31 2014

- Malware Sample (MD5 hash)
- Dropped File (SHA256 hash)
- File Name
- File Path
- Date Run

50,911,788 unique malware samples by MD5 hash.

357,121,519 unique dropped files by SHA256 hash.

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

11

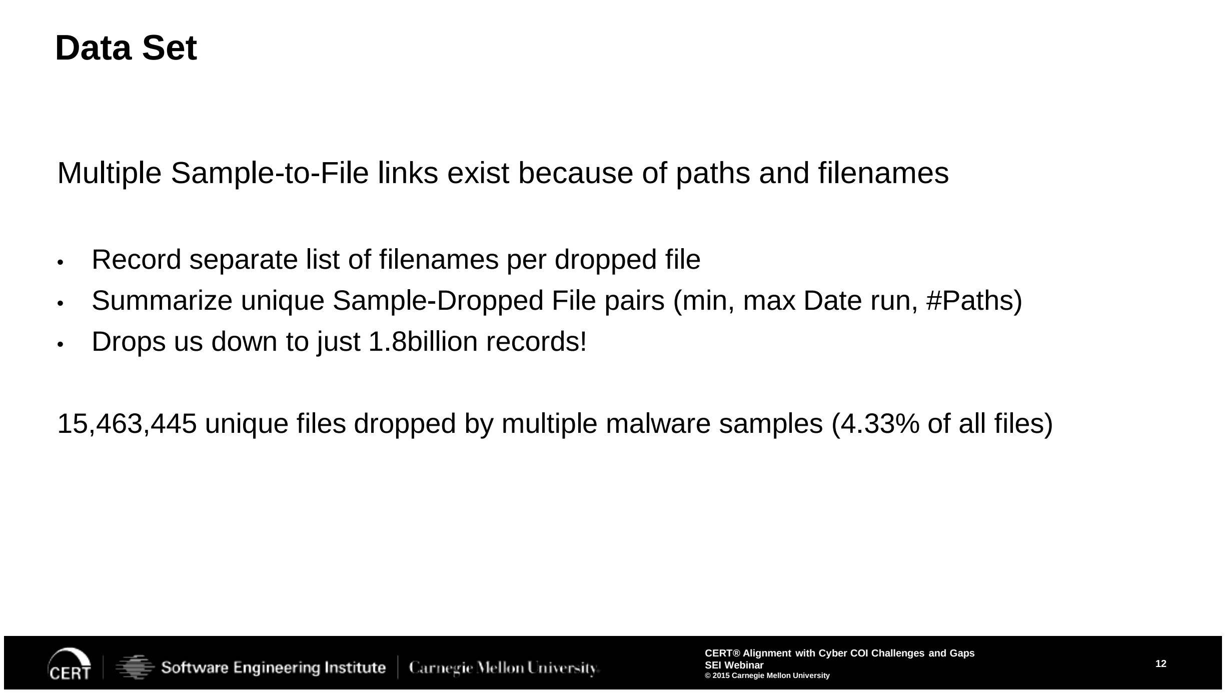
**011 So my data set. This I did last summer, so this is up through July 31st of last year. Well, I did it last summer. Last fall. So I had just under two billion records that is summarizing the dropped files for all of the malware samples that we had run through Anexa. And that was about, as it says here, about 50 million. We ran about 50 million of our at that time 150 million we had run through Anexa.

So what we got was we have the malware sample itself. And its unique identifier is, again, an MD5 hash. And then the dropped file is, was, hashed with the SHA256. And now that means that there could be in some cases, a malware could drop another sample that showed up on the malware sample side, but since the SHA256 and the MD5 aren't the

same, we won't see this. So
essentially that means that our
relationships are--I have one set over
here of malware samples and I have
one set over here of dropped files,
and all of my links go between these
two sets. That's known as a bipartite
graph.

So what I got was the malware
sample, the dropped file. I got a file
name, a file path, and the date it was
run. And this was, as I said, about
51 million unique malware samples,
and that was 357 million unique dropped
files by the SHA256 hash. So it was
pretty big data set. I got it as a 350
gig text file. So was kind of fun
working through it.

Data Set



Data Set

Multiple Sample-to-File links exist because of paths and filenames

- Record separate list of filenames per dropped file
- Summarize unique Sample-Dropped File pairs (min, max Date run, #Paths)
- Drops us down to just 1.8billion records!

15,463,445 unique files dropped by multiple malware samples (4.33% of all files)

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

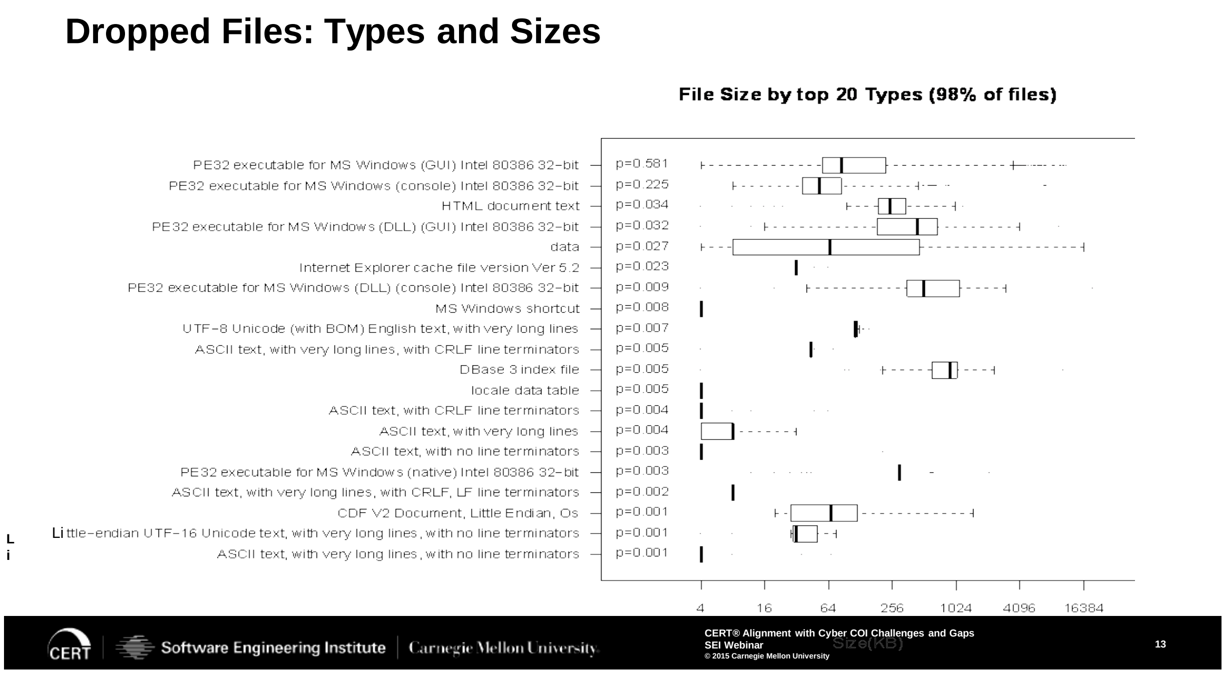
12

**012 Multiple sample-to-file links
existed because of paths and
filenames. And I wanted to sort of

reduce it down and say basically consider, only have one link, between a malware sample and the dropped file. So I basically took the filenames and separated them out and made a separate list of all of the different filenames that might've been associated with one particular dropped file. And then I summarized the unique samples, the unique pairs of sample and dropped file just by the number of paths, the dates, the number of dates that it was run, and the minimum and the maximum of the date that was run.

And from those 1.9 billion, that got me down to about 1.8 billion, so I'm doing pretty well here. But what's interesting about this is that almost every single file that was dropped was unique to the file that dropped it. So almost every single dropped file was unique to the malware sample that dropped it. Only 15 million of my 357 million were dropped by more than one sample. And so this is sort of coming up with that idea of, "How unique is a hash for identity?" Because probably all of these files are not completely unique. It's just that the identifiers are--that the granularity of the identifier is not quite what we want. But that's fine. Fifteen million is still quite a bit. And there's still quite a bit of relationships that I can explore from these 15 million I should also say that those 15 million files were shared among 40 million of the 50 million malware hashes. So there's still a lot of connections going on in this graph, even though most of the dropped files are what we would think of as a leaf node.

Dropped Files: Types and Sizes

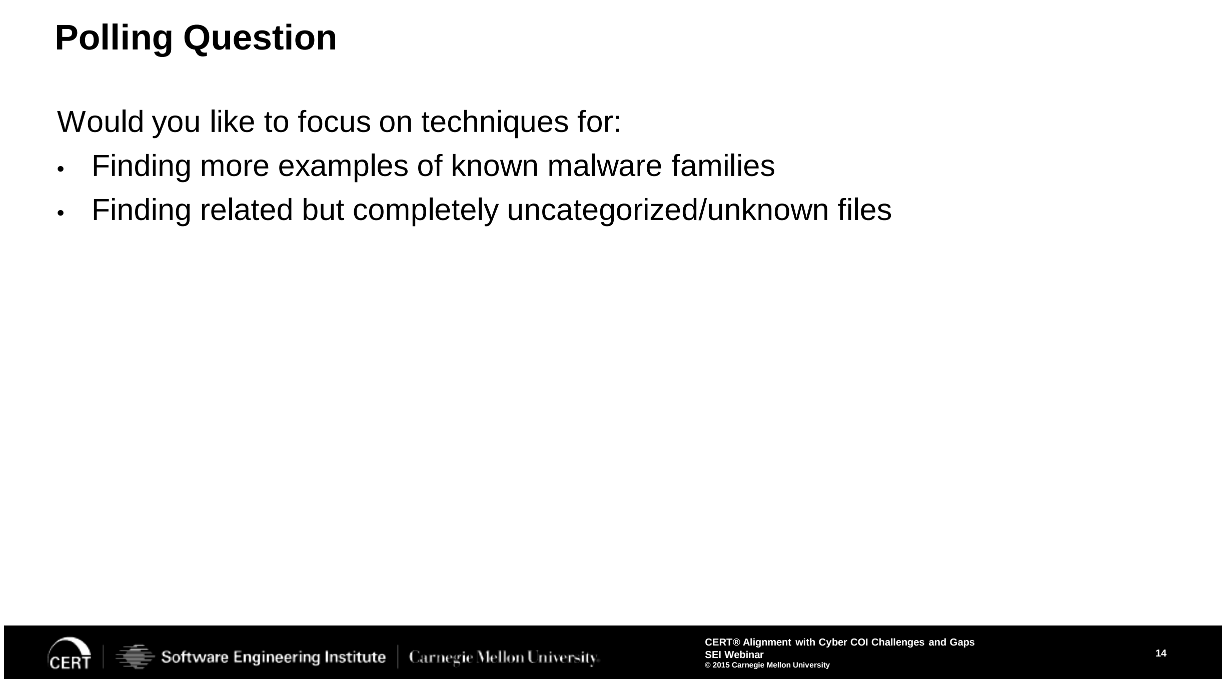


**013 I did a little bit of just some summarizing of what the files looked like. So this is a set of box plots. And you'll notice that the x-axis on this is on the log scale. So it's log-based too for the file size. Right. So these are actually very skewed distributions with very, with a few, small sizes. I'm sorry. A few very, very large sizes and then small sizes are sort of dominating them. But this is just I took about 20,000 of them and--a random sample of 20,000 of these dropped files and just ran the file command on them in UNIX to see what they were. Most of them are executables for MS Windows. That P there is the proportion in my sample. So 58 percent of them. So the GUI versus console takes up about--60--

80 percent of them. So most of them are that. And then the file sizes you'll notice are pretty--they're pretty wide ranging of those. And this is just the top, the top 20, which represented, like, 98 percent of all the files that I had.

