
Token all the Things!
{azhar|nick|marco|jason}@thinkst.com
Thinkst - 2016
https://www.flickr.com/photos/nikoskoutoulasphotography/8672040494

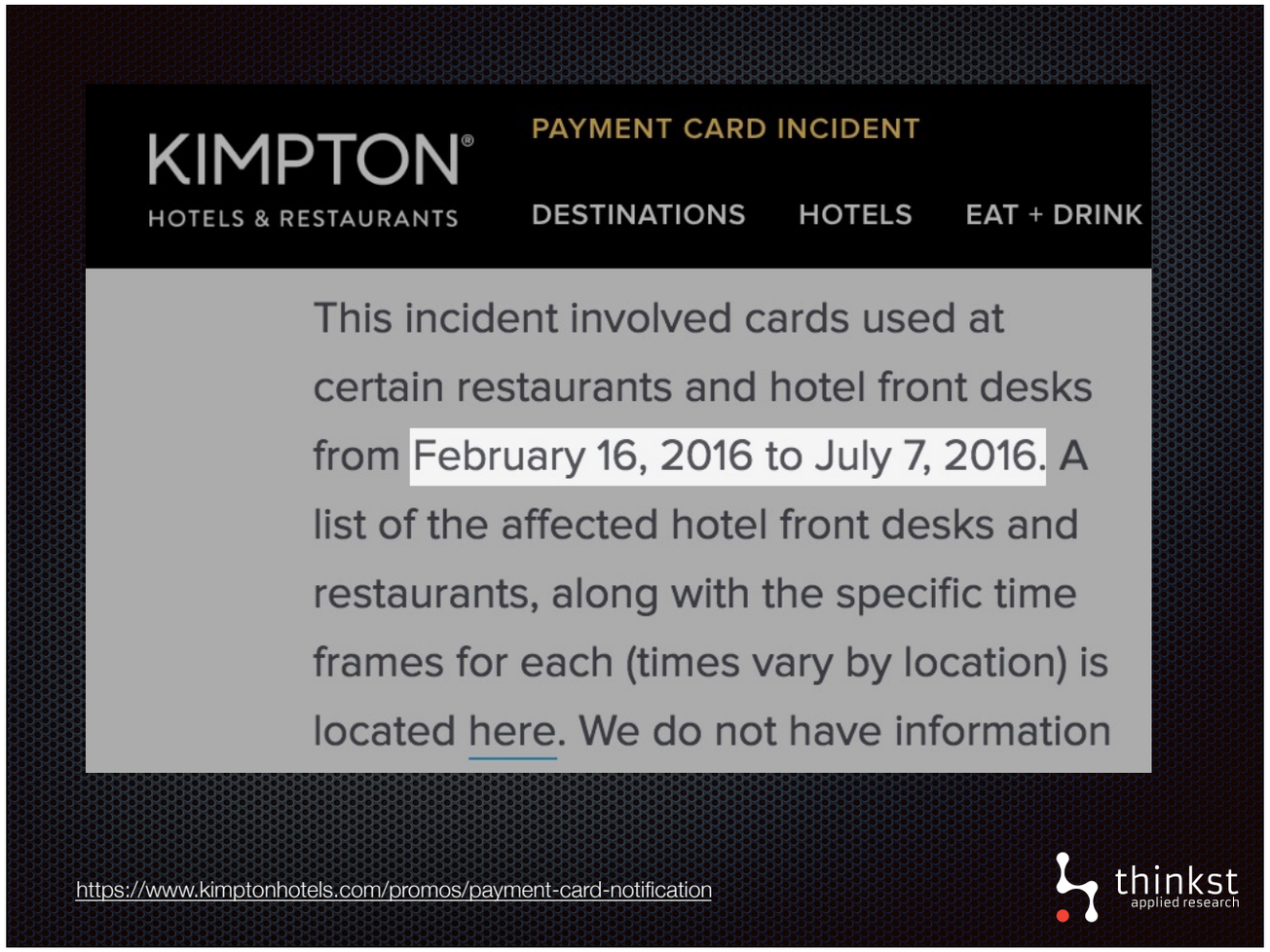

Herr Olsen (2012) https://www.flickr.com/photos/herrolsen/7727320856

- We are Azhar and Nick
- We work at Thinkst Applied Research
- We work on Canary, our honeypot solution as well as Canary Tokens (which is what this talk is about)

# The Plan

We have three aims in talking here today:

- We'd like to introduce tokens,
- Talk about the infrastructure we've built to revive tokens to take it in a few new directions.
- Show how we can deploy or apply tokens in a number different settings to detect threats.




- Two weeks ago, a hotel chain publicly announced that earlier this year, their PoS systems were breached and credit card details were stolen from people who had stayed at their hotels or went to their restaurants.
- There was period of 5 months or 142 days in which card data was being stolen.

## 26 Kimpton Hotels Probes Card Breach Claims

JUL 16

**Kimpton Hotels**, a boutique hotel brand that includes 62 properties across the United States, said today it is investigating reports of a credit card breach at multiple locations.

http://krebsonsecurity.com/2016/07/kimpton-hotels-probes-card-breach-claims/
http://krebsonsecurity.com/2016/09/kimpton-hotels-acknowledges-data-breach/

thinkst applied research

- Brian Krebs suspected a breach a month before the announcement.
- The hotel chain learnt not long before him.
- This is quite typical.

# time to breach detection
# median = 146 days

https://www2.fireeye.com/rs/848-DID-242/images/Mtrends2016.pdf

thinkst
applied research

- Their breach time is almost exactly the median time to detect breaches according to recent a Mendiant report.
- The report surveyed breaches they responded to last year across different industries.
- Half of those breaches examined for the report, still took longer than 146 days to detect.
- Those are sizable time periods in which attackers get to play, but it’s also a lengthy time in which they can tip off defenders.


- Our focus here is in detecting threats on internal networks during that time period.
- This is in contrast to early honeypot research (and even ones today) that attempt to track, learn and study the behavior of other attackers. Our colleagues, Haroon and Marco went into more detail about that in a talk they gave last year.
- Here, we'd like to know someone broken in and been places they shouldn't be, like exploring a critical database or amongst documents on an important file share.

- In this setting the defender has particular advantage


- Once attackers land insider your internal network, they’re at disadvantage. They don’t know the lay of the land and they need to explore it, while remaining hidden.
- The defender, on the other hand, has good idea of how things are played out and where the valuable things are.
- To make use of this advantage, the defender can make the territory in inhospitable to attackers. Places where attackers are likely to poke around and where your regular users don’t go, are good places to lay traps or tokens that trip up attackers alerting you to the presence.

# Old tokens


- *traps* — City of “Agloe”
- *tripwires* — Spafford & Kim (1994)
- *honeytokens* — Spitzner (2003)

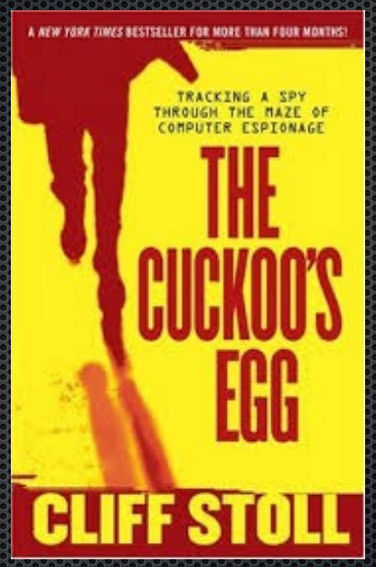
https://www.acsac.org/2003/papers/spitzner.pdf

http://www.symantec.com/connect/articles/honeytokens-other-honeypot

http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2114&context=cstech

thinkst
applied research

- This idea of laying traps or honey tokens to trip up adversaries is an old one. Related ideas crop up different names.
- Map makers would sometimes use trap streets or towns. If a sneaky map maker copies the a map instead of making their own, the trap would give the game away.
- In computing, people have explored setting tripwires on files and on spreading honey tokens (or just tokens) to try detect people snooping around.
- Earlier tokens tended to be a bit more passive.




- A really old example of a honeytoken is from 1987. It was documented by Clifford Stoll in his book the Cuckoo's egg. He had discovered that a hacker that had broken into a computer lab where he was the system administrator. From watching the hacker's activity, Cliff Stoll could see that the hacker was interested in military affairs. The hacker would also go through emails looking for useful information. (Later it was discovered that the stolen data was been sold to the soviet union.)
- As a trap to find out who the hacker was, Stoll manufactured fake emails to appeal to the hacker's interest in military affairs.

# Old tokens: postal address

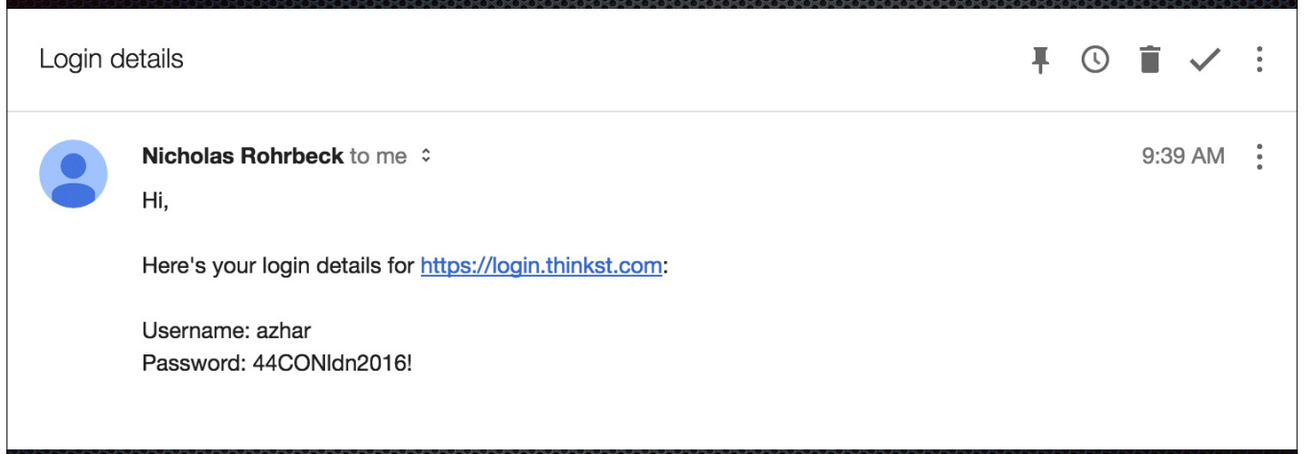
SDI Network Project
Lawrence Berkeley Lab
Mail Stop 50-351
1 Cyclotron Road
Berkeley, CA 94720

"Please send your request to the above address"

thinkst
applied research

- One of the fake emails encouraged the reader to send their address to of the secretaries, so that she could send sensitive military documents back to them. (The hope here was that the address they sent in would reveal something.)
- Quite amazingly - it actually worked!
- The hacker got the email passed the postal address onto others, who posted a letter (via intermediaries) to the address.

# Old tokens: mailboxes




thinkst
applied research

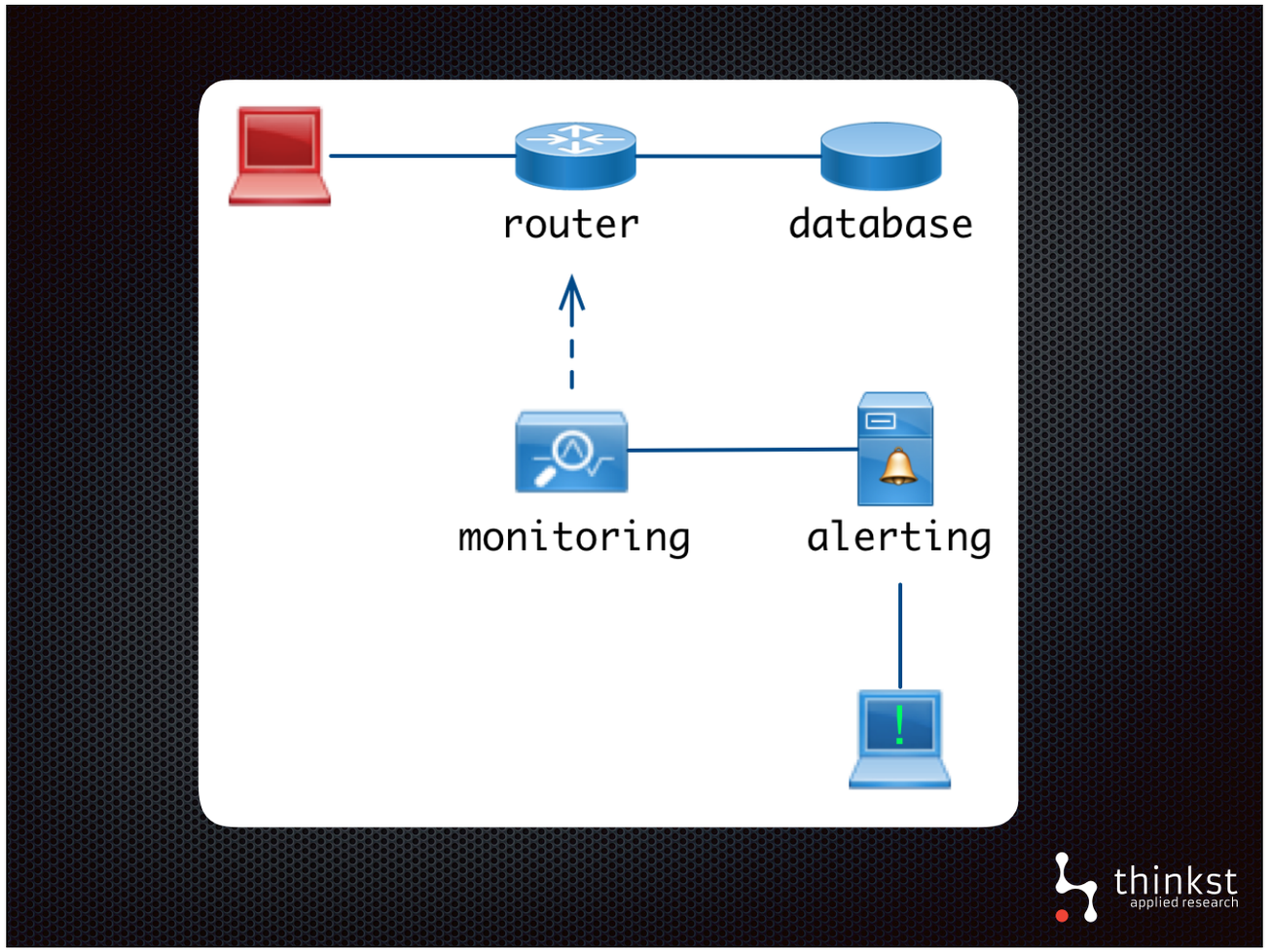
- An modern version that tokened emails would work similarly. Instead of a postal address, there would be a link and credentials.
- If someone is snooping through these emails and sees the credentials, they'd be very tempted to go and try out the login with the credentials to see what else they can access.
- If the defender sees that anyone tried to use these credentials, it's a good cause to go investigate. However, this still requires work to monitor for that login, either on legitimate site or create a fake site, just for this alert.

# Old tokens: databases

| Name | Surname | Credit Card Number |
|---|---|---|
| Roald | Dahl | 5219 8645 6473 9535 |
| Rowan | Atkinson | 4716 9577 4195 0479 |
| Totally | Fake | 5555 4444 3333 2222 |
| Stephen | Fry | 4485 4731 8418 6638 |
| … | … | … |

thinkst
applied research

- For databases, early tokens could embed fake data in the database.
- The important thing is that the data looks valuable so that the attacker is tempted to take the data out. Credit card data is one example. That would appeal to attackers like those in the hotel chain incident mentioned earlier. The difficulty with using this token is that it's quite passive, so that defender needs to actively monitor for it's use.




- Early database tokens required monitoring a networking chokepoint for the token data.
- The attacker, on the left, accesses the database via some chokepoint, and when the data comes back out, the monitoring the defender has setup picks it up and sends and alert.
- An alternative, which isn’t much better, but suited to credit cards, would be to keep an eye out on shady forums too see if data crops up.
- Now, we just learnt last week, that it turns out not all old tokens were so bad.

# Moonlight Maze
1998

https://medium.com/@chris_doman/the-first-sophistiated-cyber-attacks-how-operation-moonlight-maze-made-history-2adb12cc43f7

thinkst applied research

- Now it turns out, not all old token was that clumsy. Last week we learn that recently (2mons) some documents were published on operation Moonlight Maze. This was investigation by US agencies into widespread intrusions on university and military networks. The incident was well-known at the time, but details were kept secret. A researcher sent in a FOI(A) request and more details about it were revealed.
- The documents revealed that investigators used a honey tokened document to try determine where the attackers were coming from. It was surprising to us that already in 1998, the document used DNS request to alert the investigators. We'll touch on this technique again in the talk.


- With our relook at tokens we aim at attacker detection without honeypots, networking monitoring at a chokepoint, and endpoint agents. (Honeypots will get a brief mention at the end, but it isn't necessary for talking about tokens.)
- The aim is to use the defender's homeground advantage to alert on significant events, by placing tokens that actively are in the way of an attacker snooping around.


- In order to achieve this aim, we built Canarytokens which is just our take on tokens, with new ideas and a full implementation.
- Our goal has been to make it easy for anyone to quickly generate new tokens that can be scattered around a network and left, only to alert you if triggered.
- Importantly, we are creating tokens that look valuable.  Their point is to trip attackers up and alert us of their presence
- We are bringing the old, passive tokens into 2016


- We are going to emphasise this point throughout the talk.
- We've built canary tokens to make it easy for you to use them.

# Canarytokens are simple

- Unique tags that can be embedded in

  Documents, emails, databases, file watchers, executables, process watchers, LinkedIn, Bitcoin, Imgur (and we've barely scratched the surface.)

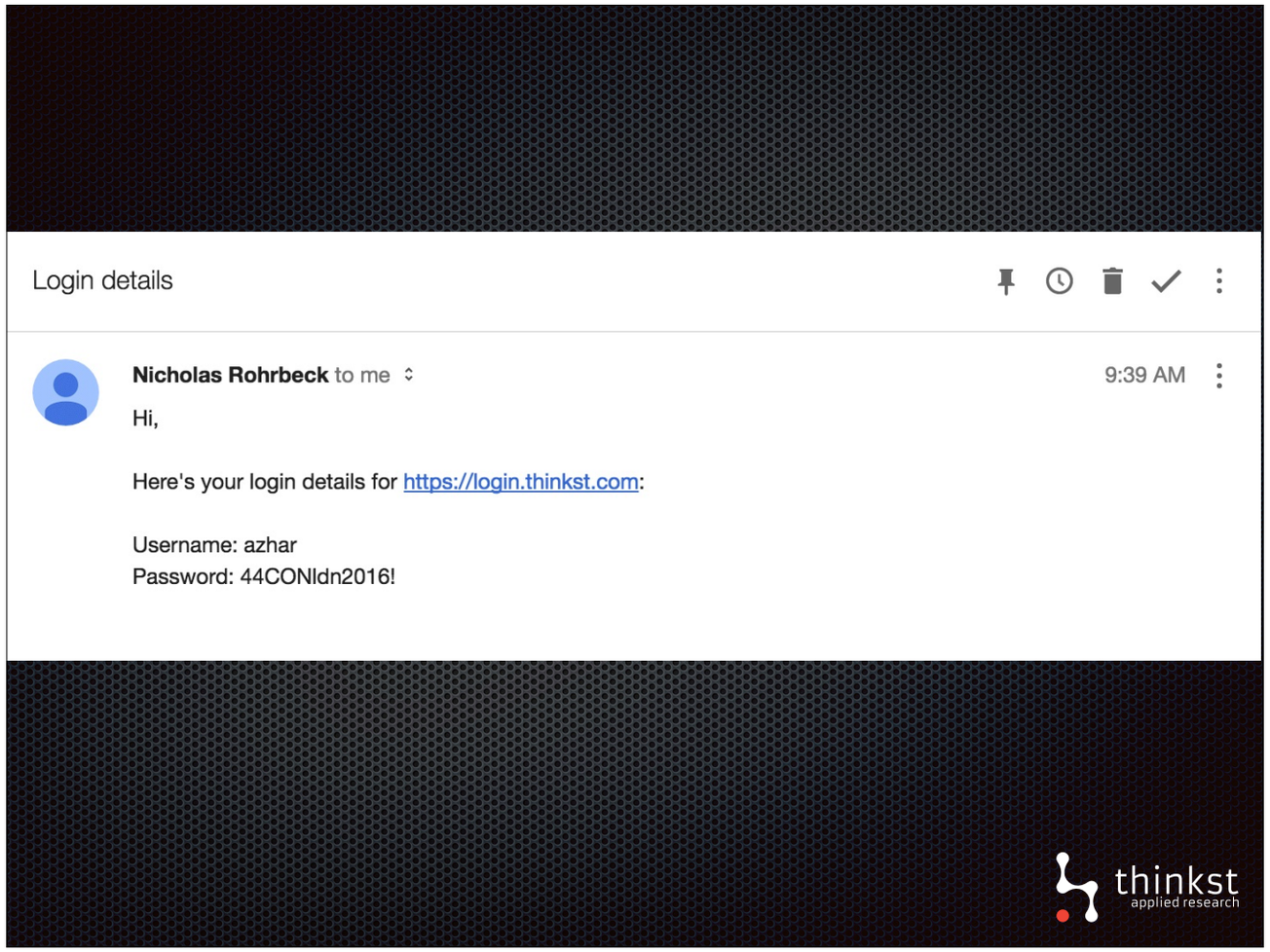
- When that tag is triggered, you get an alert.

thinkst
applied research

- So, Canarytokens are a basic idea.  You embed a unique identifier into the thing you want to token.  When the token is triggered, you get your alert, telling you that someone is doing something they shouldn't.
- They beauty is that these basic ideas are composable, meaning we can combine them in different ways, to alert you in different settings.

# HTTP Tokens


- Lets take a look at the simple channels used by canary tokens.
- Perhaps the simplest token is a URL.  When it is hit, we get an alert.
- It is a simple action. But powerful as we'll see.

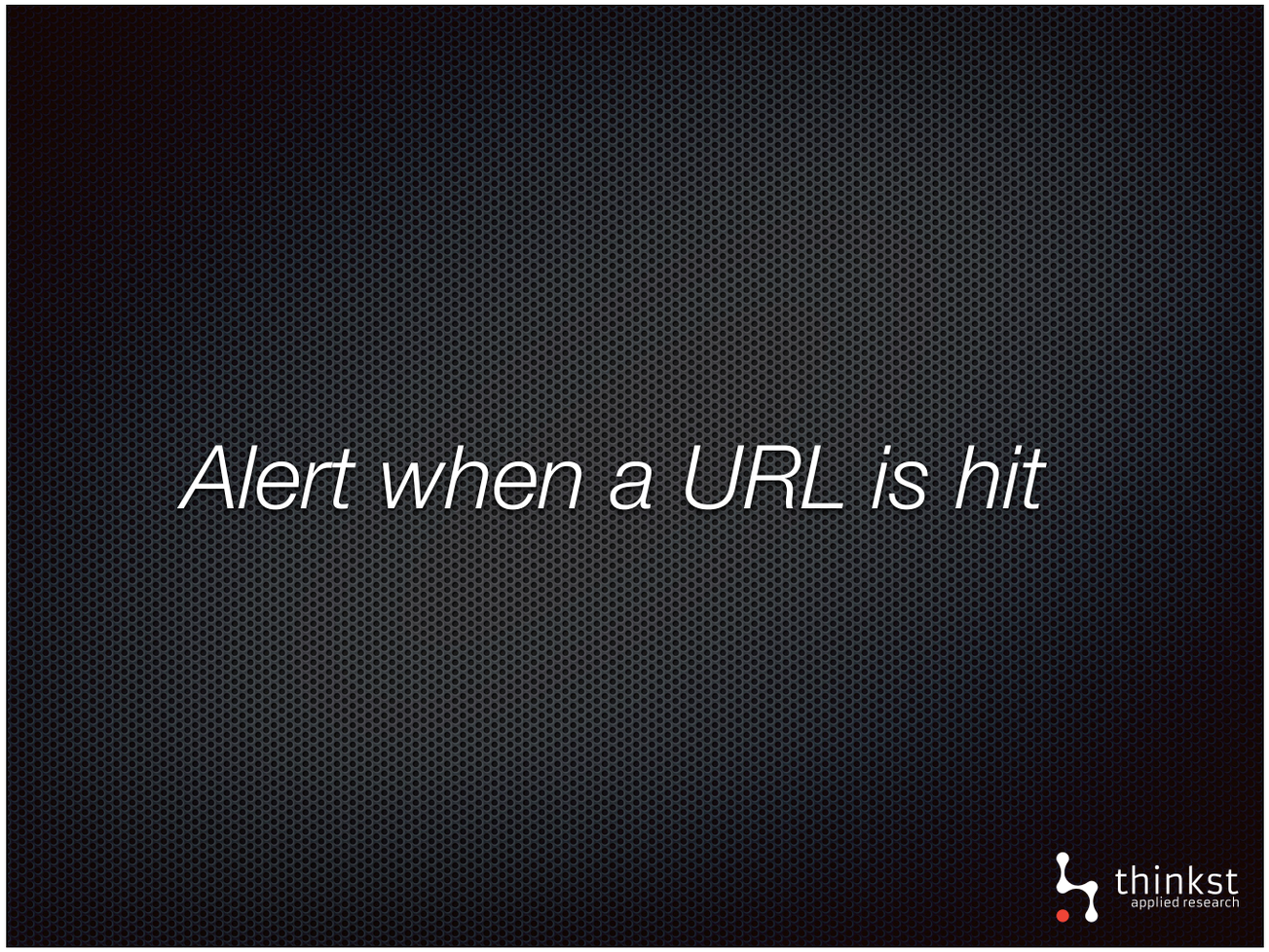



- Coming back to our example Az mentioned earlier of leaving a URL in our mailbox, we can now create a tokened url and leave it in the mail.
- If it ever gets hit, we know someone is snooping around our mail.

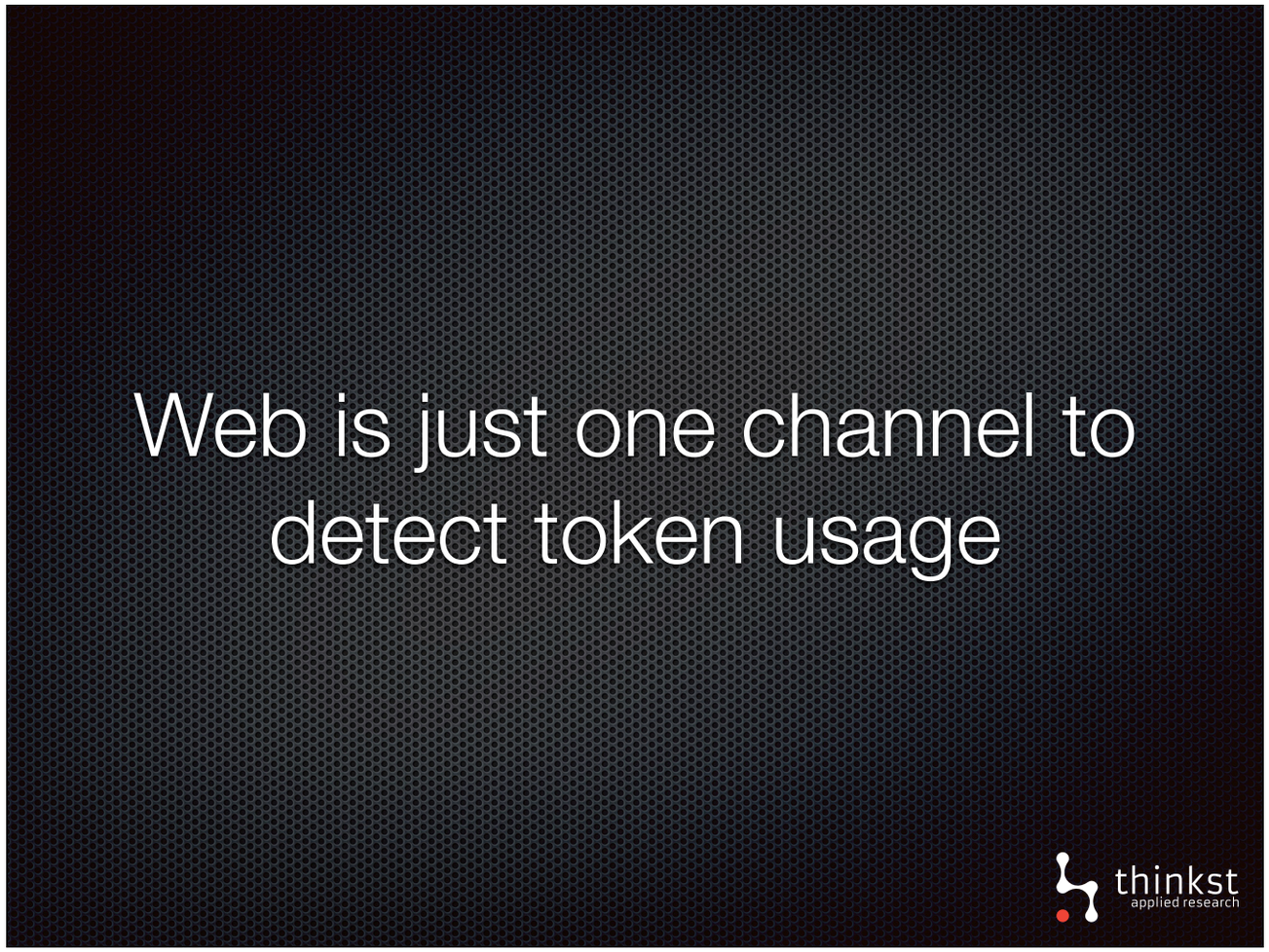
# Demo: Basic Canarytoken

thinkst
applied research

- Quick demonstration of generating and triggering Web bug




- So what we just saw was a demonstration of our HTTP Token, which is a building block for other tokens.
- It is important to understand this what exactly happened here as it is used to create more sophisticated tokens.