So just sort of to get a snapshot of what this data set looks like in terms of what these files really are. And from these files, now I want to start thinking about relationships to what I know and then relationships to possibly completely new avenues. So I think that brings us to--

Polling Question



Polling Question

Would you like to focus on techniques for:

- Finding more examples of known malware families
- Finding related but completely uncategorized/unknown files

CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

14

**014 Announcer: Okay. Our next polling question, which is going to be on your screen now. We'd like to know, would you like to focus on techniques for finding more examples of known malware families or finding related but completely uncategorized/unknown files? So this will help us drive the flow a little bit. I think you're going to cover both topics, just---

Presenter: Yeah if we have time.

Announcer: --where we--

Presenter: Looks like I'm going pretty fast, so...

Announcer: Yeah. Where we may spend more of our time will depend on how you answer. So we'll wait about another 10 seconds here, and we'll go with our results.

And we're going to go with, at 75 percent right now, Rhiannon, we got finding related but completely uncategorized unknown files.

Presenter: All right.

Announcer: Okay. Back to you.

Presenter: Well, that's the second bit, so...

Announcer: Okay. We'll go to the first part then.

Leveraging Knowns: Finding “More Like X”

Leveraging Knowns: Finding “More Like X”

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

15

**015 We'll spend more time there maybe.

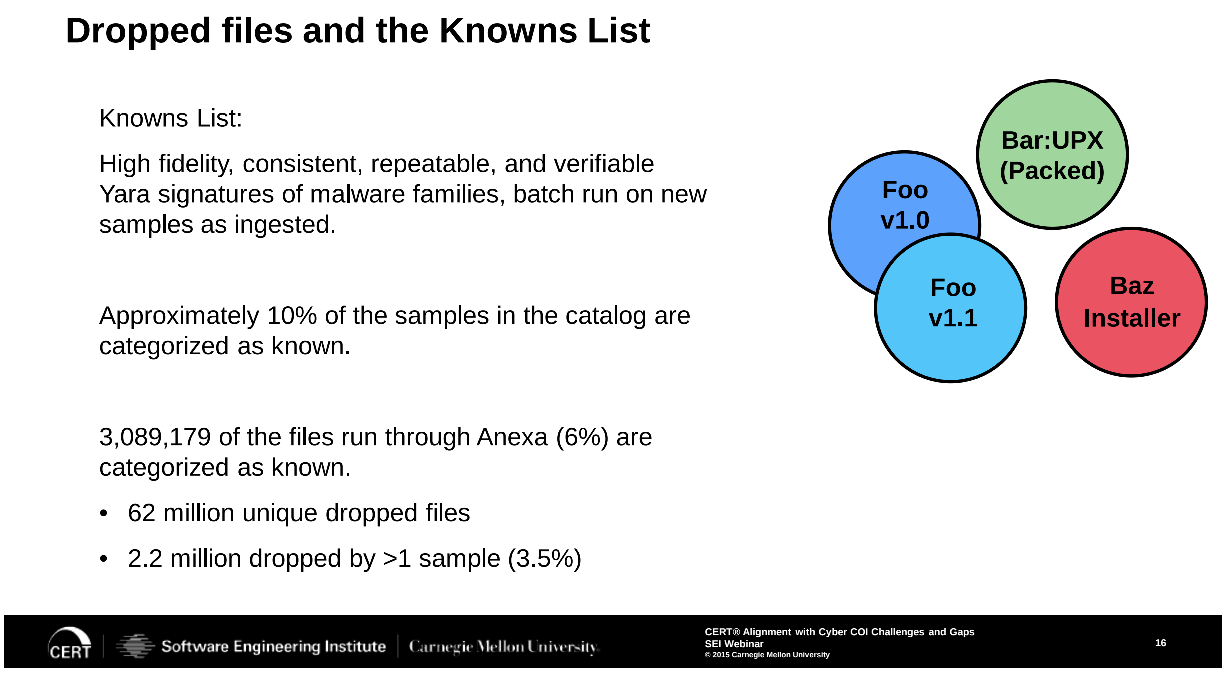
Presenter: We will--right.

Announcer: Yeah.

Presenter: So leveraging knowns is sort of finding "more like X." This is actually a pretty simple analysis but powerful, which is--I think that a lot of times people focus on the cool whiz-bang machine-learning algorithms you can do. But if you just sort of start thinking about what really makes sense, sometimes you can get away with a lot simpler things.

So leveraging knowns.

Dropped files and the Knowns List



Dropped files and the Knowns List

Knowns List:

High fidelity, consistent, repeatable, and verifiable Yara signatures of malware families, batch run on new samples as ingested.

Approximately 10% of the samples in the catalog are categorized as known.

3,089,179 of the files run through Anexa (6%) are categorized as known.

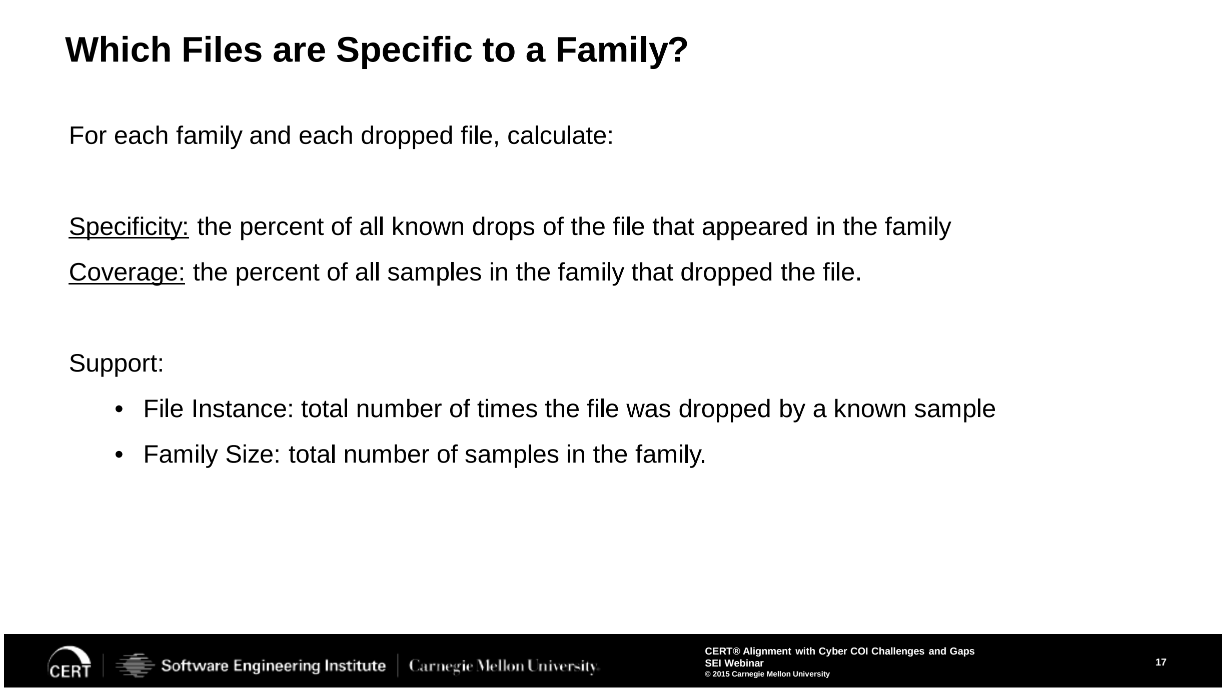
- 62 million unique dropped files
- 2.2 million dropped by >1 sample (3.5%)

**016 We have, I call, it's called, the knowns list.

This is the list of known files that we have. And it's a high fidelity, consistent, repeatable and verifiable signatures of malware families that are batch run on new samples that are ingested. And this is from static analysis. So there's a program called Yara that you can use to write signatures that describe signatures that are found in the static analysis. So these are sort of results of point-query and sort of expanding on those point query type things. And as I say here, names here have been changed to protect the guilty in this case. But the family names here, I'm going with some aliases, but... So you get different versions of the same kind of family, like foo_version 1 and foo_ version 1.1. Sometimes you get packed versus unpacked, right? So this one is a family that we're seeing is packed a particular packer called UPX. And then different types of things like, you know, the baz installer versus the baz dropper, versus et cetera. So these are the types of names that can, we can find for known files. So approximately 10 percent of the samples in our catalog are categorized as known.

When I looked at my data, so those 50 million that had been run through Anexa, about 3 million of them are categorized as known. So it's about six percent. It's a little less than the overall catalog. They shared, these 3 million, shared 62 million unique dropped files, and again, really small percentage, 3.5 percent, 2.2. million, were dropped by more than one file. But I can still take those 2.2 million and then learn which ones might be connected with particular families. With some confidence.

Which Files are Specific to a Family?



Which Files are Specific to a Family?

For each family and each dropped file, calculate:

Specificity: the percent of all known drops of the file that appeared in the family

Coverage: the percent of all samples in the family that dropped the file.

Support:

- File Instance: total number of times the file was dropped by a known sample
- Family Size: total number of samples in the family.

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

17

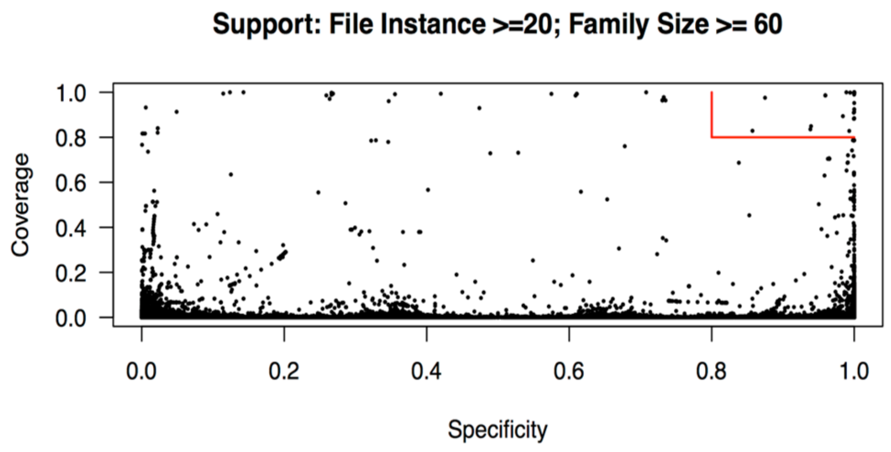
**017 So for each family that I have and for each dropped file I can calculate two metrics. One is the specificity, which is the percent of all known drops of that file that appeared in the family. So how specific is this dropped file to this family? If it shows up 50 percent of the time in baz and 50 percent of the time in foo, then it's not very specific to either one. However, if it shows up 100 percent of the time it shows up, it shows up in foo, then it's pretty linked with that family foo.