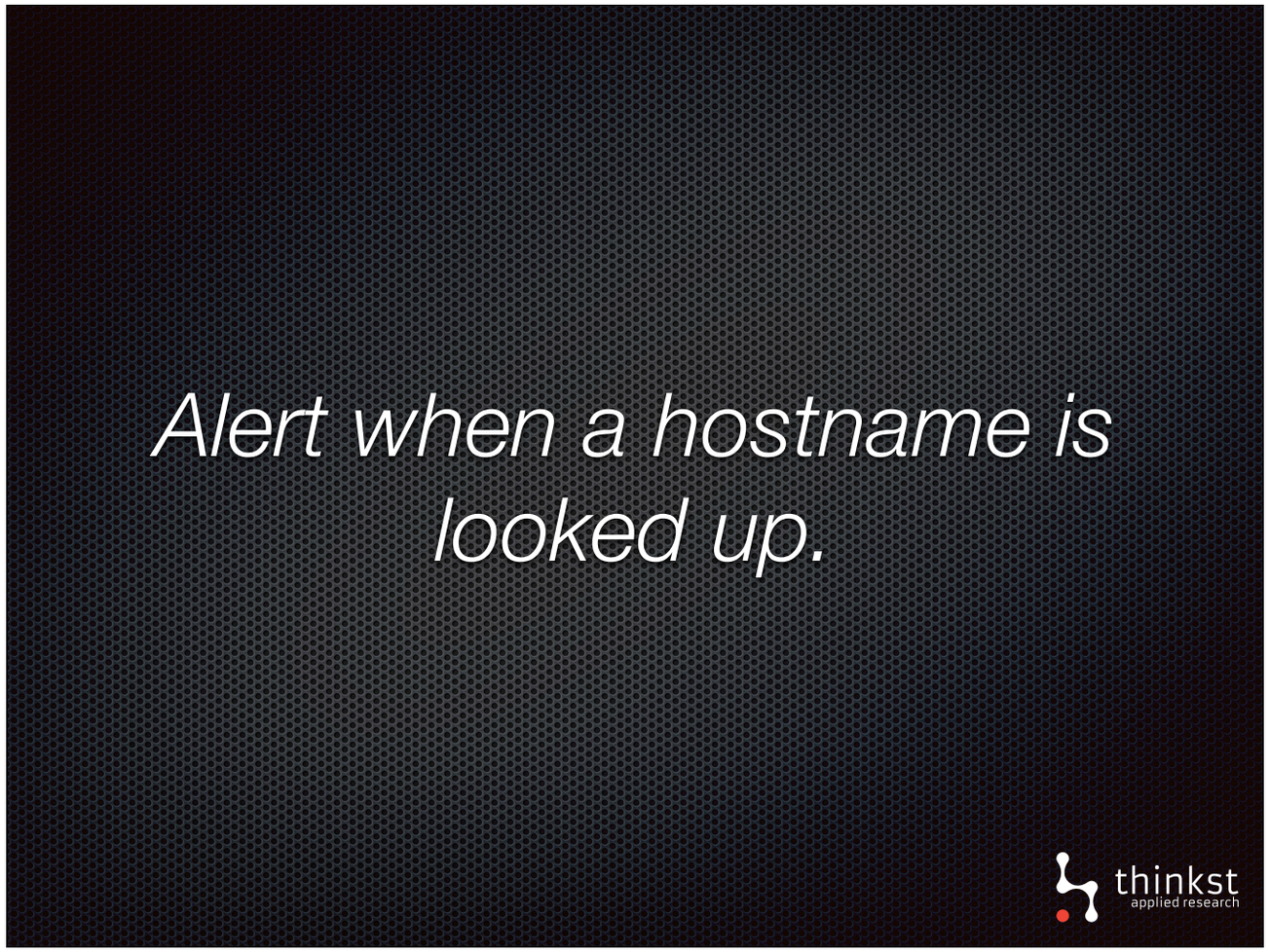



Let’s look at other channels supported by Canarytokens


- Using DNS we can Alert when a hostname is looked up.
- Here the token is the hostname, and when it is resolved, it triggers an alert.
- Let's see how it works.

# Demo: DNS channel


- Quick demonstration of generating and triggering DNS Token




- This is another simple building block which will be used to create more sophisticated tokens,.
- So we now have HTTP and DNS channels to trigger.
- DNS is useful since it doesn’t need a direct connection, and there are a lot of places where you get DNS query out, where http would fail.
- It is important to note that using DNS does mean you lose the source IP address whereas HTTP gives you a source IP, but you will lose out if on a restricted network.

# SMTP Channel


- The next channel looks at smtp.
- Simply put, we are modernising the old idea of the fake postal address mentioned earlier.
- The key here is that the token is the email address and the trigger is when an email gets sent to it.

# Demo: SMTP channel


- Quick demonstration of generating SMTP token
- This could be used in a database of users - you leave it there and if it is ever hit, you know your database is compromised


- So what we saw there was tokening using smtp.
- We create a fake email address and leave it lying around somewhere that shouldn't be found
- If someone sends an email to it, an alert is fired and we know there is compromise

# Niche Channels


- We've now looked at three basic channels that form the core of canary tokens.
- These three channels are used to build new, more powerful tokens
- However, we've also built other, more specific token channels that have direct uses.

# Tokens can also be tied to recurring queries, which are potential flags

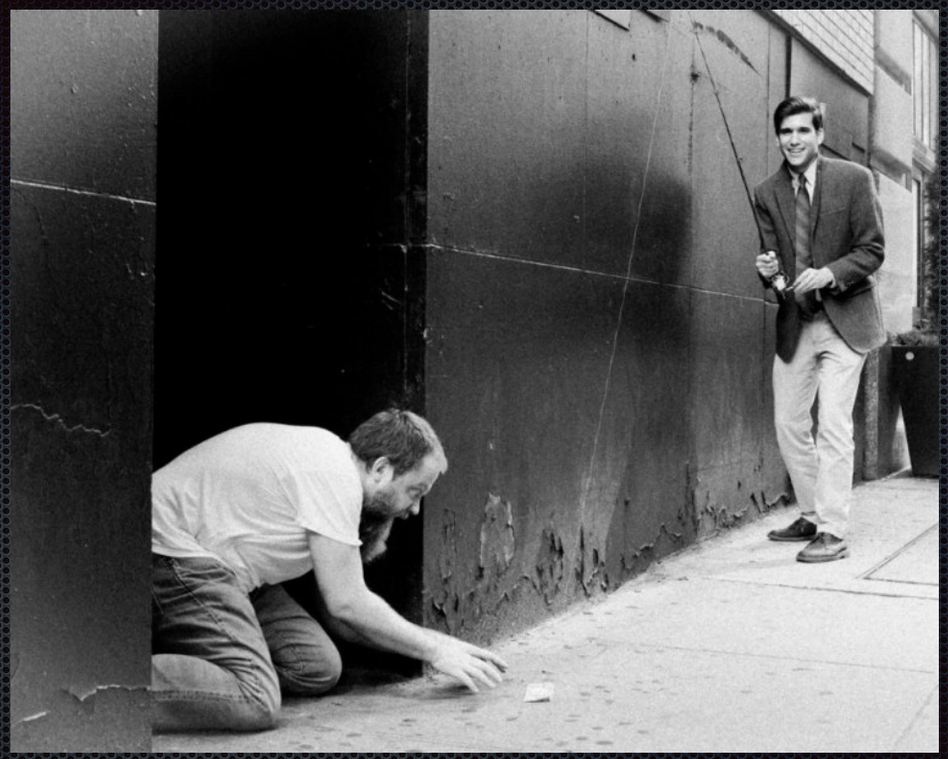
Examples:

- Poll Imgur pic for view count changes.
- Poll LinkedIn for profile view changes.
- Poll Bitcoin address for change in balance.

thinkst
applied research

- We can setup the canary token server to poll services and watch for changes - alerting if there is an unexpected difference.
- It's worth remembering what we want here: a simple way to get a heads up that someone got owned

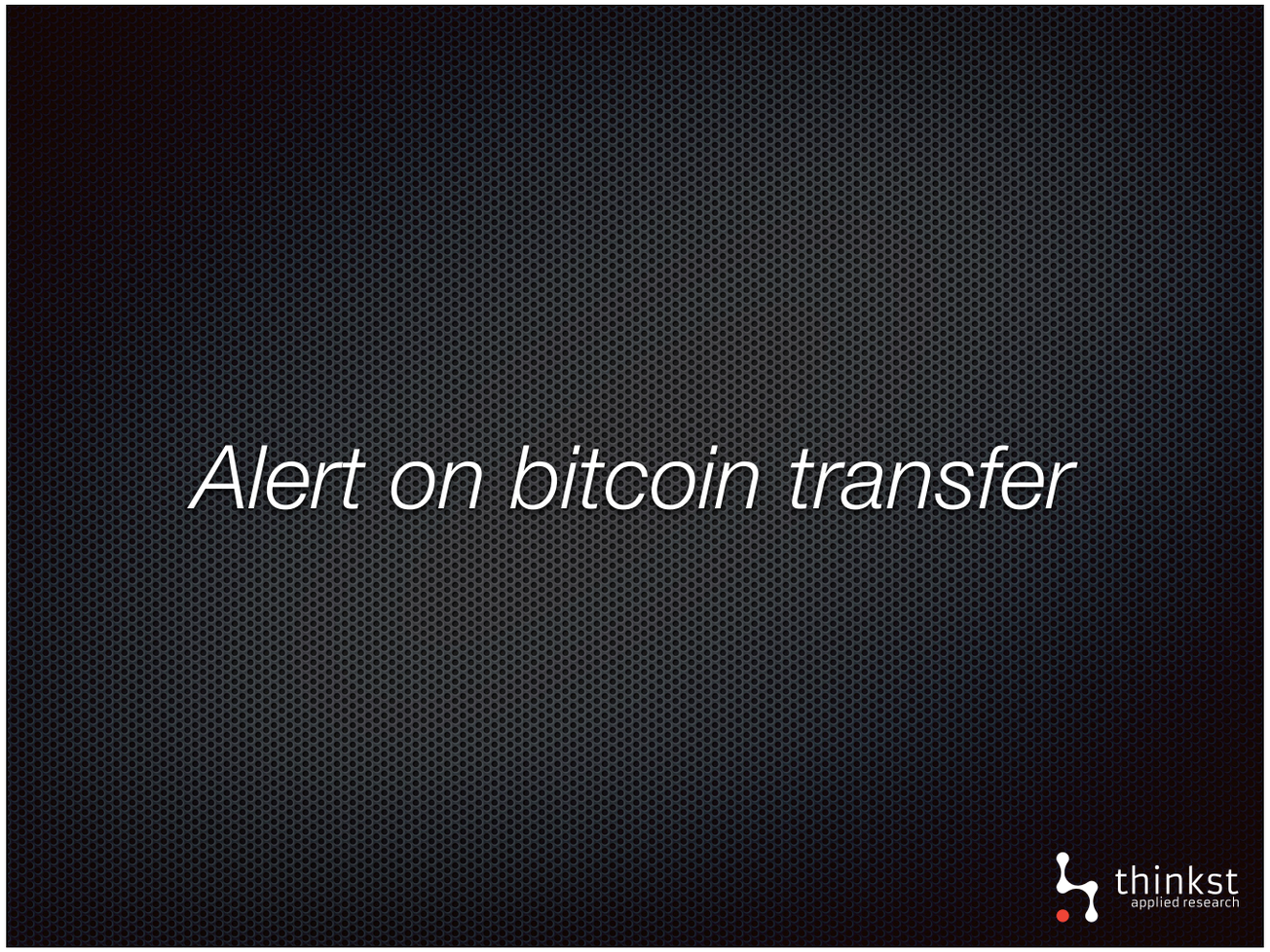

- To demonstrate one of the polling tokens, we can look at our bitcoin token.
- Let's set the scenario... you have a server and want to know if it is ever owned
- You know that when an attacker breaks in they are going to be looking for anything useful and anything they can monetise.
- So, you leave a bitcoin wallet (or many) lying around on the server
- If it is ever emptied, you know the server has been taken.
- Tokens lets you monitor the wallet and alert you on the change.



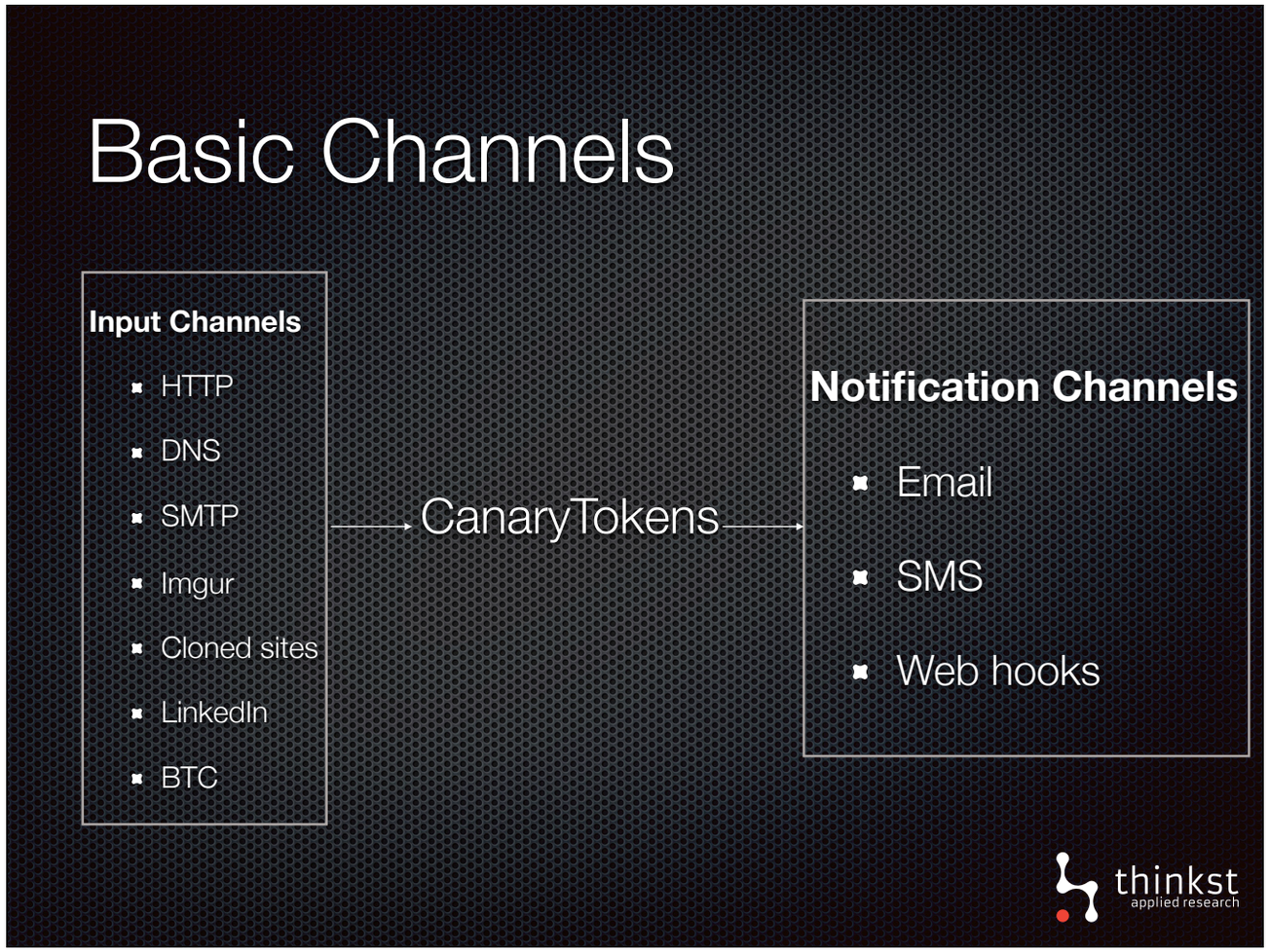
http://www.vanityfair.com/news/photos/2012/10/mitt-romney-high-school-pranks#9

thinkst
applied research

- This isn’t blockchain tech,  we are just playing on the old trick of leaving money around to see if it’s get’s stolen
- The way it works is:
  - You give us a bitcoin address, we save the value
  - We then we poll the address for the balance (at blockchain.info)
  - If the balance drops, we alert you

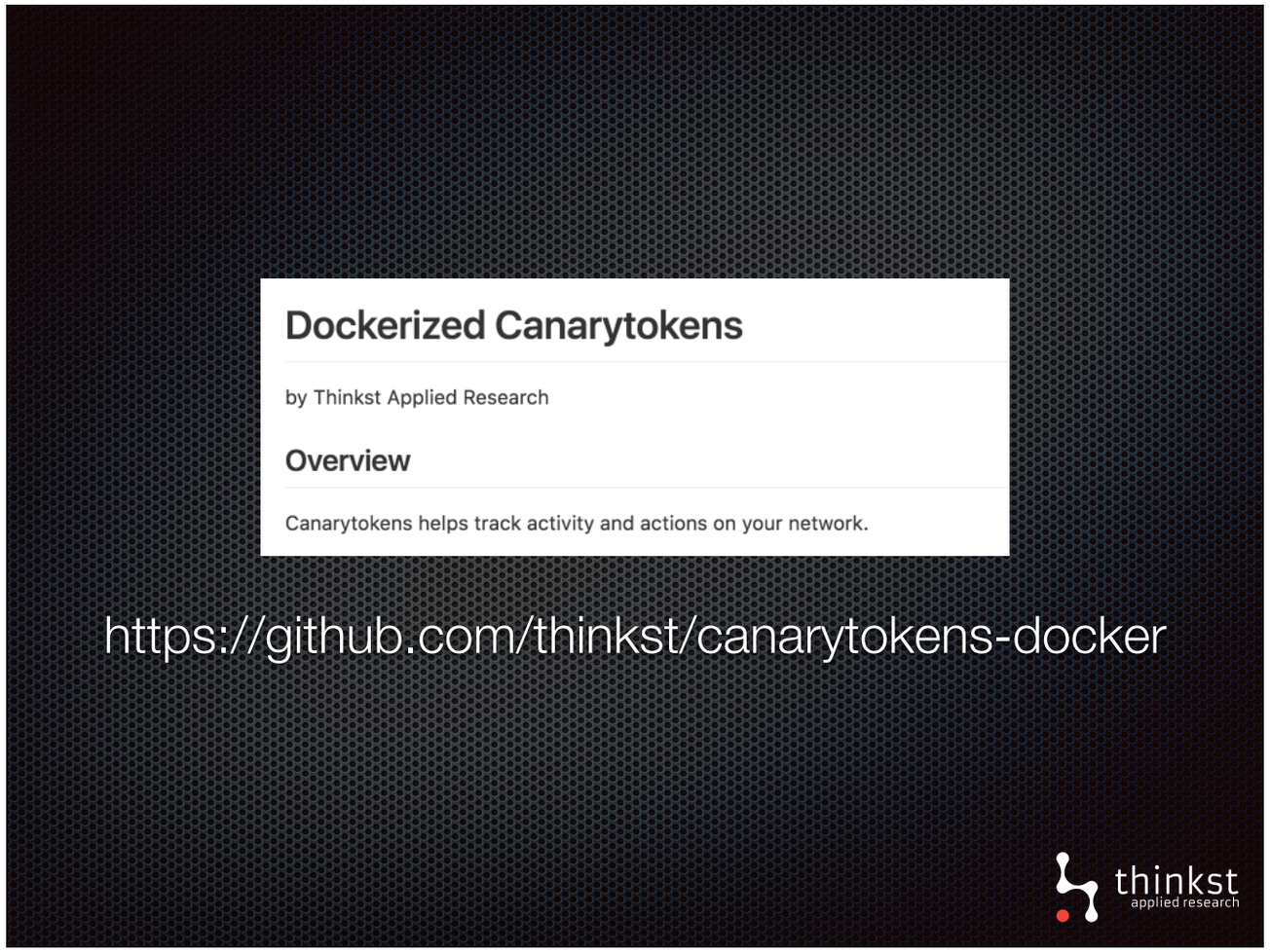



- We've now demonstrated how you can use tokens to poll a service and alert on changes.
- Of course you could be saying that you can easily do this yourself with a bash script and you'd be right.
- What we say is, use tokens because you just need to supply an address and we do all that for you, no need to create the script or maintain it.
- It's all about ease and simplicity

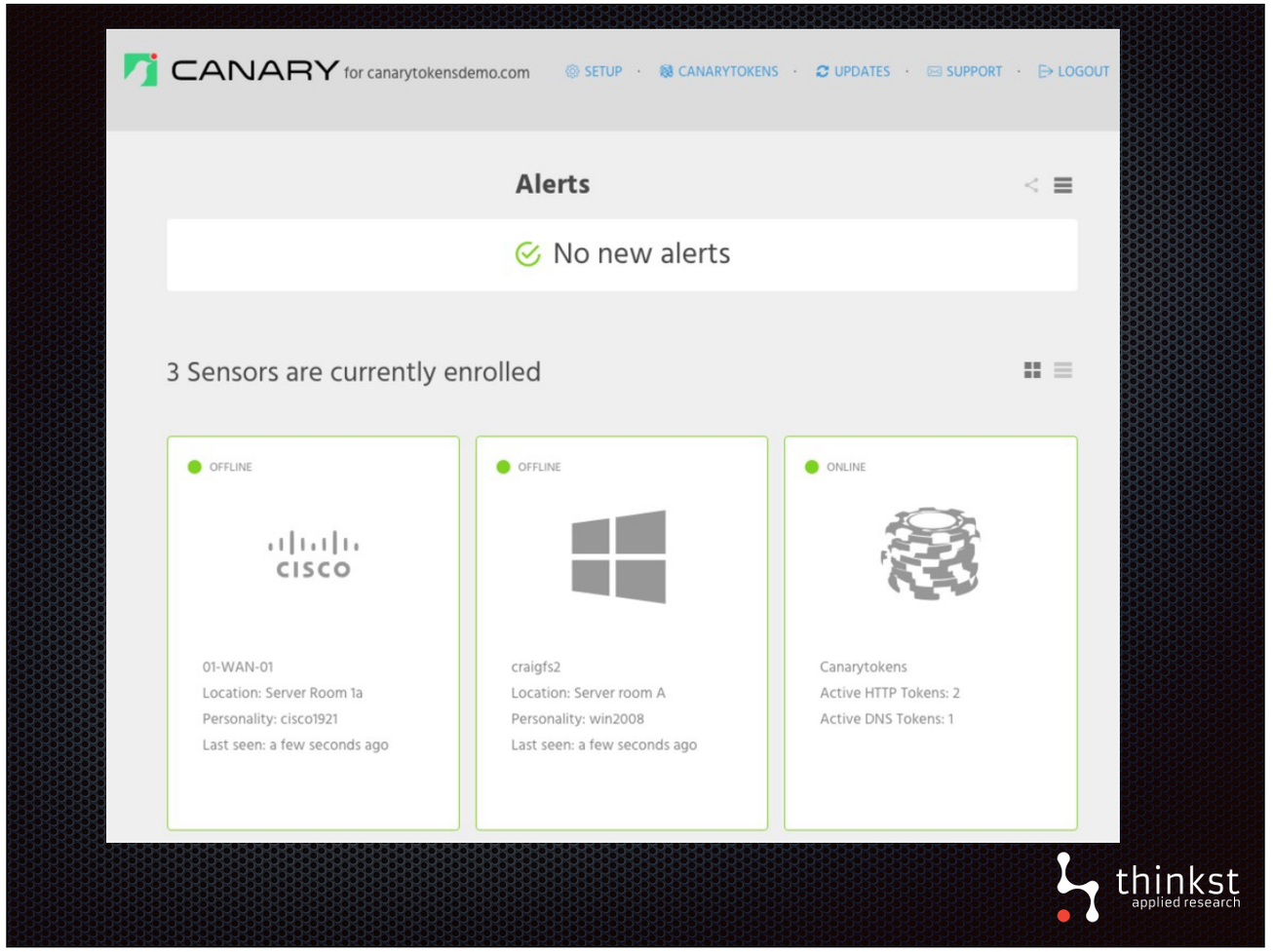



- We've now looked at fundamental channels provided by canary tokens
- When these channels are hit you get notified over email / sms or our recently added web hooks.
- You may have noted that our core channels all rely on DNS and wondered if an attacker could just block traffic to our publicly hosted canarytokens.org You'd be right, but lets look at a few points.


- Firstly, we have many people using our hosted tokens server everyday.
- We have thousands of tokens currently created on our server.




- Secondly, our canary tokens code is open source.  You can easily download it yourself and setup your own server.
- To make it even easier, we've provided docker images which you download, quickly configure and have your own server running using your own domain.




- Finally, if use our commercial honeypot, we are bringing canary tokens to your console, meaning you'll have your own tokens server tied directly to your console.


- This brings us back to the point we are try to make.. canary tokens are simple and easy to use.
- There is no maintenance or monitoring required.
- You simply generate your tokens and scatter them around your network, and leave them.
- If someone trips over them, you know.
- We are removing any reason for you not to use them.


- We’ve seen how we can create simple tokens and generate alerts.
- Where this gets interesting is that we can build on top of these basic channels and deploy tokens in different settings to alert on more significant events.

# How could tokens help detect attackers on a network?

thinkst
applied research

- We're using tokens  to trip attacker snooping around during breach that hasn't been noticed yet.
- While this is intended to increase visibility, it is in no way a replacement for well-managed logging which is useful in it's own right.
- However, one strength of tokens comes for alerting in places where it can be difficult to do more traditional logging.

# Actions worth alerting on

- Exploring a database in production (viewing trap tables)
- Browsing trap files and directories
- Process execution
- Viewing sensitive admin config pages
- Viewing physical devices in physically restricted spaces
- Browsing trap cloud storage
- Opening trap apps on a phone home screen

thinkst
applied research

- Our approach here is consider significant actions we'd like to notice that would immediately warrant some investigation. For example, if someone is exploring a database in production, it'd be worth investigating. For action we want to alert on, we find ways to embed token in the way that the action will trigger it.

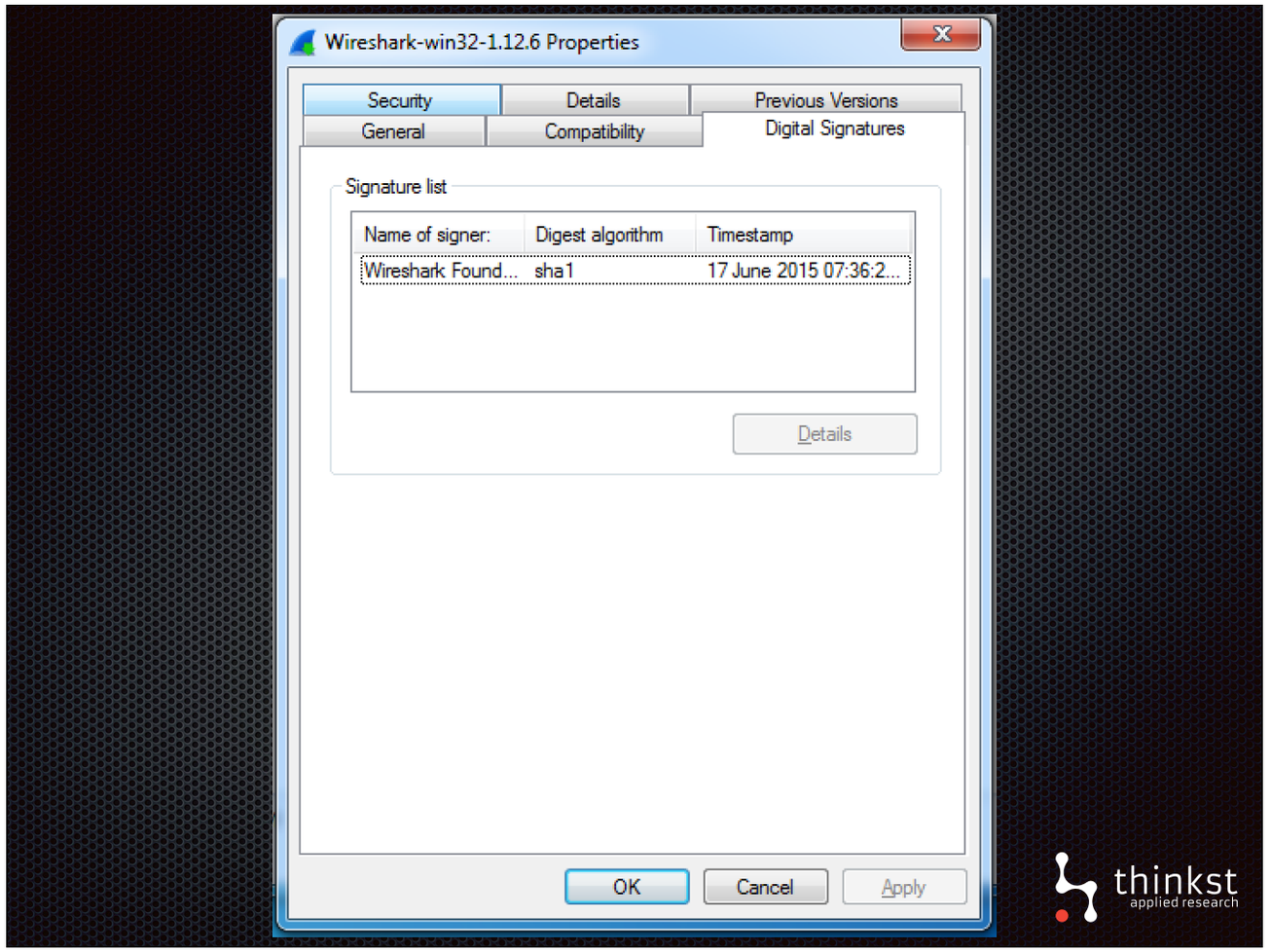
- Certainly agent-based solutions could handle some of these as well, but with Canarytokens we're exploring lightweight alternatives that are less intensive to deploy.