The other thing is coverage, which is the percent of all samples in that family that drop the file. So this is sort of the how, you know, if you get into the math of things you have one-to-one and the you have onto. This is the onto side. So when it showed up, what percentage of all of that family did it show up in? So say it's a very specific file, right, and then 1100 percent of the time it showed up in family foo, but it only showed up in 50 percent of the samples in family foo. Then that has 50 percent coverage. However, if I have things that are close to one, 100 percent on both of these metrics, then I have a file, a particular SHA256 unique hash of a file, that is almost one-to-one with the families that I have categorized here.

I also want to notice, mention here, that you have this notion of support, which is how many times did the file actually, did that dropped file, actually show up? So if a dropped file only shows up one or two times it can have, like, a really high specificity but I don't care about it as much. So file instance would be the total number of times that the file was dropped by any sample in the knowns. And then the family size, so on the other side, if a family only has a couple of representatives, then a file could have real easy coverage of it, but I don't really have a lot of information. So the total number of samples in the family, these two things are sort of measures of the support of this, of these metrics in the population.

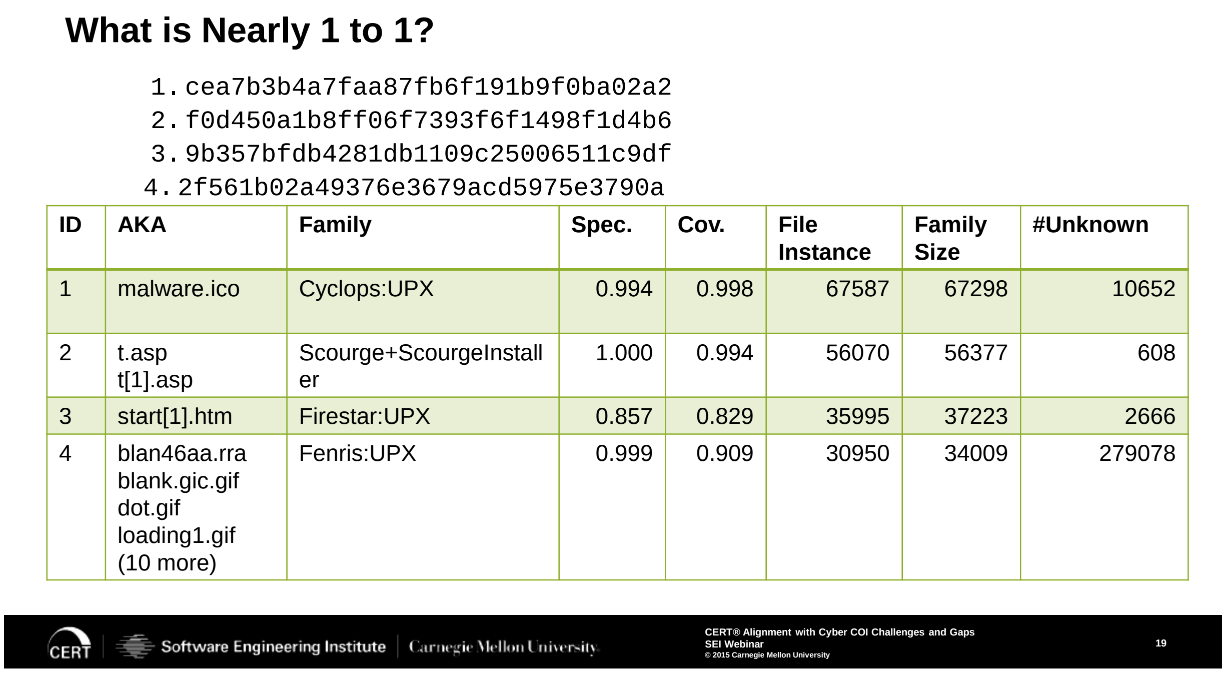
Which Files are Specific to a Family?

Which Files are Specific to a Family?



**018 So here is a graph of the specificity and coverage of the files that I was looking at. So this is for support for the file instances greater than, equal to 20, for all families that had more than 60 malware samples. So files, dropped files that showed up more than 20 times in families that had 60 or more malware samples associated with them. And what I care about, as I said before, is these ones up in the right-hand corner, because these are the ones that are coming close to that one-to-one relationship. The red lines there are .8 specificity and .8 coverage that block. And there were 35 files in that upper left that had specificity and coverage for this support that were greater than .8.

What is Nearly 1 to 1?



What is Nearly 1 to 1?

1. cea7b3b4a7faa87fb6f191b9f0ba02a2
2. f0d450a1b8ff06f7393f6f1498f1d4b6
3. 9b357bfdb4281db1109c25006511c9df
4. 2f561b02a49376e3679acd5975e3790a

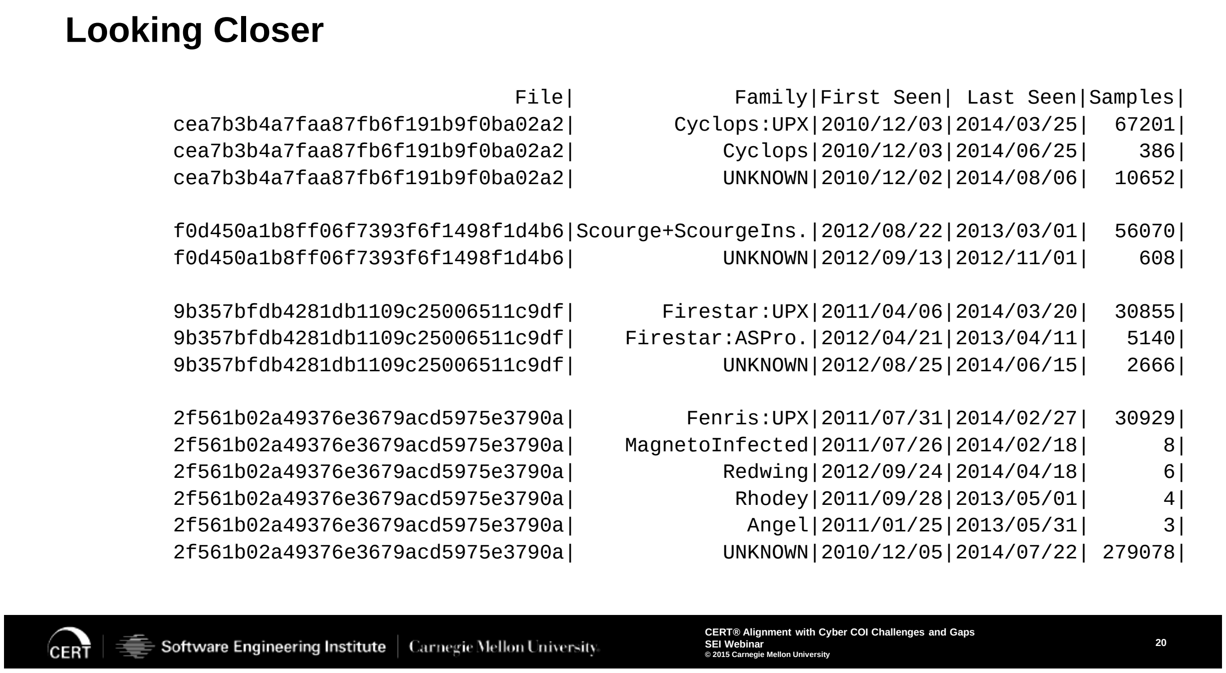
ID	AKA	Family	Spec.	Cov.	File Instance	Family Size	#Unknown
1	malware.ico	Cyclops:UPX	0.994	0.998	67587	67298	10652
2	t.asp t[1].asp	Scourge+ScourgeInstall er	1.000	0.994	56070	56377	608
3	start[1].htm	Firestar:UPX	0.857	0.829	35995	37223	2666
4	blan46aa.rra blank.gic.gif dot.gif loading1.gif (10 more)	Fenris:UPX	0.999	0.909	30950	34009	279078

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

19

**019 A couple of examples. So up here you can see these one through four. This is the first 32 bits of the SHA256 hash of the dropped file. As I said before, those hashes show up as hexadecimal, just strings of hexadecimal numbers. So this is the unique identifiers of these dropped files, and here's some things that I found out about them. So the first one, the filename is malware.icon. So that's a little bit troubling. It has really high specificity and really high coverage for this family Cyclops:UPX packed family called Cyclops. As I said, names have been changed to protect the guilty. So I had 67,000 of these Cyclops files. And almost all of them showed up with this particular file. But what's interesting here is that I also had almost 10,000 completely unknown files that hadn't been categorized by anything that have this one particular file in them. Same thing for two, three and four. I have really high specificity and coverage for these types of things, and also a whole bunch of files that are considered unknown that are in my catalog that I might be able to direct somebody and say, "Hey. If you want to find more like Cyclops, look at these 10,000 files."