# Process execution (Windows)

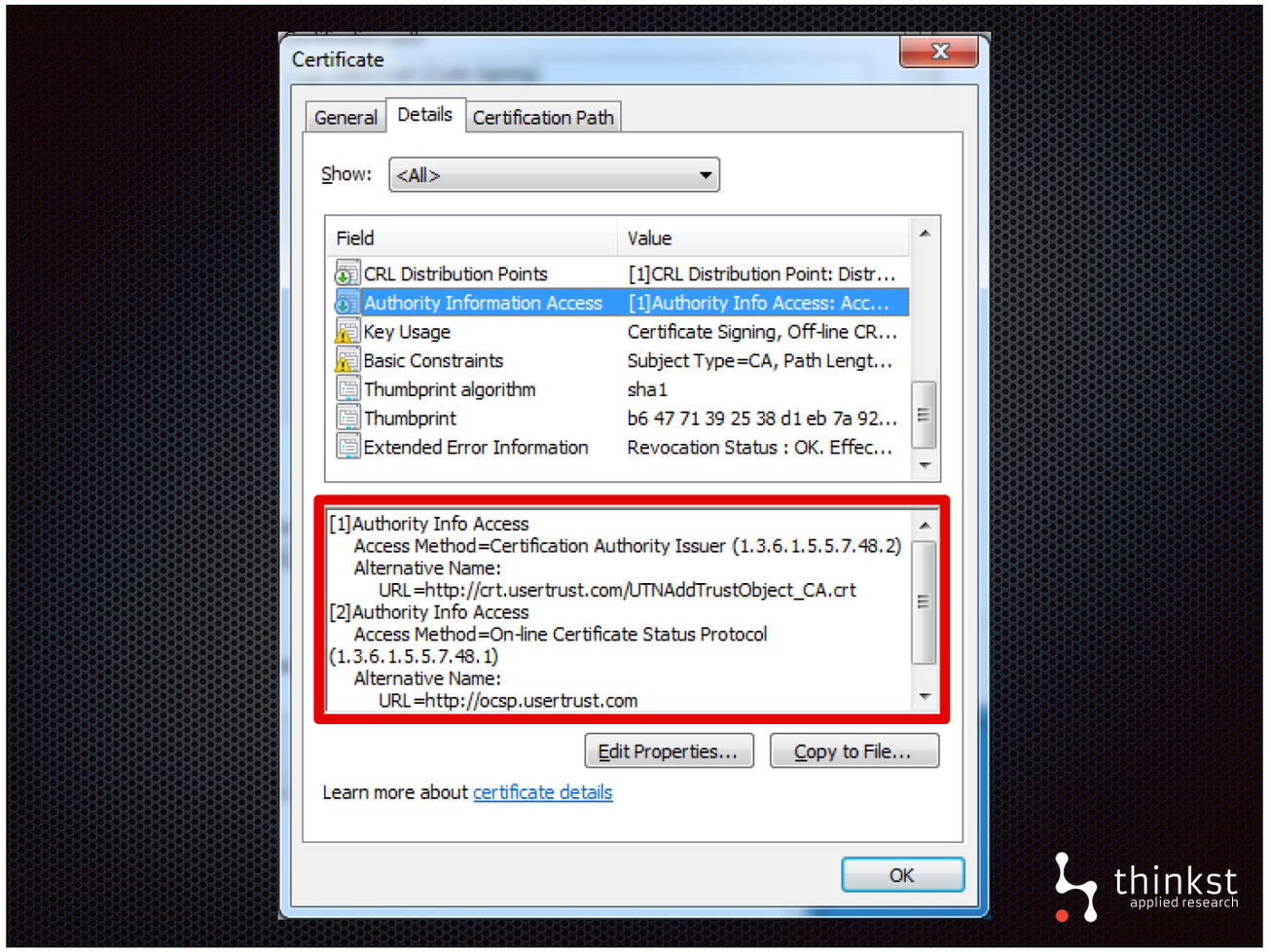
http://www.slideshare.net/zanelackey/attackdriven-defense

- To start us off with, there are times when process execution is a significant action worth flagging. When attacker first gains access to a Windows machine, it wouldn't be surprising if she executed ifconfig or net.exe.  Typically, it's instinctive step to orientate herself on the network of whatever's she's compromised. The clever idea to watch this from from Zane Lackey's talk on attack driven-defense which is well-worth watching.

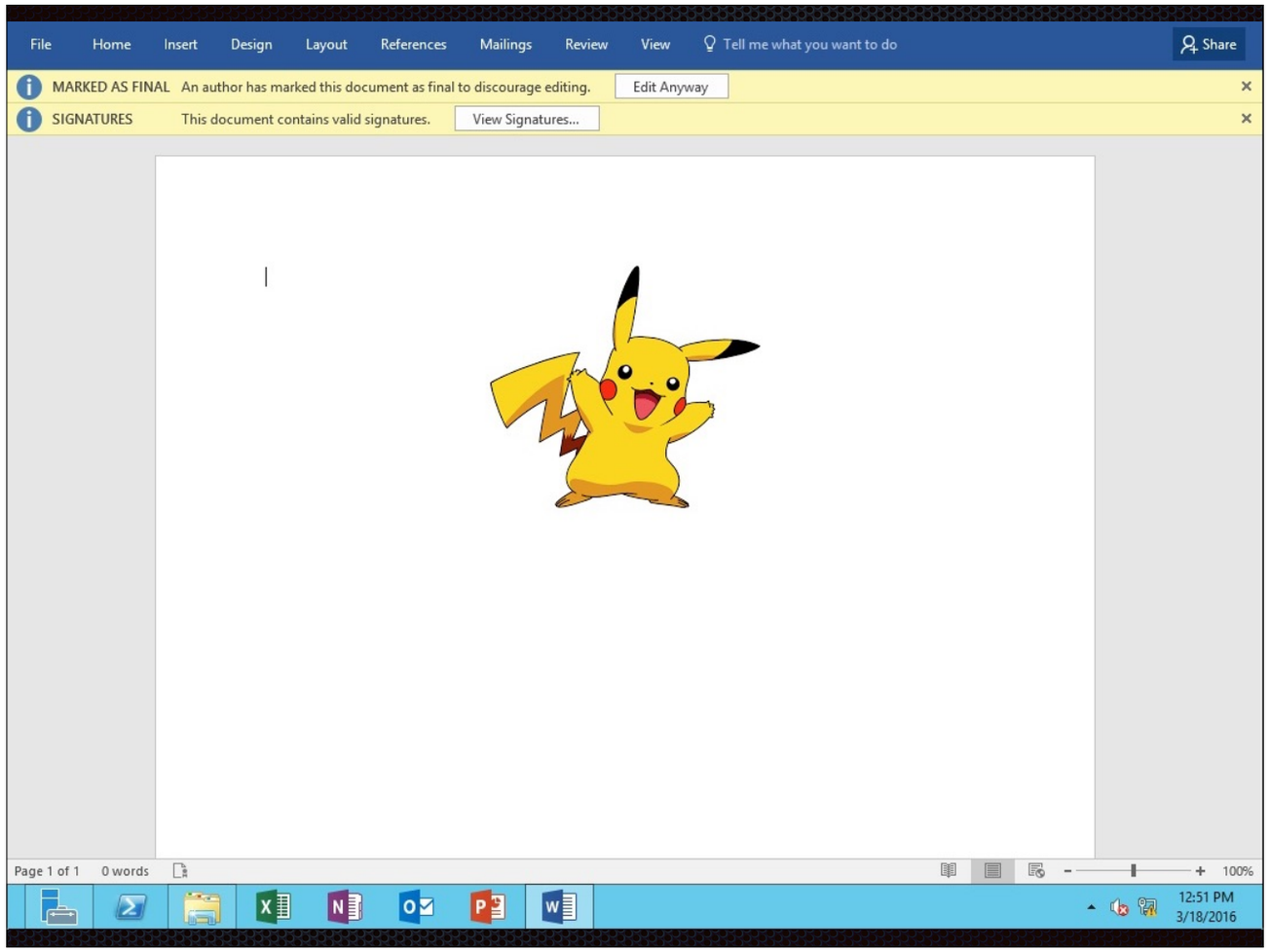
- On a stable and quiet server running these could signal potential misuse. This could tracked with fully fledged logging, but there's a more lightweight solution.




- Windows executables can be signed, and have that signature verified when it's run. We can rely on the signature verification to alert us when the executable is run.




- When the verification happens, a URL in the certificate is hit to fetch more information about CA. By replacing this with a token url, we'll get an alert when attacker runs our binary.




- As an aside, this also works for signed office documents, but only when not opened in protected view.

# Verifying Signed Files

- EXEs and DLLS verification needs software restriction policy enabled with certificate checking
- No need to be a recognised CA
- Builds on work by Alexey Tyurin and Didier Stevens

thinkst
applied research

- To get this to work, we need to enable certificate checking for executables. We don’t need any trusted CA for this to work, our fake one will do.

# Demo: Signing EXE


- Quick demonstration of signing IPConfig and triggering an alert when it is run.


- By tokening the signature verification process, we can get an alert when an attacker has landed on one of our quiet servers and simply runs a routine command.

# Detecting file and directory browsing

- Early tripwires monitored when trap files have been read, or trap directories browsed.
- This is useful for detecting when someone is going through files looking for valuable data to take.
- It’s a good signal, as there are few good reasons to browse fake directory trees.
- It’s even more suspect, if the fake directory tree is on file share that few have access to.

# Detecting File Reads: Linux

- canaryfy daemon powered by inotify
- canaryfy fires a canary token to alert

thinkst
applied research

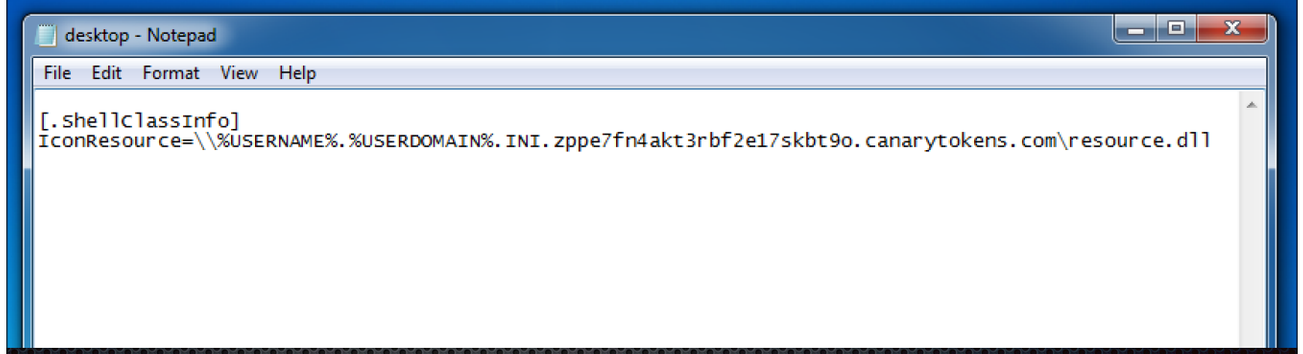
- On Linux, we've written a small daemon, canary, that does the trick.
- It uses inotify to get alerted on file reads, and can trigger a token to report back. ( This could also be done with auditctl — which may work well with your logging if you're prepared to handle beastly log format)

# Detecting File Reads: OS X

- Use Dtrace a system tracing framework
- Daemon-less probe triggered on file read
- Probe triggers token


- On OS X we can do away with a daemon.
- Dtrace, the amazing system tracing framework runs on OS X.
- It’s quick to set up a dtrace probe that triggers on a file read which directly triggers the token.

# Directory Browsing: Windows



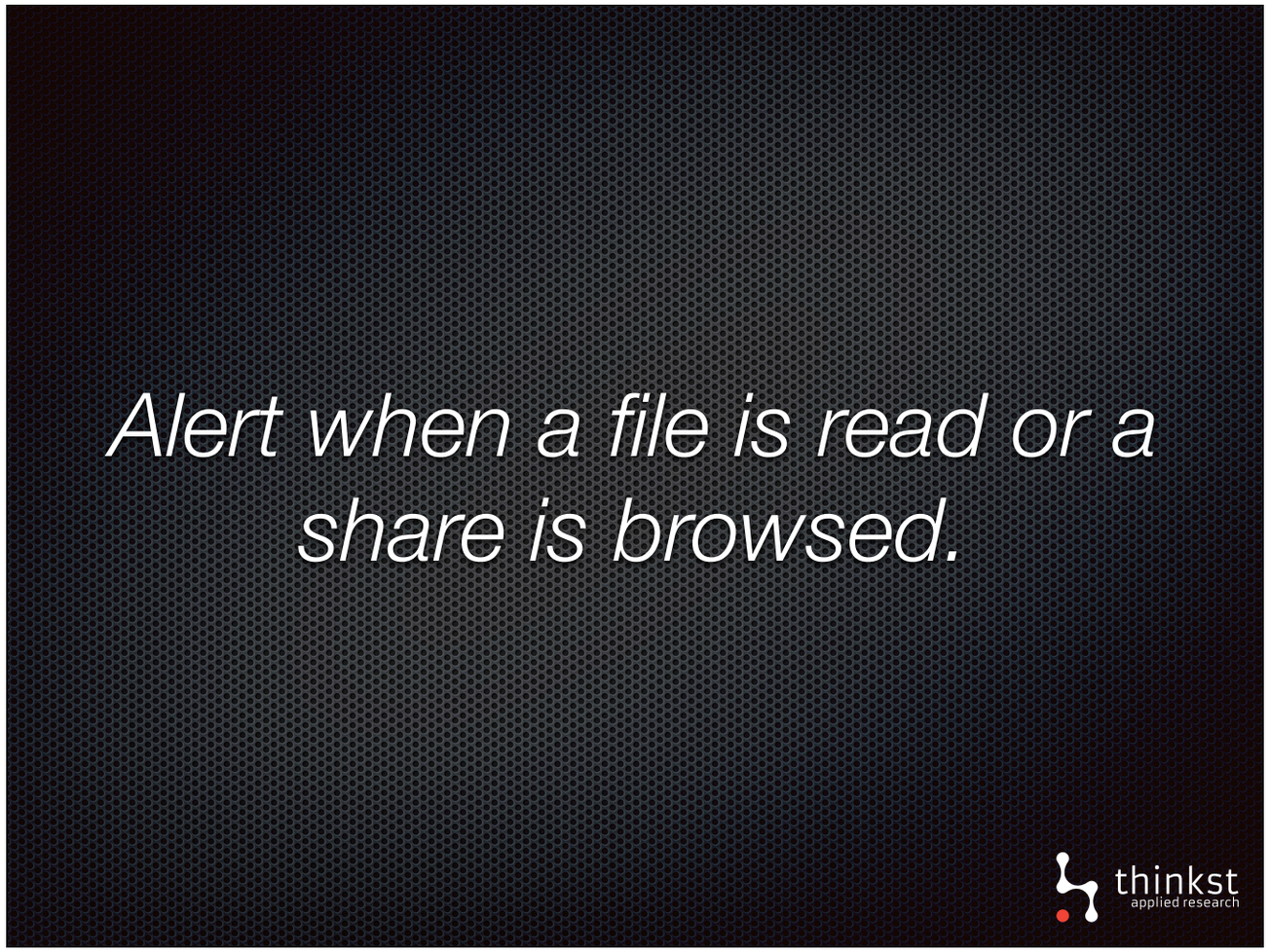

The venerable, most ancient, desktop.ini does the trick

thinkst
applied research

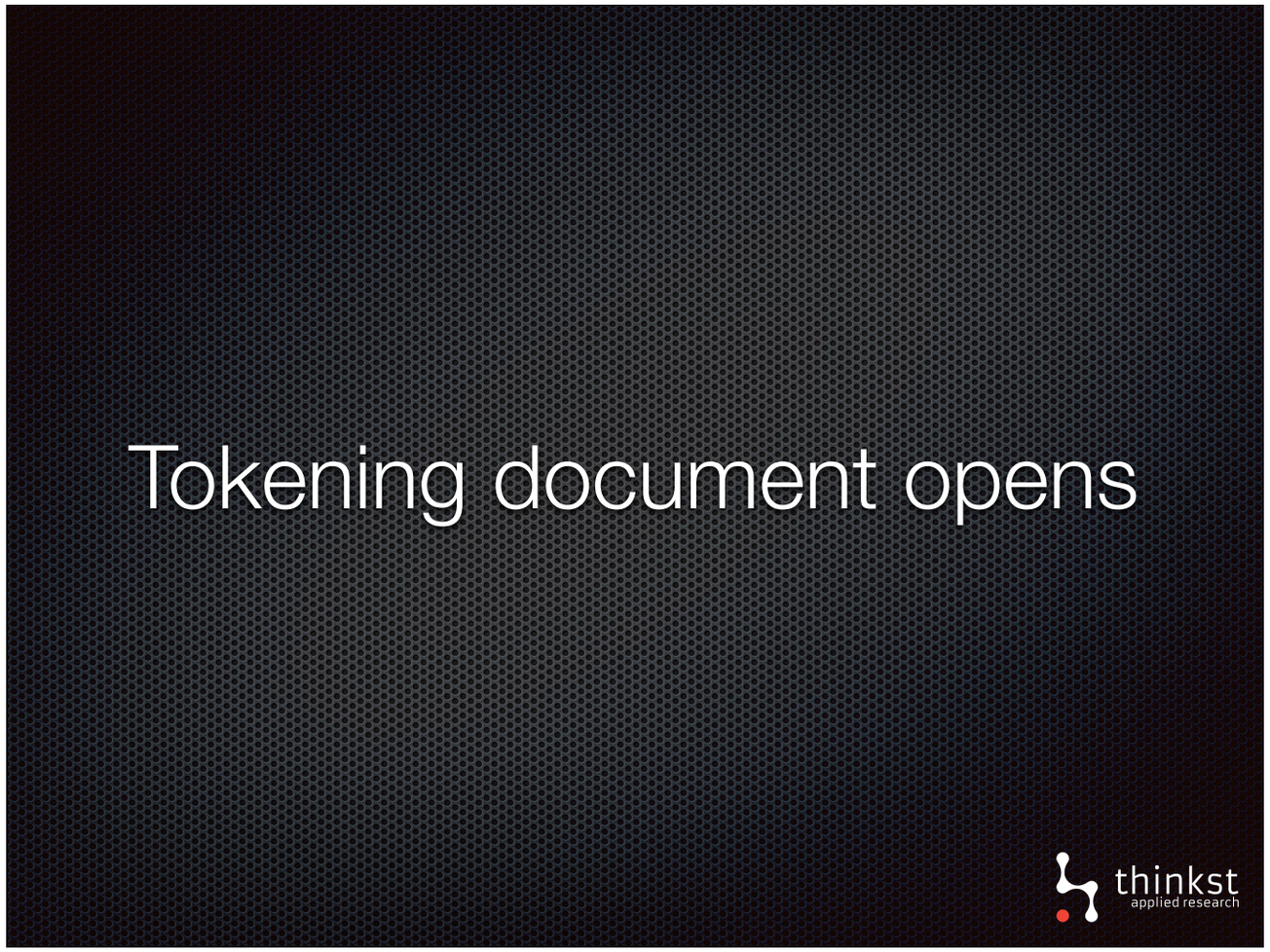
- On Windows, we can use the classic desktop.ini to specify an icon for a folder.
- The icon is a remote UNC path, we can set that to be be a DNS Canarytoken.
- So when explorer browses the folder, it tries fetch the icon , triggers DNS lookup.
- The icon path looks like that. It has the added advantage, that we can report the windows username that did the browsing, and we can package up the .ini file in a zip archive to let us know when those are extracted.

# Demo: Share browsing alerts


- Quick demonstration of triggering alert when browsing to a folder with token already setup.




- These techniques, allow us to leave fake directory trees lying about and detect if file is read in the tree, or directory browsed.
- Unlike the normal users, attackers looking for valuable data, will have good reason to check out the fake directory trees, and alerting us in the process.




- When scattering fake tokened or trap documents around, an attack could copy them and open later or pass them onto to someone else.
- Unlike in the last case, if file reads aren’t being monitored, it can be useful to be alerted to the when the document is opened sometime later.
- Since there’s no reason for fake documents to be opened, it’d be worth investigating who had access to the fake documents and it’s neighbors.

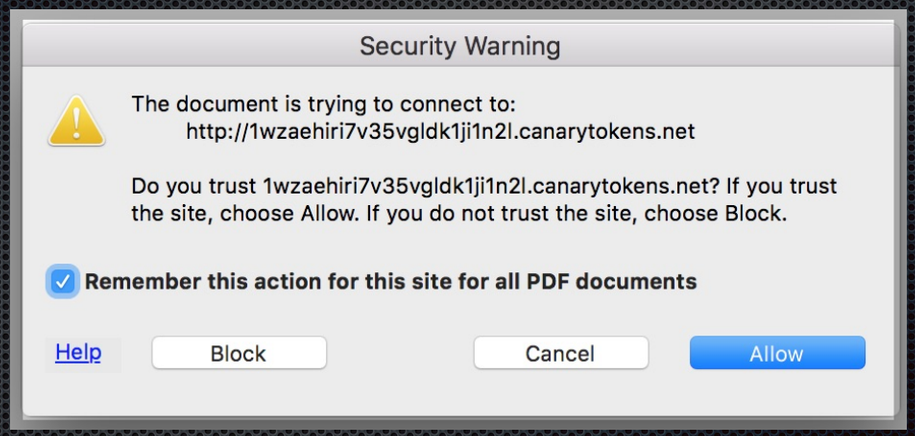
# File opens: Word

- Standard web bug using builtin Word fields.
- Cross-platform.
- No script or prompts necessary.

thinkst
applied research

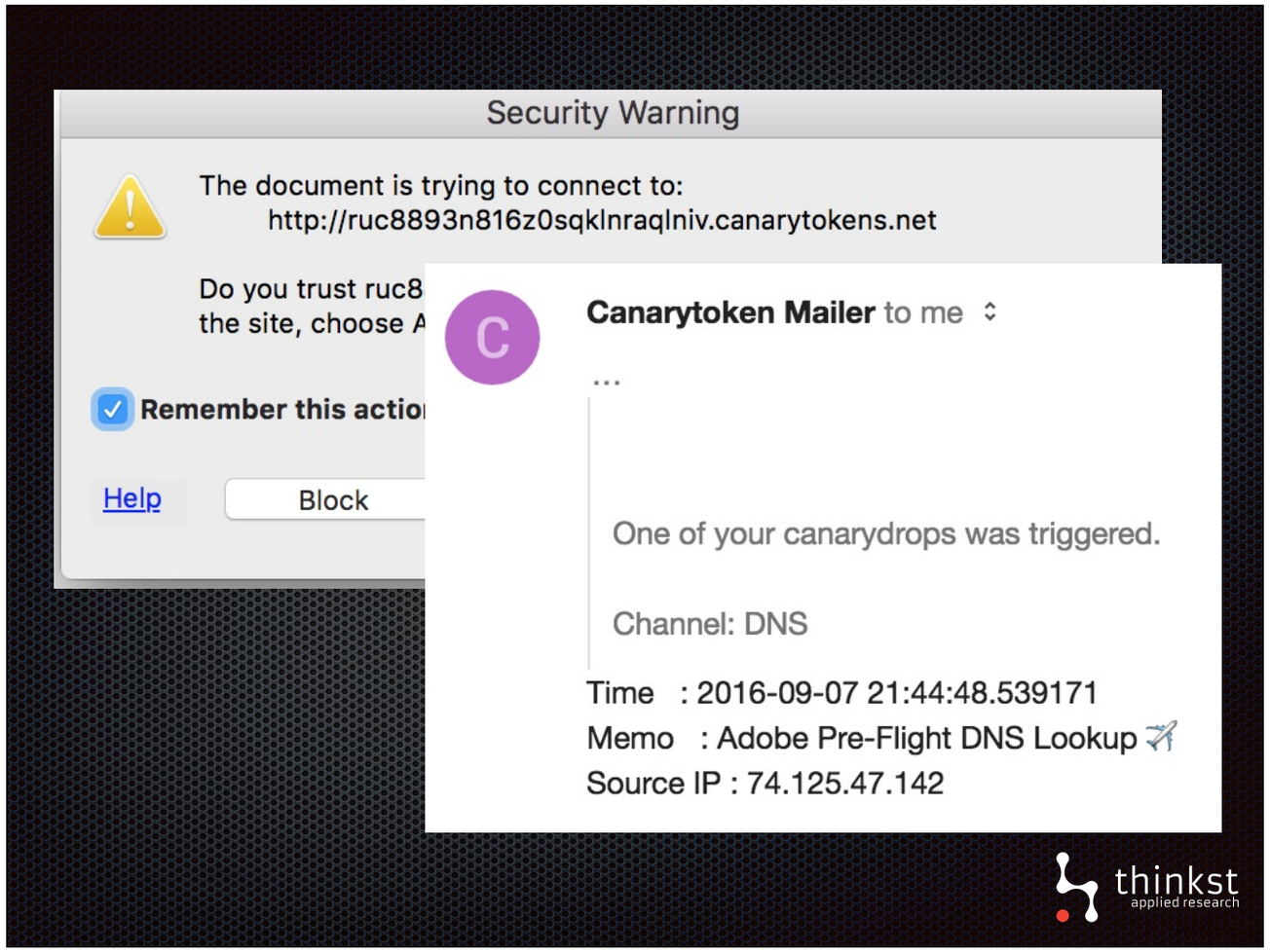
- For Microsoft word documents the straight forward web bug works, silently across platforms. On canary tokens these documents are generated automatically with a token that will send the alert.

# File opens: PDF

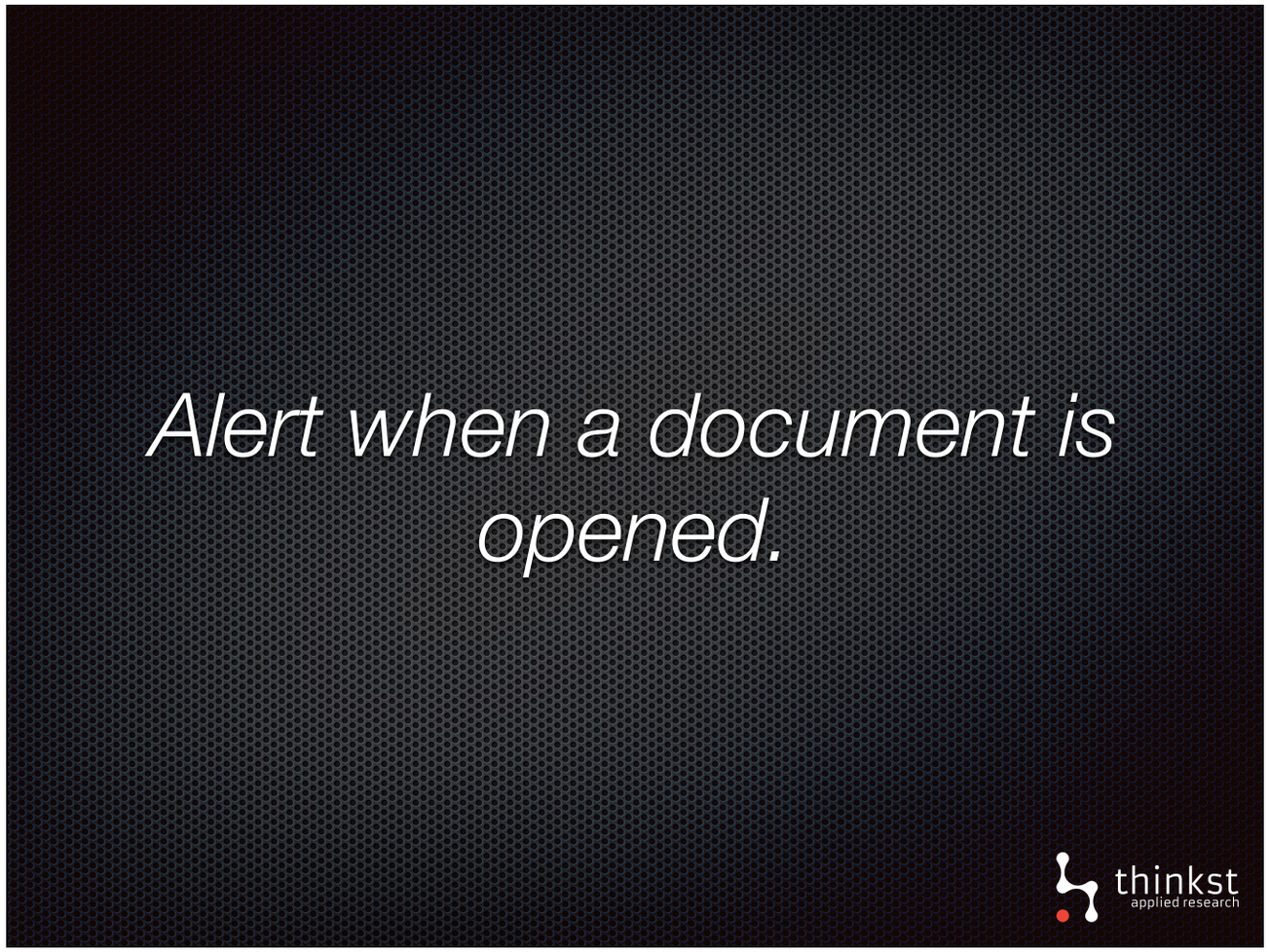



thinkst applied research

- AdobeReader isn't that straight forward. If you embed a token URL, you'll get prompted to whether to allow the connection.
- However, there's an interesting way around it. AdobeReader, will pre-flight DNS requests for the URL - if the DNS is configured in particular way. We can use that to get an alert out.