Looking Closer



Looking Closer

```
                               File|            Family|First Seen| Last Seen|Samples|
cea7b3b4a7faa87fb6f191b9f0ba02a2|        Cyclops:UPX|2010/12/03|2014/03/25|  67201|
cea7b3b4a7faa87fb6f191b9f0ba02a2|            Cyclops|2010/12/03|2014/06/25|    386|
cea7b3b4a7faa87fb6f191b9f0ba02a2|            UNKNOWN|2010/12/02|2014/08/06|  10652|

f0d450a1b8ff06f7393f6f1498f1d4b6|Scourge+ScourgeIns.|2012/08/22|2013/03/01|  56070|
f0d450a1b8ff06f7393f6f1498f1d4b6|            UNKNOWN|2012/09/13|2012/11/01|    608|

9b357bfdb4281db1109c25006511c9df|       Firestar:UPX|2011/04/06|2014/03/20|  30855|
9b357bfdb4281db1109c25006511c9df|    Firestar:ASPro.|2012/04/21|2013/04/11|   5140|
9b357bfdb4281db1109c25006511c9df|            UNKNOWN|2012/08/25|2014/06/15|   2666|

2f561b02a49376e3679acd5975e3790a|         Fenris:UPX|2011/07/31|2014/02/27|  30929|
2f561b02a49376e3679acd5975e3790a|    MagnetoInfected|2011/07/26|2014/02/18|      8|
2f561b02a49376e3679acd5975e3790a|            Redwing|2012/09/24|2014/04/18|      6|
2f561b02a49376e3679acd5975e3790a|             Rhodey|2011/09/28|2013/05/01|      4|
2f561b02a49376e3679acd5975e3790a|              Angel|2011/01/25|2013/05/31|      3|
2f561b02a49376e3679acd5975e3790a|            UNKNOWN|2010/12/05|2014/07/22| 279078|
```

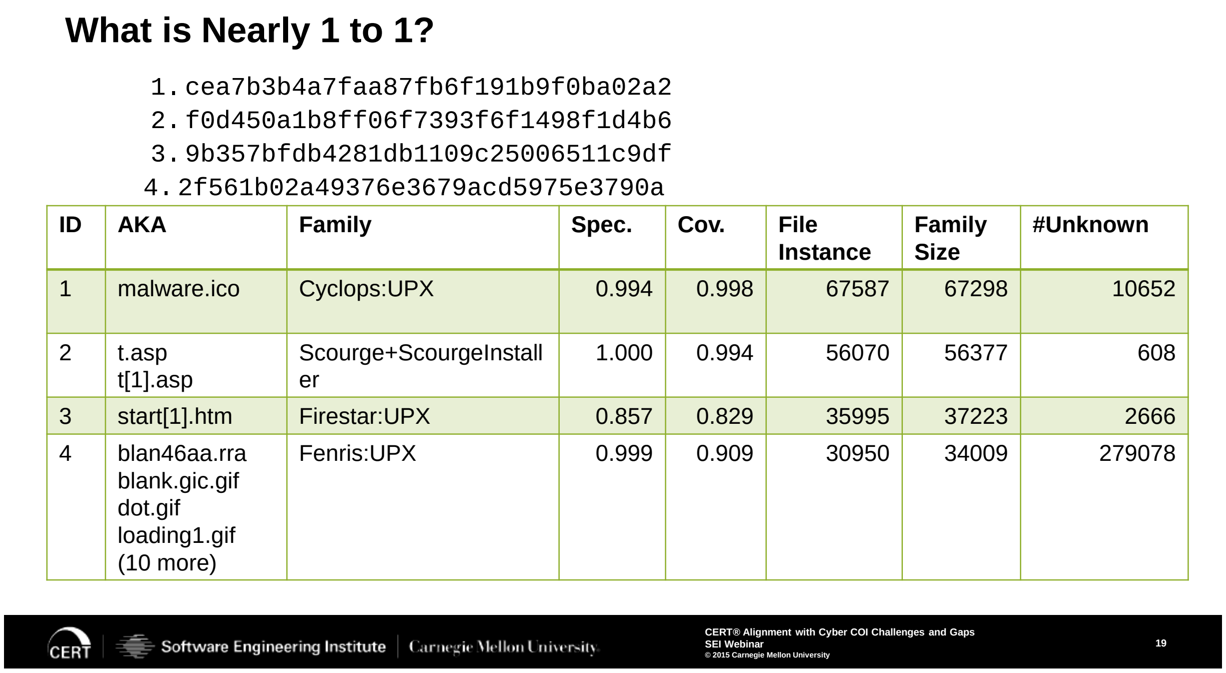
CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

20

**020 If I look closer and I try and understand why this specificity isn't completely 100 percent--so I look at that, let's see. Go back.

What is Nearly 1 to 1?



What is Nearly 1 to 1?

1. cea7b3b4a7faa87fb6f191b9f0ba02a2
2. f0d450a1b8ff06f7393f6f1498f1d4b6
3. 9b357bfdb4281db1109c25006511c9df
4. 2f561b02a49376e3679acd5975e3790a

ID	AKA	Family	Spec.	Cov.	File Instance	Family Size	#Unknown
1	malware.ico	Cyclops:UPX	0.994	0.998	67587	67298	10652
2	t.asp t[1].asp	Scourge+ScourgeInstaller	1.000	0.994	56070	56377	608
3	start[1].htm	Firestar:UPX	0.857	0.829	35995	37223	2666
4	blan46aa.rra blank.gic.gif dot.gif loading1.gif (10 more)	Fenris:UPX	0.999	0.909	30950	34009	279078

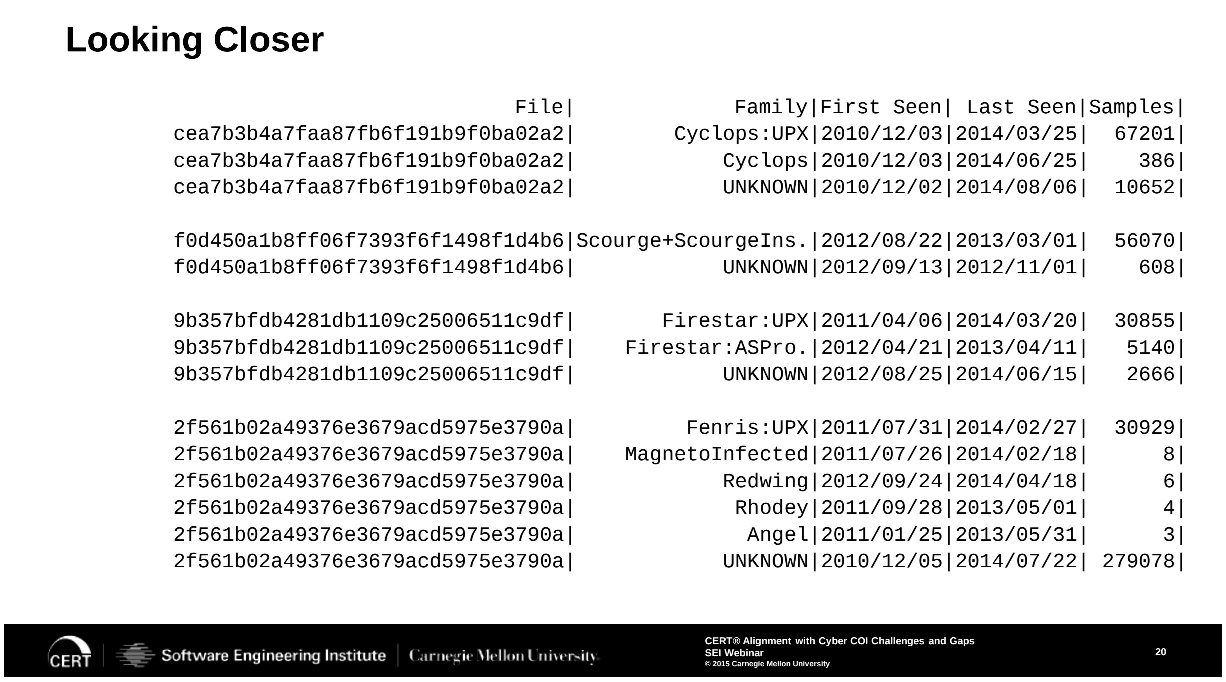
CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

19

**019 Right. The specificity is 99 percent and the coverage is 99 percent on these.

Looking Closer



Looking Closer

```
                              File|             Family|First Seen| Last Seen|Samples|
cea7b3b4a7faa87fb6f191b9f0ba02a2|         Cyclops:UPX|2010/12/03|2014/03/25|  67201|
cea7b3b4a7faa87fb6f191b9f0ba02a2|             Cyclops|2010/12/03|2014/06/25|    386|
cea7b3b4a7faa87fb6f191b9f0ba02a2|             UNKNOWN|2010/12/02|2014/08/06|  10652|

f0d450a1b8ff06f7393f6f1498f1d4b6|Scourge+ScourgeIns.|2012/08/22|2013/03/01|  56070|
f0d450a1b8ff06f7393f6f1498f1d4b6|             UNKNOWN|2012/09/13|2012/11/01|    608|

9b357bfdb4281db1109c25006511c9df|        Firestar:UPX|2011/04/06|2014/03/20|  30855|
9b357bfdb4281db1109c25006511c9df|     Firestar:ASPro.|2012/04/21|2013/04/11|   5140|
9b357bfdb4281db1109c25006511c9df|             UNKNOWN|2012/08/25|2014/06/15|   2666|

2f561b02a49376e3679acd5975e3790a|          Fenris:UPX|2011/07/31|2014/02/27|  30929|
2f561b02a49376e3679acd5975e3790a|     MagnetoInfected|2011/07/26|2014/02/18|      8|
2f561b02a49376e3679acd5975e3790a|             Redwing|2012/09/24|2014/04/18|      6|
2f561b02a49376e3679acd5975e3790a|              Rhodey|2011/09/28|2013/05/01|      4|
2f561b02a49376e3679acd5975e3790a|               Angel|2011/01/25|2013/05/31|      3|
2f561b02a49376e3679acd5975e3790a|             UNKNOWN|2010/12/05|2014/07/22| 279078|
```

CERT | Software Engineering Institute | Carnegie Mellon University

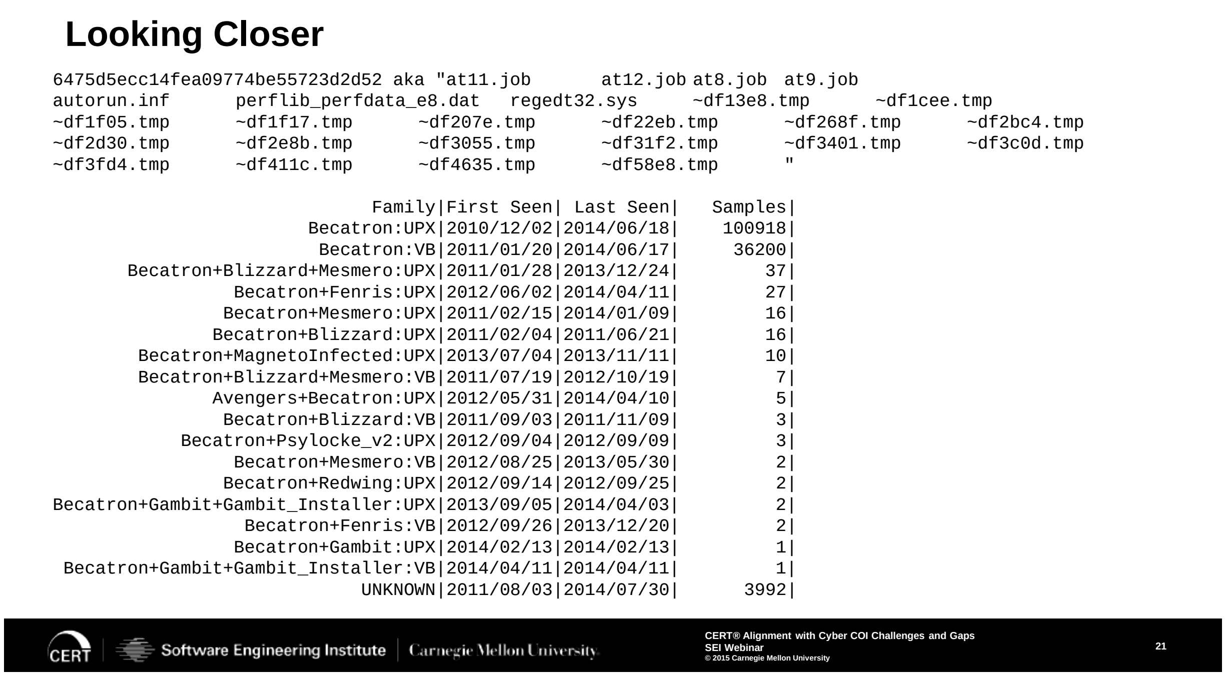
CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

20

**020 And when I look at the discrepancy in the specificity I see that it actually shows up the family is the same in this top row. So the file--the family is the same. Cyclops. Well, one of them is Cyclops:UPX packed and one of them is just Cyclops. And they've been run from 2010 to 2014, so this is a lot of different samples that have been run over a long period of time. And it's pretty solidly that this particular file is showing up on this family only. So I have these 10,000 files that I might be able to direct someone else and say, "Hey, these are ones you might want to look at if you're looking for more of these."