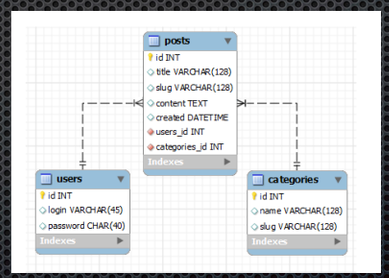



- The net result is that prompt still get shown, but it doesn't matter, as AdobeReader will have alerted us already via a pre-flighted DNS request.
- On Canarytokens, the DNS is setup correctly, and you can simply download an auto-generated PDF file with a token.



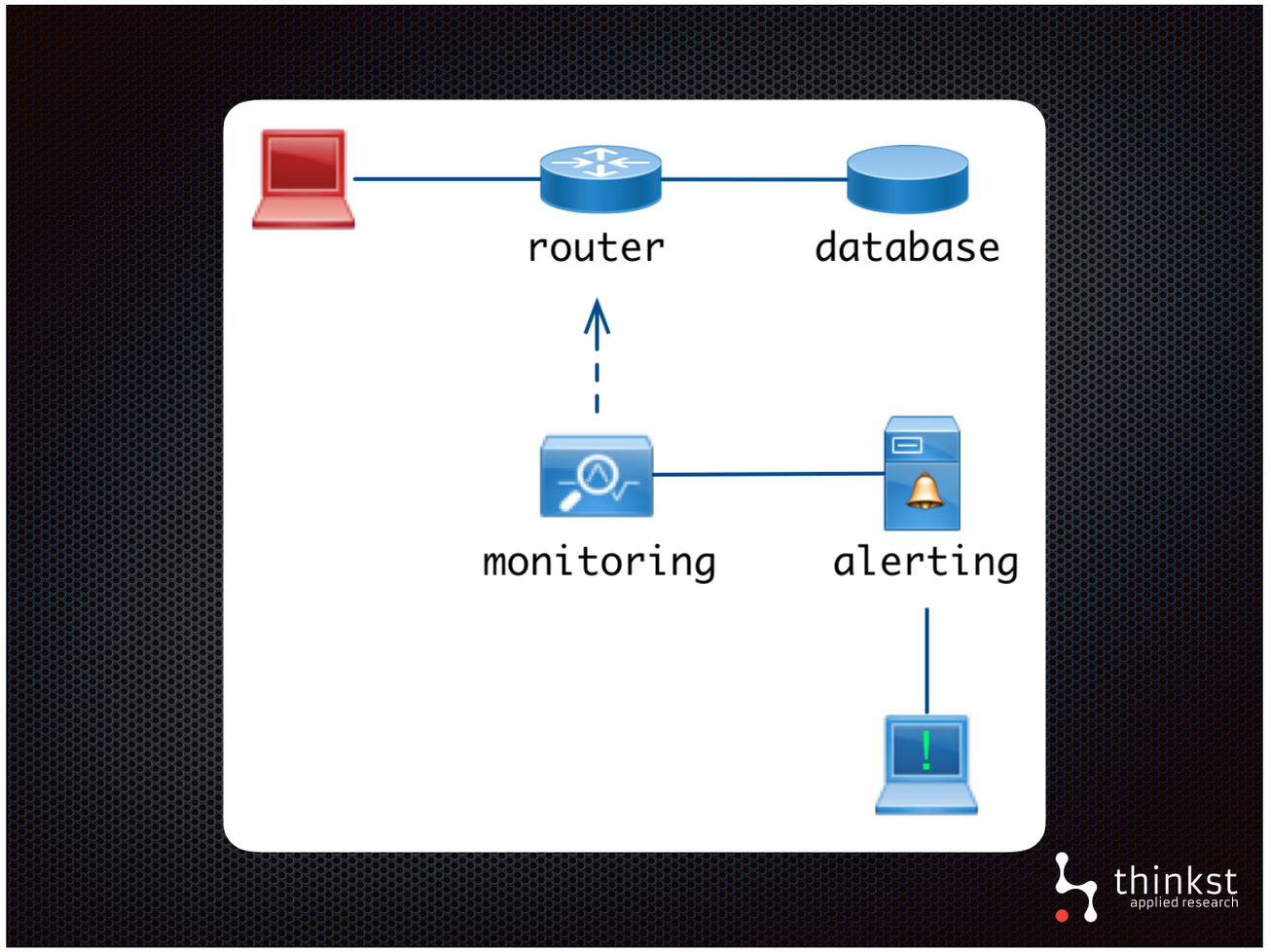

- Those two techniques are examples of how we can easily generate tokened documents that alerts us when a document is opened, even if it’s long after the document has been stolen, and it’s opened elsewhere.

# Alerting on database queries

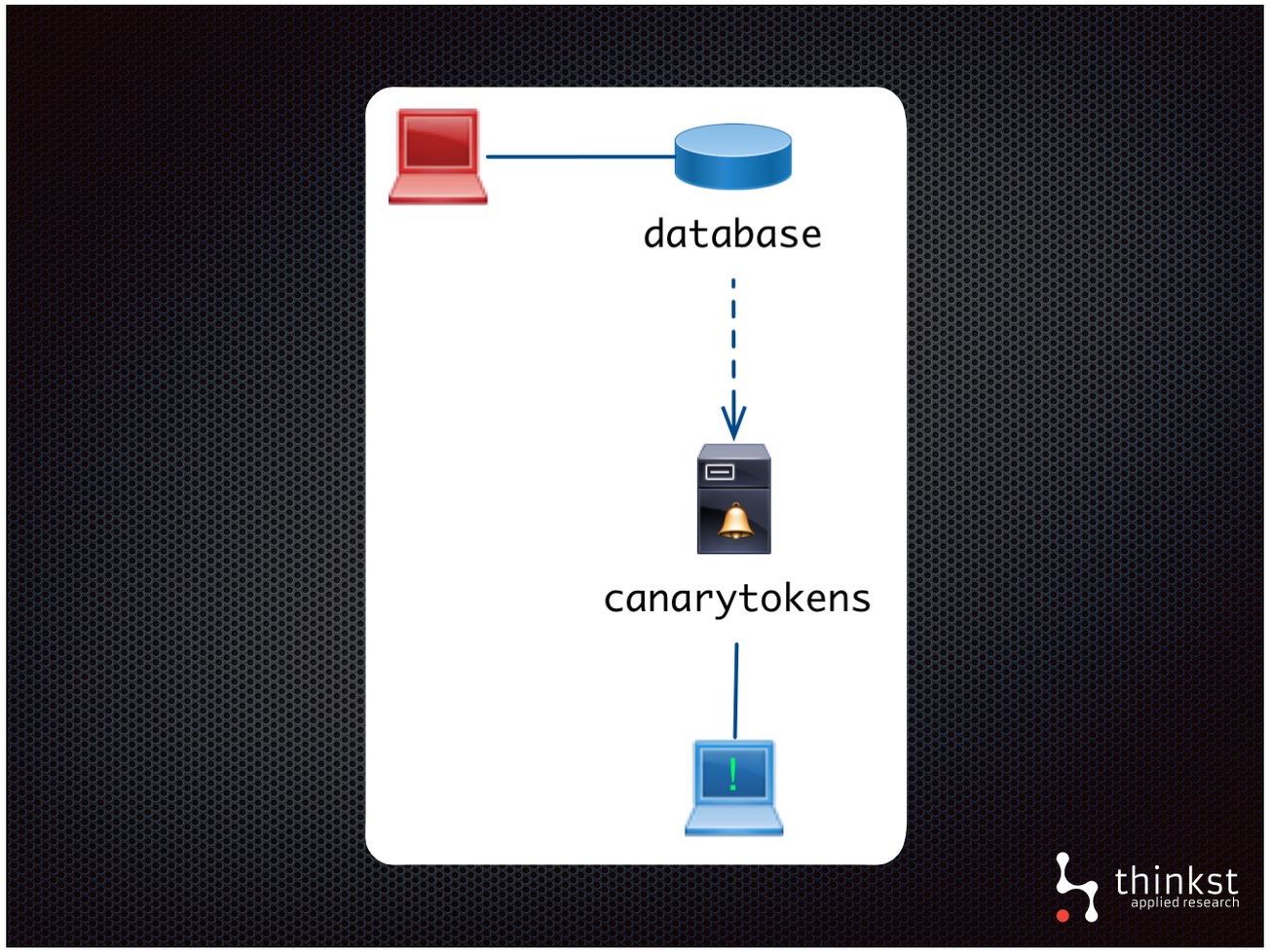



thinkst
applied research

- Moving away from files and directories, let's look at example from databases, in particular SQL server.
- It'd useful to be alerted when someone is snooping around inside a database. Attackers inside a database they know nothing about, will map it out first, and then go for the valuable or useful data.
- In theory, if we had something in the database that could as act as trap, we could dress it up to be attractive to this sort of attacker.




- As we pointed in the introduction, early passive tokens approach has serious shortcomings. It requires infrastructure to actively monitor network traffic in plain text and an alerting server to maintained.
- We can dramatically simply this.



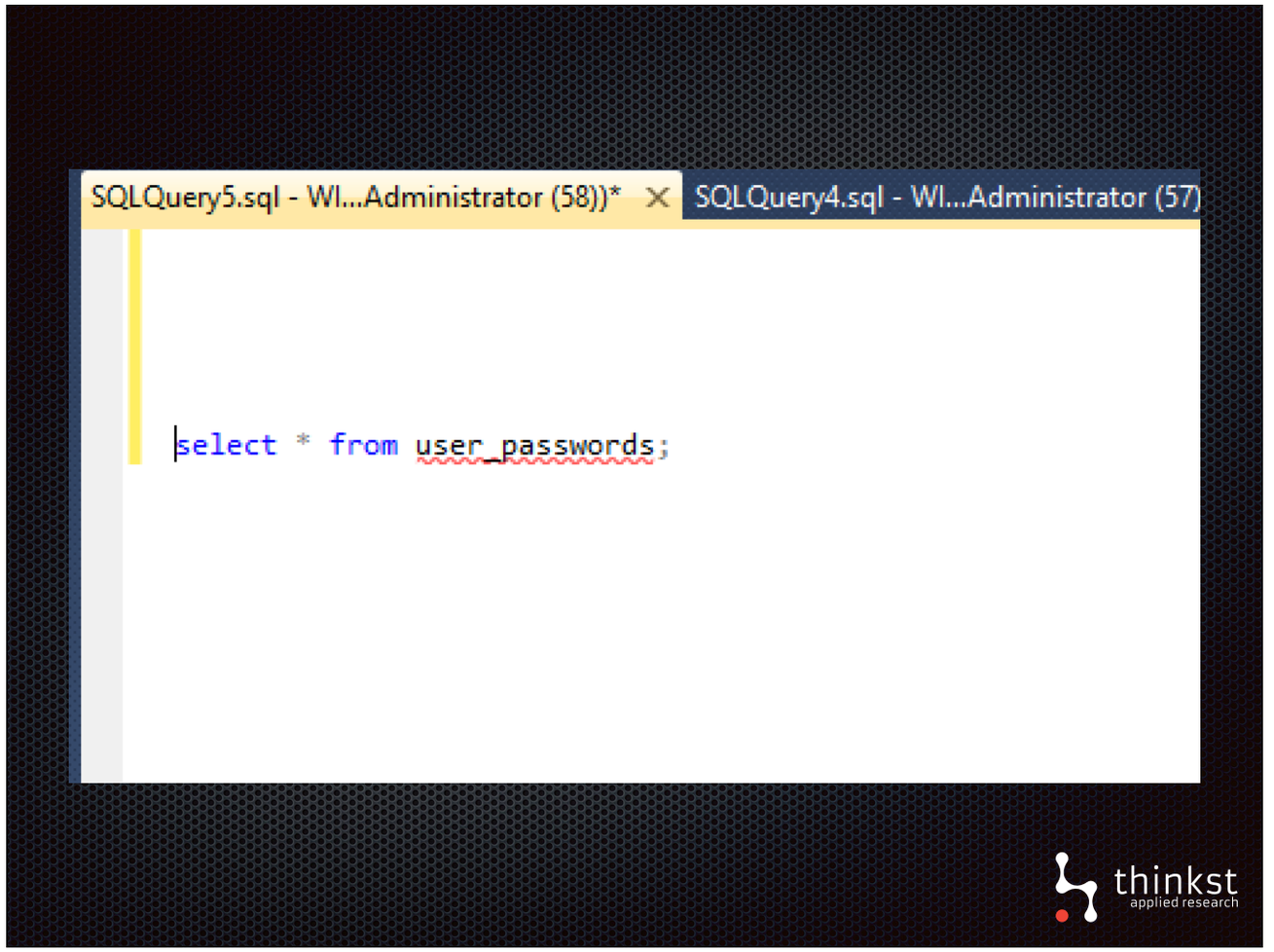

- They key to simplifying this, is to create token data in the database that when it’s read, that automatically triggers the DNS token. Since the database triggers the token, nothing else is needed as canary tokens can handle sending the alert.
- This works by creating a somewhat complex table-view, that triggers the DNS token. However, canary tokens generates the SQL script to create the table-view for you.

SQLQuery5.sql - WI...Administrator (58))* SQLQuery4.sql - WI...Administrator (57))* SQLQuery3.sql - WI...Administrator (57))* SQLQuery1.sql - WIN....master (user (53))