Same thing with the next two examples here, the Scourge and the Firestar. Right. One is Firestar, one of them's UPX packed, one of them's packed with ASProtect, but they're both basically the same. The other one, the last one here, the specificity looks a little bit stranger in that I've got 30,000 that are Fenris and then a couple of things creeping in from other types of families. But the other thing to know is that these families are all what are known as file infectors. So they might be just sort of changing the environment a little bit differently. But this also had 279,000 unknowns that we can start looking at.

Looking Closer



Looking Closer

```
6475d5ecc14fea09774be55723d2d52 aka "at11.job       at12.job at8.job  at9.job
autorun.inf      perflib_perfdata_e8.dat   regedt32.sys     ~df13e8.tmp      ~df1cee.tmp
~df1f05.tmp      ~df1f17.tmp      ~df207e.tmp      ~df22eb.tmp      ~df268f.tmp      ~df2bc4.tmp
~df2d30.tmp      ~df2e8b.tmp      ~df3055.tmp      ~df31f2.tmp      ~df3401.tmp      ~df3c0d.tmp
~df3fd4.tmp      ~df411c.tmp      ~df4635.tmp      ~df58e8.tmp      "

                              Family|First Seen| Last Seen|   Samples|
                        Becatron:UPX|2010/12/02|2014/06/18|    100918|
                         Becatron:VB|2011/01/20|2014/06/17|     36200|
        Becatron+Blizzard+Mesmero:UPX|2011/01/28|2013/12/24|        37|
                 Becatron+Fenris:UPX|2012/06/02|2014/04/11|        27|
                Becatron+Mesmero:UPX|2011/02/15|2014/01/09|        16|
               Becatron+Blizzard:UPX|2011/02/04|2011/06/21|        16|
         Becatron+MagnetoInfected:UPX|2013/07/04|2013/11/11|        10|
         Becatron+Blizzard+Mesmero:VB|2011/07/19|2012/10/19|         7|
               Avengers+Becatron:UPX|2012/05/31|2014/04/10|         5|
                Becatron+Blizzard:VB|2011/09/03|2011/11/09|         3|
            Becatron+Psylocke_v2:UPX|2012/09/04|2012/09/09|         3|
                 Becatron+Mesmero:VB|2012/08/25|2013/05/30|         2|
                Becatron+Redwing:UPX|2012/09/14|2012/09/25|         2|
Becatron+Gambit+Gambit_Installer:UPX|2013/09/05|2014/04/03|         2|
                  Becatron+Fenris:VB|2012/09/26|2013/12/20|         2|
                 Becatron+Gambit:UPX|2014/02/13|2014/02/13|         1|
 Becatron+Gambit+Gambit_Installer:VB|2014/04/11|2014/04/11|         1|
                             UNKNOWN|2011/08/03|2014/07/30|      3992|
```

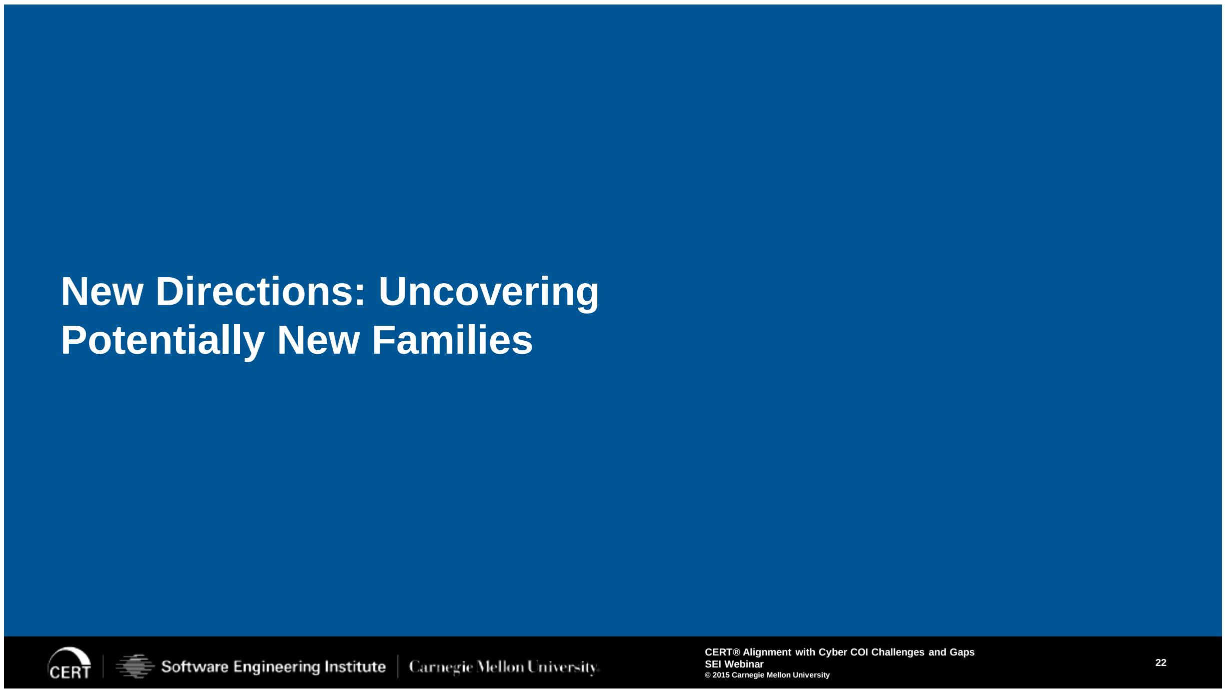
CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

21

**021 This is another example of how the different ways that you look at the families. This one had--this was right outside of my 80 percent specificity. But when I looked at them they were all versions of this Becatron family that sort of had different other types that were interspersed with it. And also, 4,000 unknowns that we can start looking at and say they're probably a version of this Becatron, so...

New Directions: Uncovering Potentially New Families



New Directions: Uncovering Potentially New Families

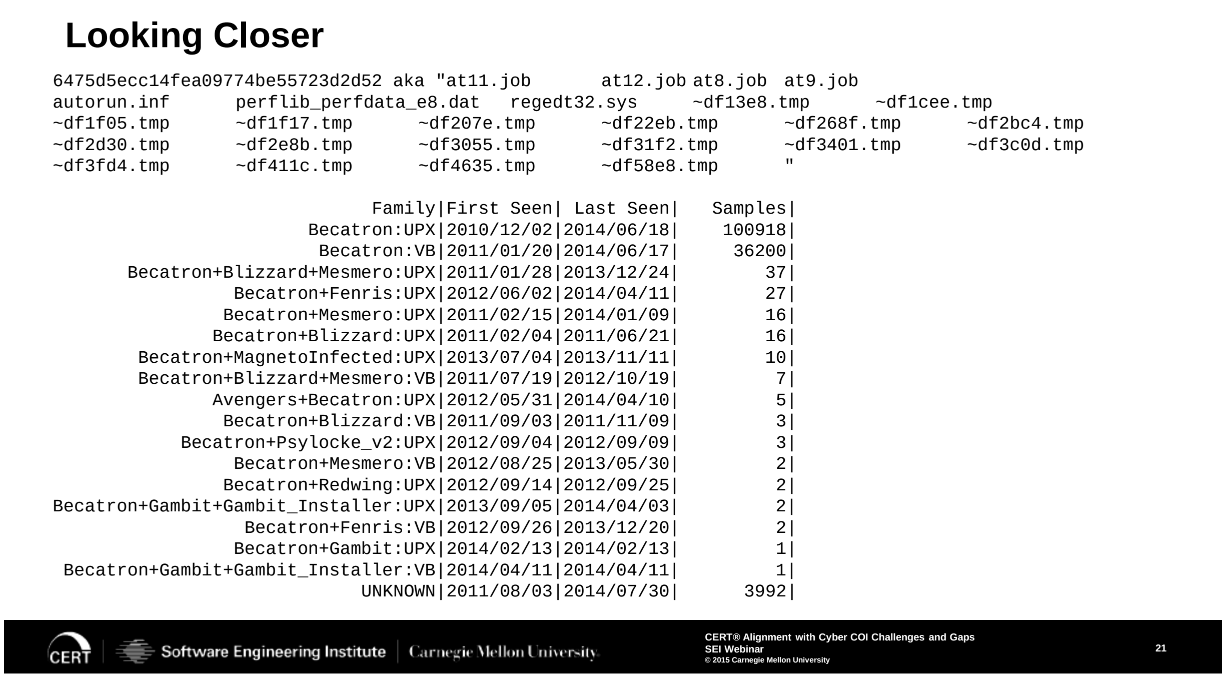
CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

22

**022 That's what I have for looking closer.

Looking Closer



Looking Closer

```
6475d5ecc14fea09774be55723d2d52 aka "at11.job       at12.job at8.job  at9.job
autorun.inf      perflib_perfdata_e8.dat   regedt32.sys     ~df13e8.tmp      ~df1cee.tmp
~df1f05.tmp      ~df1f17.tmp      ~df207e.tmp      ~df22eb.tmp      ~df268f.tmp      ~df2bc4.tmp
~df2d30.tmp      ~df2e8b.tmp      ~df3055.tmp      ~df31f2.tmp      ~df3401.tmp      ~df3c0d.tmp
~df3fd4.tmp      ~df411c.tmp      ~df4635.tmp      ~df58e8.tmp      "

                              Family|First Seen| Last Seen|   Samples|
                        Becatron:UPX|2010/12/02|2014/06/18|    100918|
                         Becatron:VB|2011/01/20|2014/06/17|     36200|
       Becatron+Blizzard+Mesmero:UPX|2011/01/28|2013/12/24|        37|
                 Becatron+Fenris:UPX|2012/06/02|2014/04/11|        27|
                Becatron+Mesmero:UPX|2011/02/15|2014/01/09|        16|
               Becatron+Blizzard:UPX|2011/02/04|2011/06/21|        16|
        Becatron+MagnetoInfected:UPX|2013/07/04|2013/11/11|        10|
        Becatron+Blizzard+Mesmero:VB|2011/07/19|2012/10/19|         7|
               Avengers+Becatron:UPX|2012/05/31|2014/04/10|         5|
                Becatron+Blizzard:VB|2011/09/03|2011/11/09|         3|
            Becatron+Psylocke_v2:UPX|2012/09/04|2012/09/09|         3|
                 Becatron+Mesmero:VB|2012/08/25|2013/05/30|         2|
                Becatron+Redwing:UPX|2012/09/14|2012/09/25|         2|
Becatron+Gambit+Gambit_Installer:UPX|2013/09/05|2014/04/03|         2|
                  Becatron+Fenris:VB|2012/09/26|2013/12/20|         2|
                 Becatron+Gambit:UPX|2014/02/13|2014/02/13|         1|
 Becatron+Gambit+Gambit_Installer:VB|2014/04/11|2014/04/11|         1|
                             UNKNOWN|2011/08/03|2014/07/30|      3992|
```

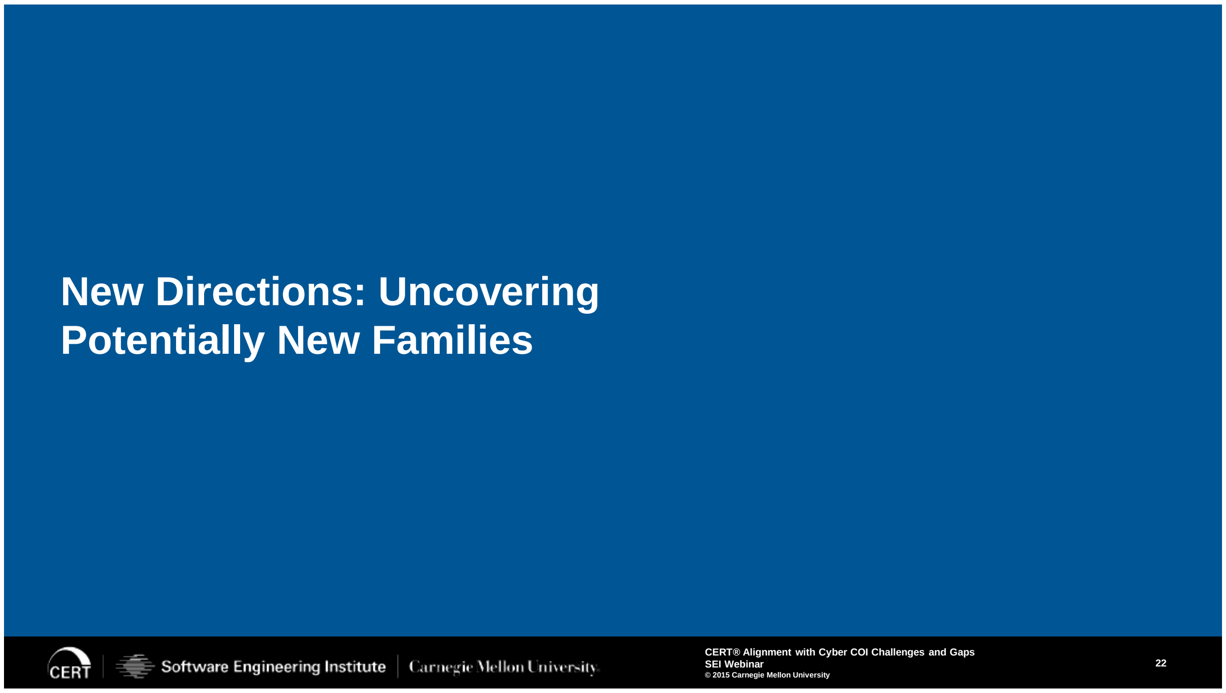
CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

21

**021 So basically the takeaway here is that I'm just looking at one feature right now and I'm looking at the run-time features or the run-time characteristics of this feature and seeing that it, I can pretty well say, use, just as one feature in the dynamic environment to help direct more static analysis. Which I think is a win.

New Directions: Uncovering Potentially New Families



New Directions: Uncovering Potentially New Families

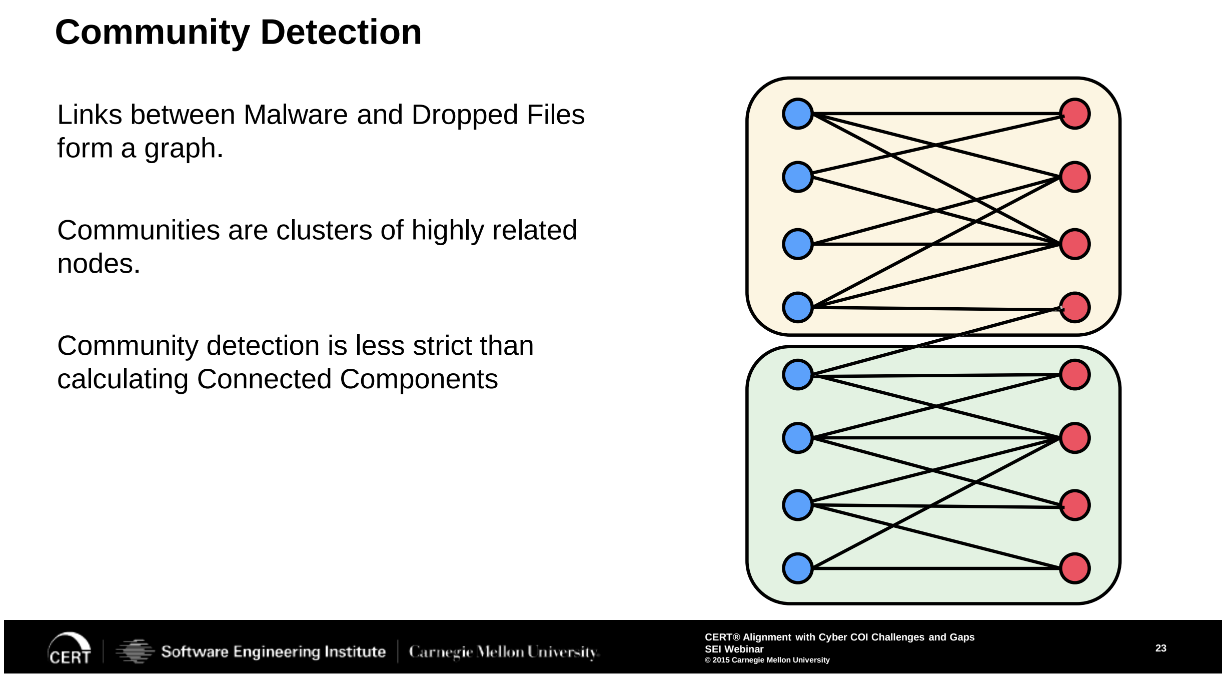
Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

22

**022 So new directions.
Uncovering potentially new families.
I do this with some just basic graph
analysis. One thing--

Community Detection



Community Detection

Links between Malware and Dropped Files form a graph.

Communities are clusters of highly related nodes.

Community detection is less strict than calculating Connected Components

CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

23

**023 So I've been using something called community detection. So as I said before, the links between the malware and the dropped files form a graph. So this is just a silly example here, right. The left might be the MD5s of the malware samples, the right is the SHA256 hashes of the files that they drop. And then the lines, the links, are the relationships between them. And the communities are basically clusters of highly related nodes.

When I talked to the malware team about this, they were very, well, there's something that happens when you try and run graph analysis on these types of environments where you have lots of replication and lots of uncertainty and sort of almost like a power law distribution of who's connected with whom, is that when you try and do strict connected components, basically, like, you get 90 percent of the graph is in one component and it doesn't help you.

So for example, here, right, if I look at this connection, this is just one connected component. But it looks like I might actually be able to tease out two different communities in here. And there's an algorithm that I ran on it, a community detection algorithm, that when I ran it, it actually did take out that one spurious link between the top and the bottom and said, "You have two communities here." So community detection in this case is a graph algorithm. It takes, it's a little bit more computationally costly to run than connected components, but it's less strict than connected components and it can actually start letting you look at more local structure in the graph.

So just as an example of what I did for the files that I had, so the data set that I had.

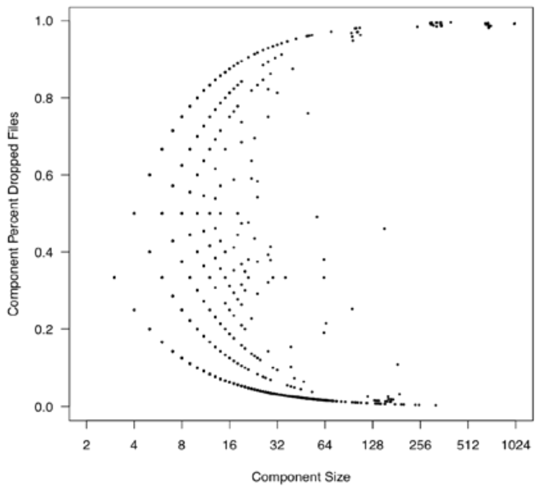
Connected Components

Connected Components

63.6m nodes, 350m edges, 35 minutes Graphchi on MacBook Pro

```
      Count    Size
1       1 63311055
2       1     1009
3       1     1002
4       1      712
5       2      692
6       1      691
7       1      690
8       1      689
9       2      687
10      2      686
11      3      685
12      3      684
13      1      663
14      1      662
15      1      659
16      1      398
17      2      347
18      4      346
19      7      345
20     16      344

 78756 components
```



**024 When I ran connected components I just used--so I had 63 million nodes and that was my 40 million malware samples or however many it was, plus 15 million dropped files. Right. I had 350 million edges and it took me--I just ran it on, with a program, called GraphChi on my MacBook and it took me about 35 minutes. But you'll notice that I'm seeing this one problem, which is like the top connected component has 63.3 million things in it. So it's not all that helpful. And so the graph here is the component size, also on the log scale, and then the percentage of the component that is dropped files versus malware. So the bottom is a lot of malware samples sharing only a few files and the top is a lot of dropped files that are shared by only a few malware samples, and then the middle is sort of in between. That large one is not on the graph, because it would've just shoved everything over to the side. I took that one out.

But when I look at it in terms of community detection, when I run community detection instead of connected components, I get something that's a little bit more interesting, in this case.

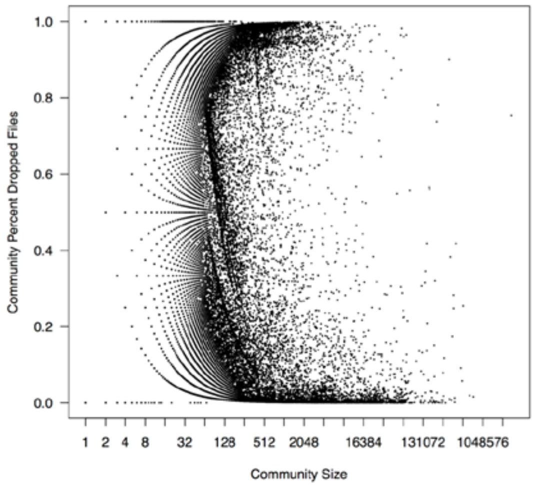
Community Detection

Community Detection

63.6m nodes, 350m edges, 45 minutes Graphchi on MacBook Pro, 2 minutes in Hadoop/Spark

```
     Count    Size
1        1 2853713
2        1  758666
3        1  662108
4        1  529684
5        1  511513
6        1  489196
7        1  409880
8        1  391510
9        1  391009
10       1  377154
11       1  376032
12       1  361557
13       1  350148
14       1  338525
15       1  328812
16       1  324644
17       1  286737
18       1  274181
19       1  267238
20       1  262437

632531 communities
```



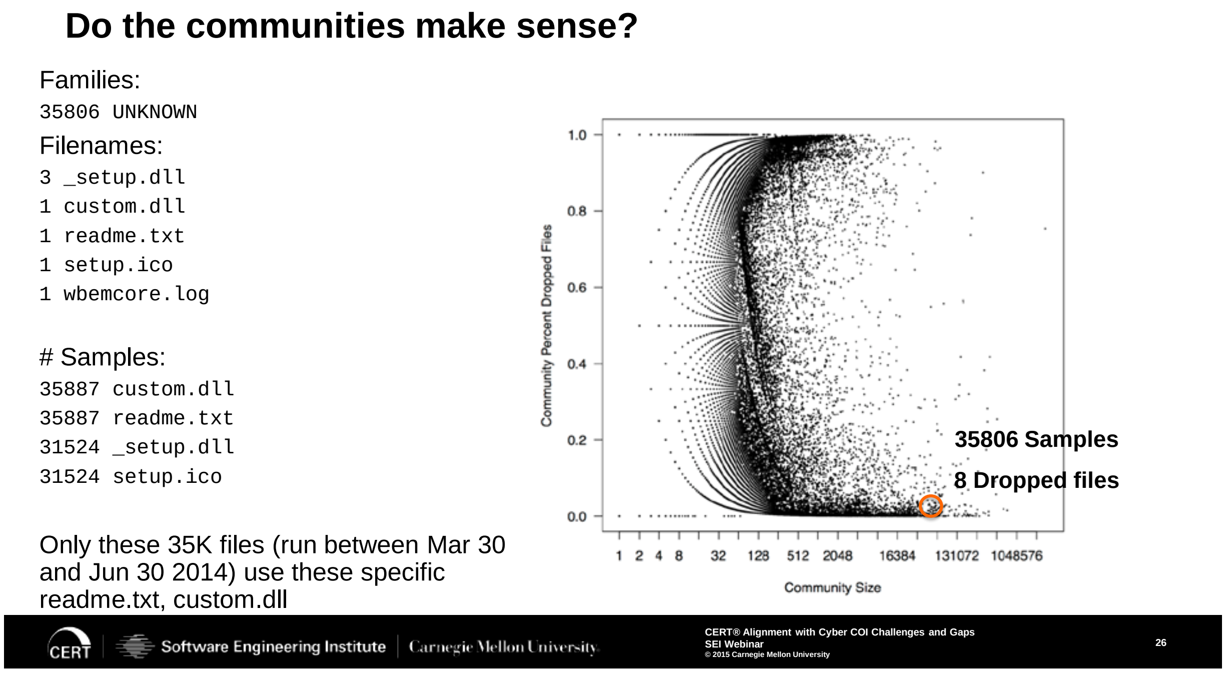
CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

25

**025 Still 63 million edges, still 350 million--sorry. Still 63 million nodes, 350 million edges. It took a little bit longer for me to do it on my Macbook Pro but then I shot it over to our engineering department and they ran it in two minutes on Hadoop, which was kind of cool. But the communities here, as I said, are a little bit more local in this sense. So you get more of them. And I've sort of started to think, like, "Can I see, like, do these communities make sense?" right. So how might these communities start making sense? And I just have been picking ones. I'm curious about the ones where it's just a couple of files, like a couple of dropped files that are shared by lots of malware. So that's the bottom part of the graph there. So I just picked one.

Do the communities make sense?



Do the communities make sense?

Families:

```
35806 UNKNOWN
```

Filenames:

```
3 _setup.dll
1 custom.dll
1 readme.txt
1 setup.ico
1 wbemcore.log
```

# Samples:

```
35887 custom.dll
35887 readme.txt
31524 _setup.dll
31524 setup.ico
```

Only these 35K files (run between Mar 30 and Jun 30 2014) use these specific readme.txt, custom.dll

CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

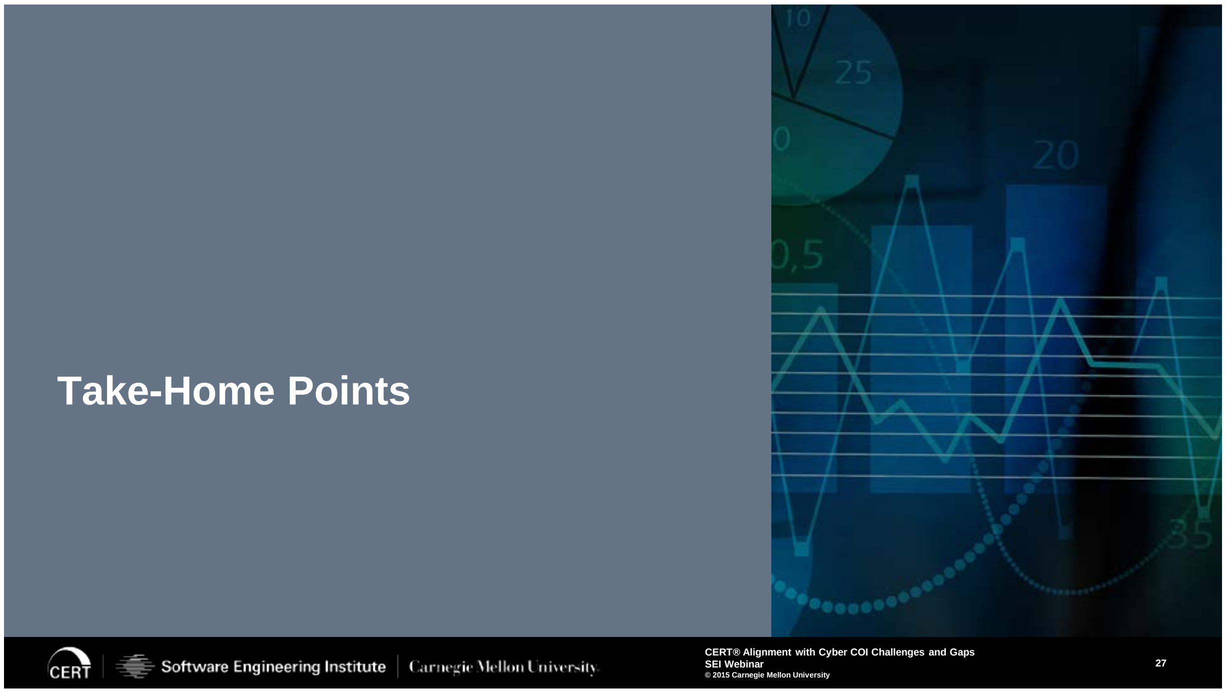
26

**026 And I had basically it was like 35,000 malware samples. And they all shared eight specific dropped files.

And when I looked at the families they were all unknown. So this is complete, this is uncharted territory in terms of what I know about these samples at all. I can look at the filenames and it looks basically-- there's this WebCore.log is kind of something that shows up a lot in different places, but I have setup, I have custom, I have readme. It looks like it's trying, like it's doing some sort of installer. And here's the number of samples that were attached to each of those custom or, oh, each of those files. So I had three files that were named setup, and they might've been the same just up to, like I said, that weirdness about the hash, like, you know, that the hash's small changes will make a different hash.

But what's interesting is that only these 35,000 files use these specific dropped files that were named readme and custom that had these particular hashes. So this is very unique of these two. And they were run across a pretty wide range of time and there was a lot of other things that were run at that time. So that's telling me that these files here, this is probably not a result of the environment but result of some sort of intrinsic thing that these files are sharing.

Take-Home Points



Take-Home Points

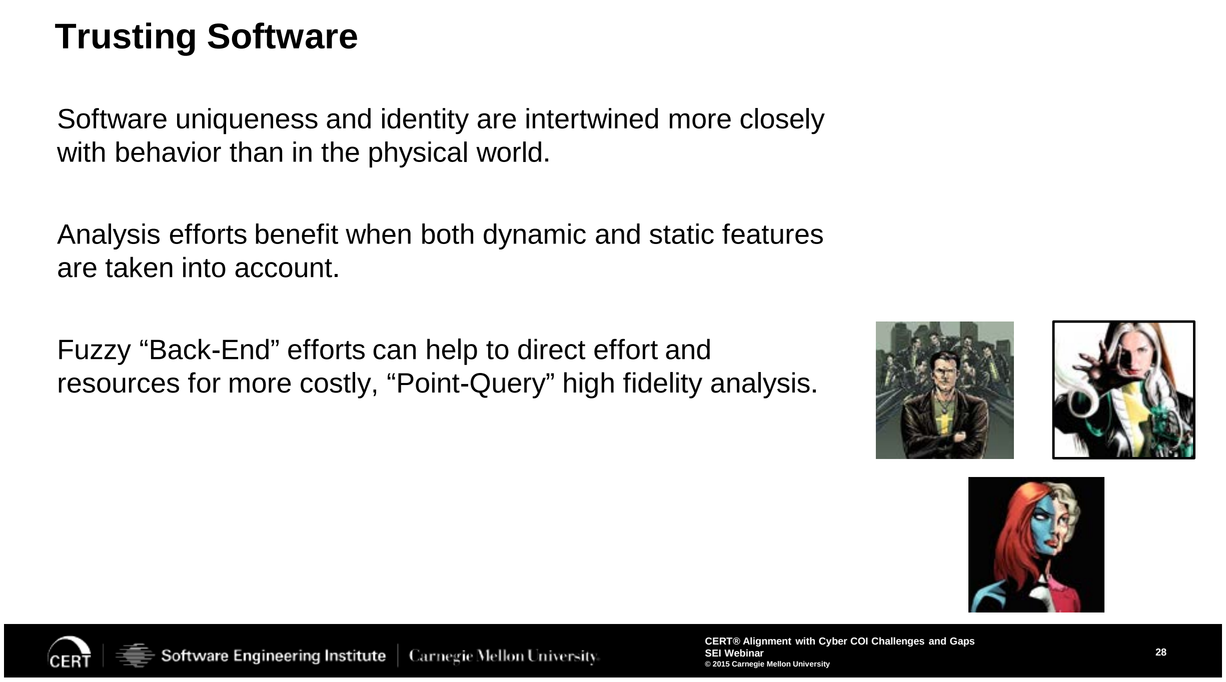
CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

27

**027 I think that's basically what I have in terms of different types of analyses with run-time features. I just wanted to just reinforce a couple of take-home points.

Trusting Software



Trusting Software

Software uniqueness and identity are intertwined more closely with behavior than in the physical world.

Analysis efforts benefit when both dynamic and static features are taken into account.

Fuzzy “Back-End” efforts can help to direct effort and resources for more costly, “Point-Query” high fidelity analysis.

CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

28

**028 Back to the idea of trusting software. There's a couple of things. One is that this idea of uniqueness and identity, that they're intertwined more closely with behavior when you're talking about software than in the physical world, because the features of individuals are engineered and intrinsic. Almost the same way that, you know, and you can be as creative as, almost as creative, as just, you know, artwork in this case, or constructing characters.

When you want to do analysis, analysis efforts benefit when both types of features, dynamic and static, are taken into account. In particular it's harder for authors to obfuscate behavior in a system than it is strings in a binary file. So being able to bridge that gap of learning about both kinds is really important, I think.

And then this idea of this fuzzy back-end efforts that maybe are, you know, they'll produce a lot of things that maybe you can start ranking or maybe you can use those efforts to start to direct where you want to send your resources next for the more costly kinds of analysis.

Q&A

FloCon 2016

Daytona Beach, FL
January 11–14, 2016

CERT | Software Engineering Institute | Carnegie Mellon University

CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University

29

**029 Announcer: Okay. Before we jump into Q and A with Rhiannon, I just wanted to say I could see why the CERT hired someone with a PhD in Statistics now, so... Very well done. Thank you for that.

I want to address just a couple questions that came in the end of the last presentation, was looking for Mr. Behler's slides. We're going to add those post event. So look for those. Anybody that's attending or not attending event, obviously the event is being archived. We'll send out the location for the archive and the link to all the slides tomorrow. But we will add his slides in after the event.

So first question for Rhiannon coming in from Todd asking, "What effect would distance preserving hashes have on this kind of analysis?"

Presenter: So yeah. So there, as I said before, MD5 and SHA256 do not preserve distance. So a very small change would result in two completely separate files. There are examples of hashes that preserve locality. So basically similar files or similar pieces of information would get, would have, close hashes. And in some ways that makes the job easier for detecting, you know, a multiple small variance of a file statically. However, I was thinking about it. I think that it basically invites creativity again in terms of how malware authors would try and circumvent it, for example.

So suppose you had a file and you're trying to get it to look at close as you can to something that's good, for example. Or, you know, and then maybe you have, like, two sections of code and then you flip a coin at the top and you say, "If," you know, "If x equals 1, run this section. If x equals 2, run this section." Statically, right, the only change is going to be x equals 1 or 2 at the top but they're going to do completely different thing. So behaviorally, I think you still need to look at the behavior of these files even if you do have

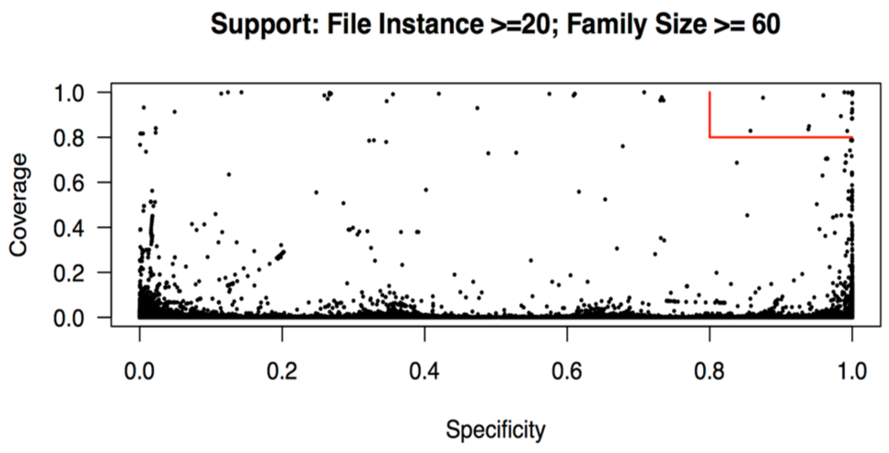
distance preserving. In some ways
it's going to help, but I think in some
ways it's also going to invite, mm,
invite the creativity of the adversary.

Announcer: Okay. So we got one
more question to queue for
Rhiannon. So if you have other
questions, folks, feel free to type
them in now. From Robert asking,
"How do control for common files like
the empty file or files from bootup
procedures that many different yet
unrelated pieces of malware might
access or modify?"

Presenter: So part of that is taken
care of in the environment, I think,
with that profiling. But I also think
that this kind of analysis can help
drive that. So if I wanted to go back to...

Which Files are Specific to a Family?

Which Files are Specific to a Family?



**018 This graph. So what's interesting about this graph is you can actually start using the files that show up across multiple families a lot. So instead of the top right-hand corner, right, so start looking at, you know, files that are specific to families. If you looked at the top left-hand corner you might be able to discover some more types of files and features that you don't care as much about if you want to be able to easily label families, for example. And you could take them out of the analysis if you wanted. I think--yeah. So that's, I mean, there is a lot of it. And so that's what I think, I think both that the types of methods that you're using can help in, help you get better, at making your environment better as well, so... In terms of that. And I should also say, I didn't put it in here but I did have a distribution of the number of malware samples that shared, or that shared each dropped file. And I had a histogram of it, and it's kind of like lexical analysis where you don't care about words like "the" and "and" and stuff like that, but neither do you care about words like "antidisestablishmentarianism" or, you know--

Announcer: Right.

Presenter: --those types of words that only show up once. You also don't care about the words that show up, you know, all over the place. There's sort of that sweet spot that you have to get at. And they actually, I did discover, when I did this, there were--I did find the empty file and I got rid of it. And I also found about 38 other files that were shared by, like, 40 million, all 40 million of these samples, like, a lot of them. So I sort of set, I just set a threshold and said--and they, it was sort of like a bump too, where it was like these 38 and then I had a big jump down to the ones that were shared a little less frequently. So yeah. Just looking at them in terms of that I think is helpful.

Announcer: Okay. From Allen wanting to know, "Was there any attempts to normalize resultant samples?" For example, pack, unpack, a foo file, and then compare the different samples.

Presenter: That's something that I would probably be handing over to the malware guys.

Announcer: Okay.

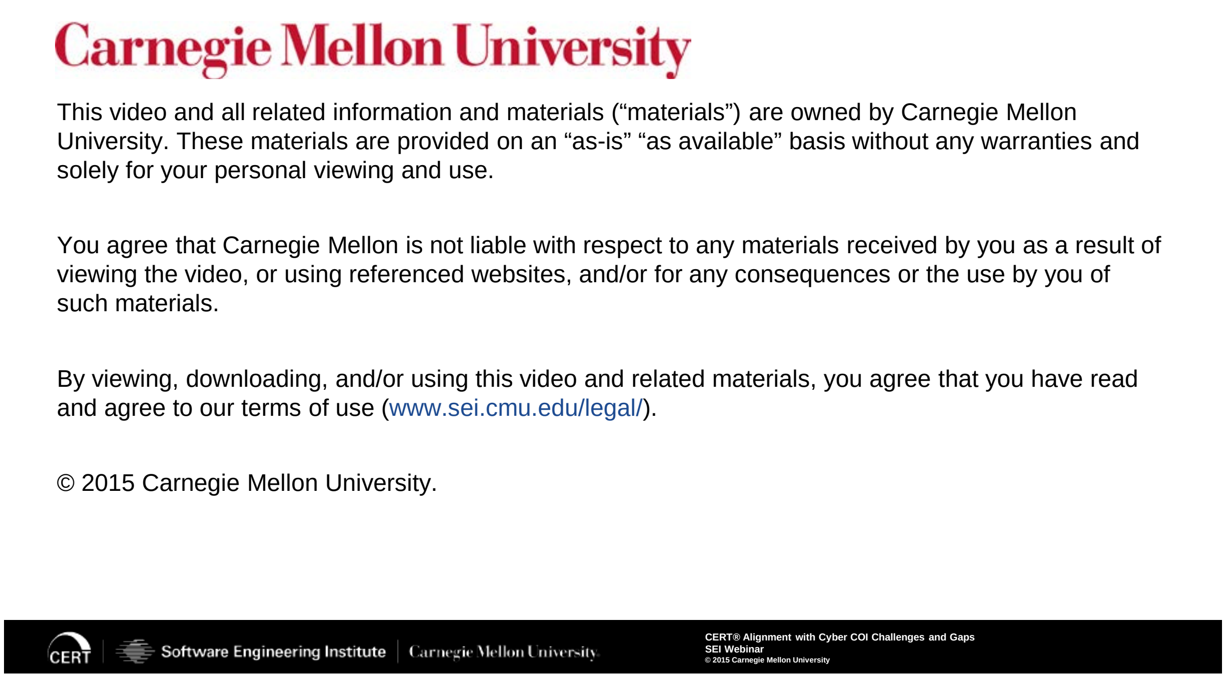
Presenter: Right. Yeah. So they were certainly interested in the results of this. And also it sort of had us thinking about how we tagged the families, because they had a particular lexicon that involved, you know, colons and pluses and such like that. And I think what I did was I stripped all that out and had tags for each family and I called them base families. And so I think probably the better way to, or a way to go in the future, would be to put each of those tags as separate so that you could filter on all of them more easily. Does that...

Announcer: Okay. That's the last question we have for Rhiannon. We're just about to wrap up here. Thank you very much for your presentation. Appreciate your time presenting today. Great job.

Presenter: Thanks for having me.

Announcer: So we're going to have about a 15 minute break so we can set up for our resilience panel and our panelists will include Matthew Butkovic of CERT, John Haller of CERT, Katie Stewart of CERT and Sean McCloskey of the Department of Homeland Security. So we'll be back promptly at 1:40 to start the resilience panel. Thank you.

Carnegie Mellon University



Copyright 2015 Carnegie Mellon University