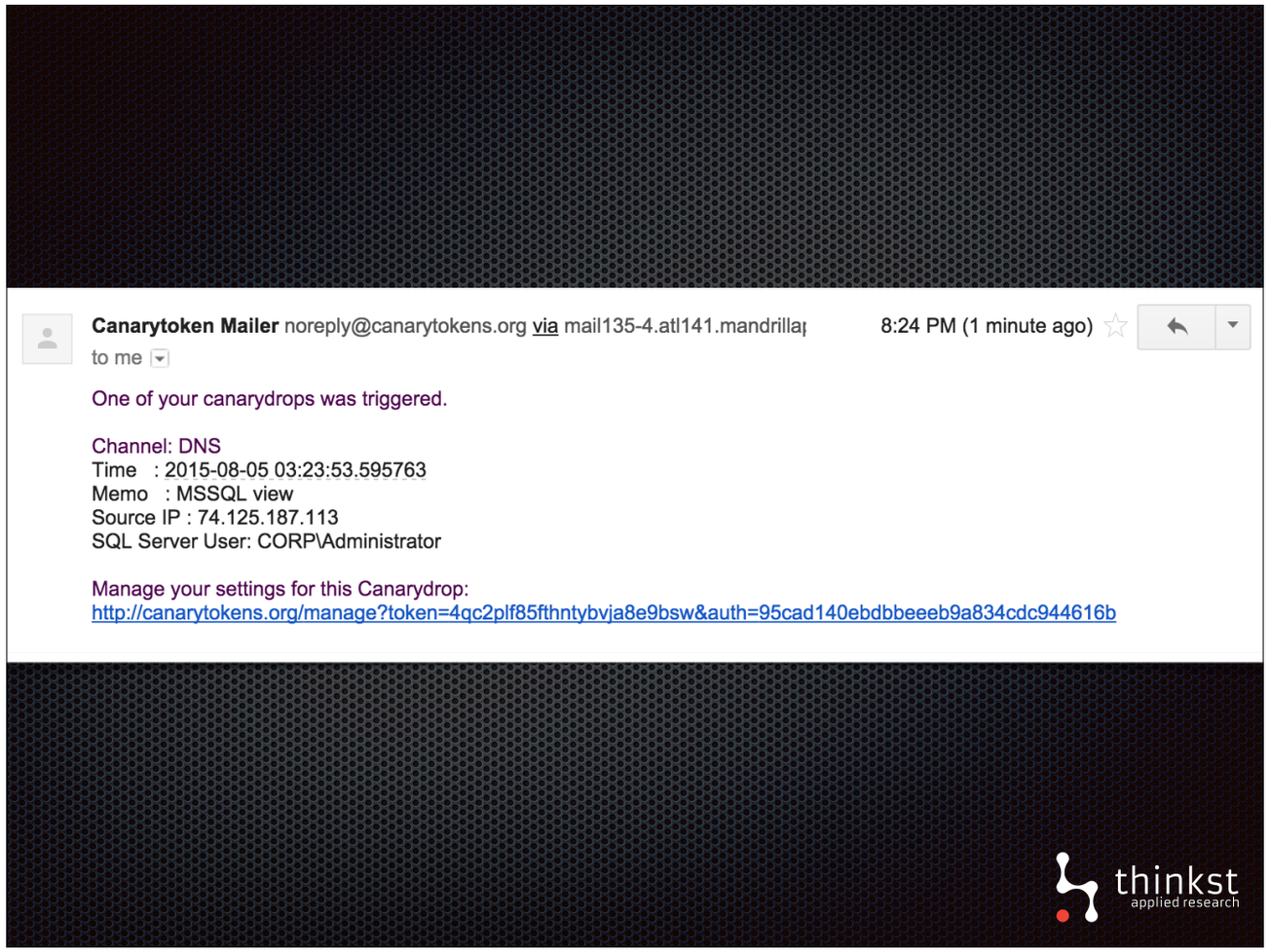
```
CREATE function query_users(@RAND FLOAT) returns @output table (col1 varchar(max))
    AS
    BEGIN
        declare @username varchar(max), @base64 varchar(max), @tokendomain varchar(128), @unc varchar(128), @size int, @done int, @ran

        --setup the variables
        set @tokendomain = '4qc2plf85fthntybvja8e9bsw.canarytokens.com';
        set @size = 128;
        set @done = 0;
        set @random = cast(round(@RAND*100,0) as varchar(2));
        set @random = concat(@random, '.');
        set @username = SUSER_SNAME();

        --loop runs until the UNC path is 128 chars or less
        while @done <= 0
        begin
            --convert username into base64
            select @base64 = (SELECT
                CAST(N'' AS XML).value(
                      'xs:base64Binary(xs:hexBinary(sql:column("bin")))'
                    , 'VARCHAR(MAX)'
                )   Base64Encoding
```

thinkst applied research

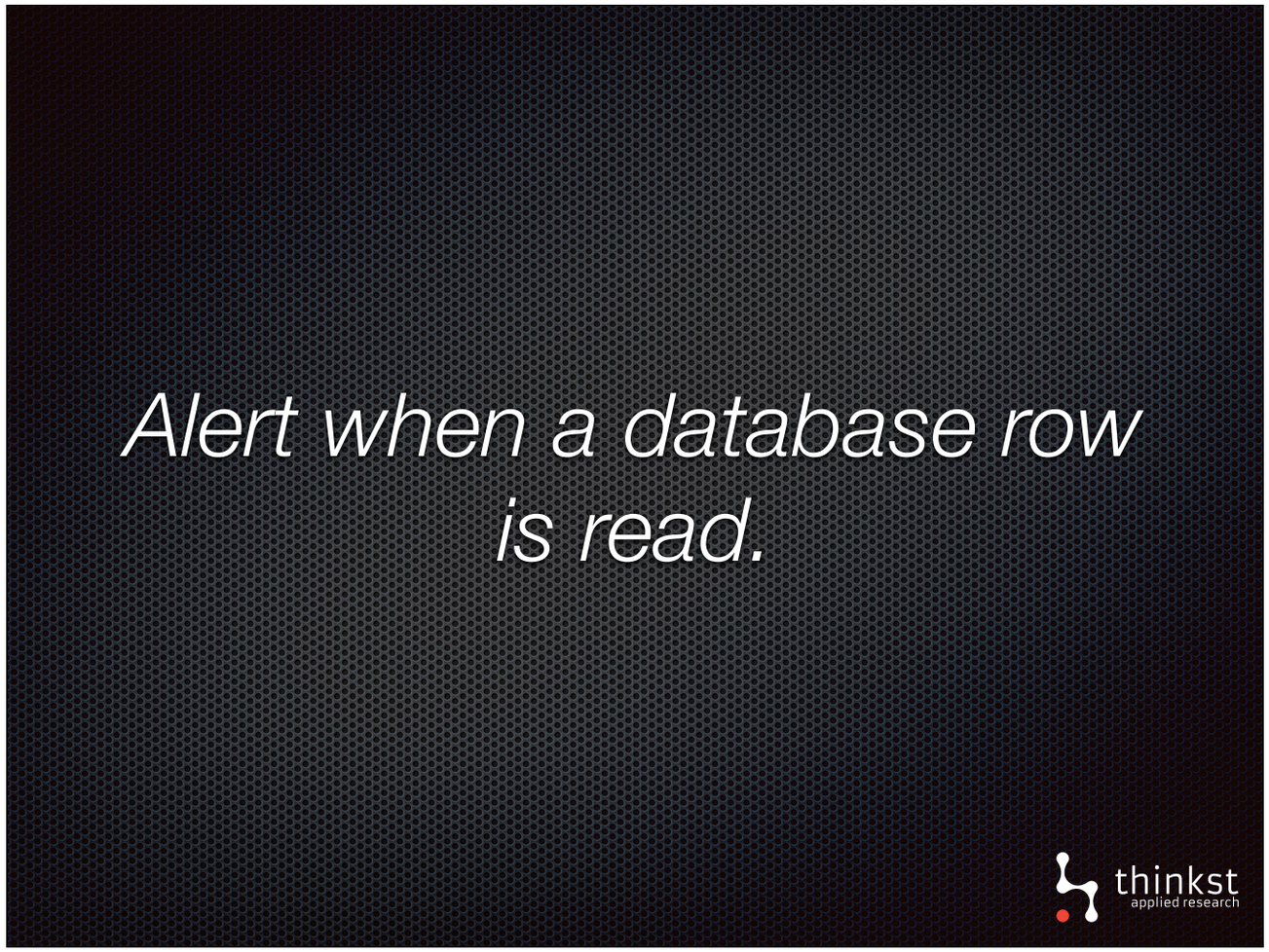
- With canary tokens  automating the heavy lifting, you can simply download and run the MS SQL script to run to create the token table-view that will alert on certain actions.




- Now when someone queries our user passwords tokened table view - by running a command like this.




- We'll get an alert as usual.




- With that automatically generated SQL script, we can easily create tokened table-views which will alert us, when someone begins exploring a database server, and reads our particular table row. It'll be well-worth investigating when that happens.
- Now, let's move to completely different topic.

# Phishing campaigns

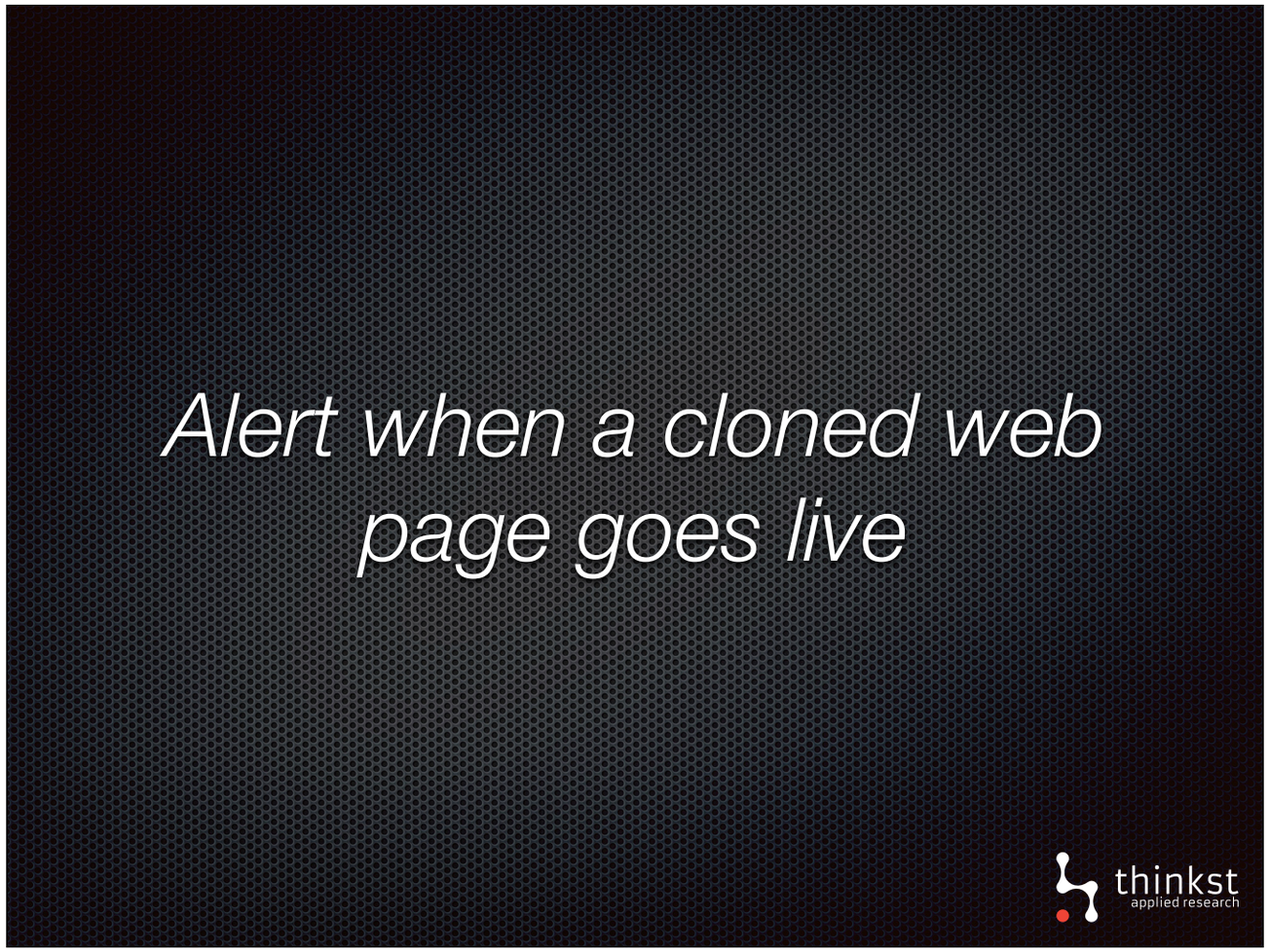

- Phishing campaigns can be quite a menace, it would handy to have some alert when they kick off.
- One place we can token, is an  early step in creating a phishing campaign, where a login webpage is cloned.

# Web page cloned

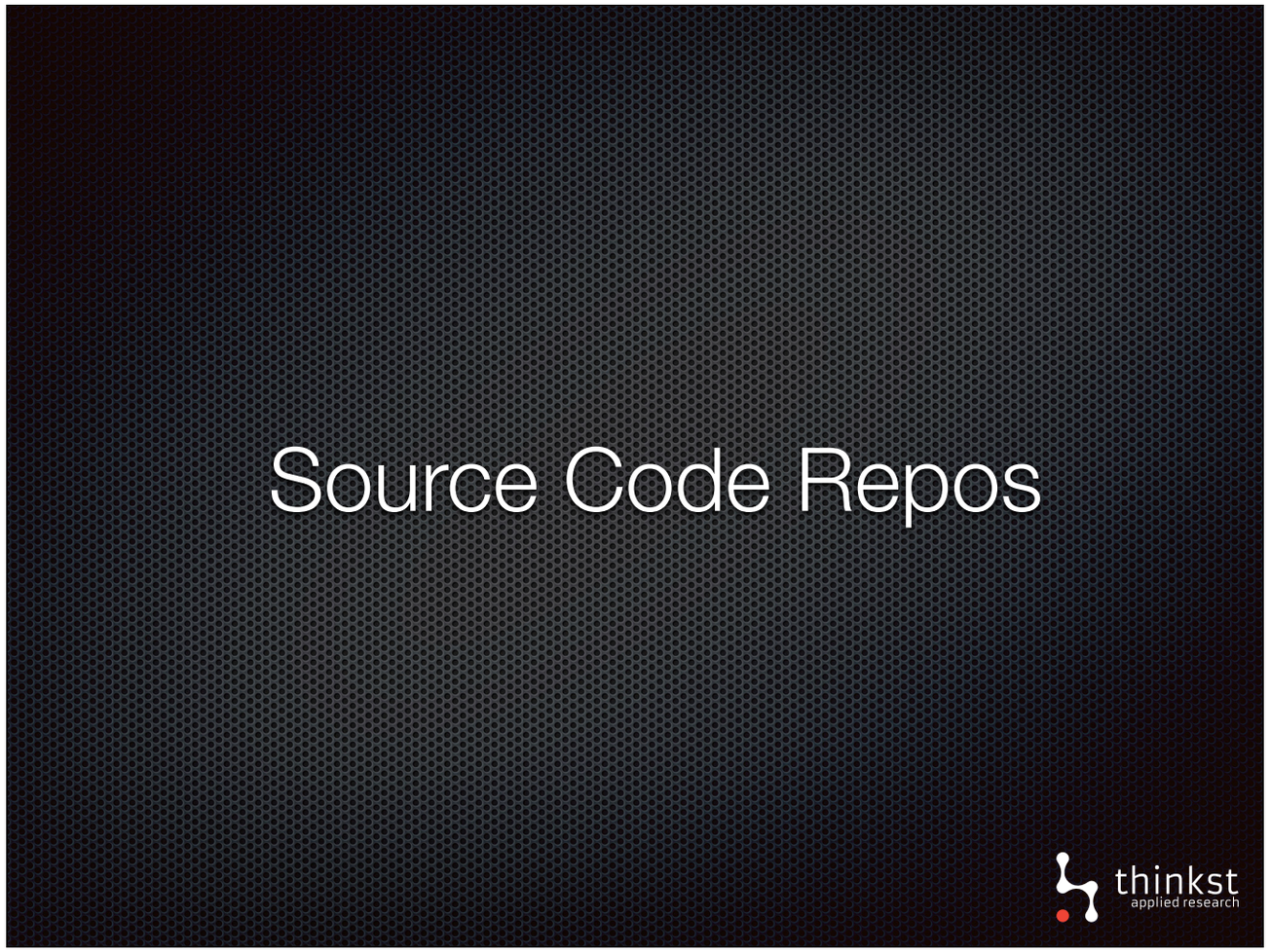
```
//Javascript runs at page load
if (document.domain != "thinkst.com") {
   //create <img> with src=token
}
```

thinkst
applied research

- All it takes is 3 lines of Javascript that on page load, checks the domain of it's page. If it's not where your page normally resides, it'll trigger an alert. (You could even get an alert before the cloned site is up, if the person running the campaign is tests the page locally first.)



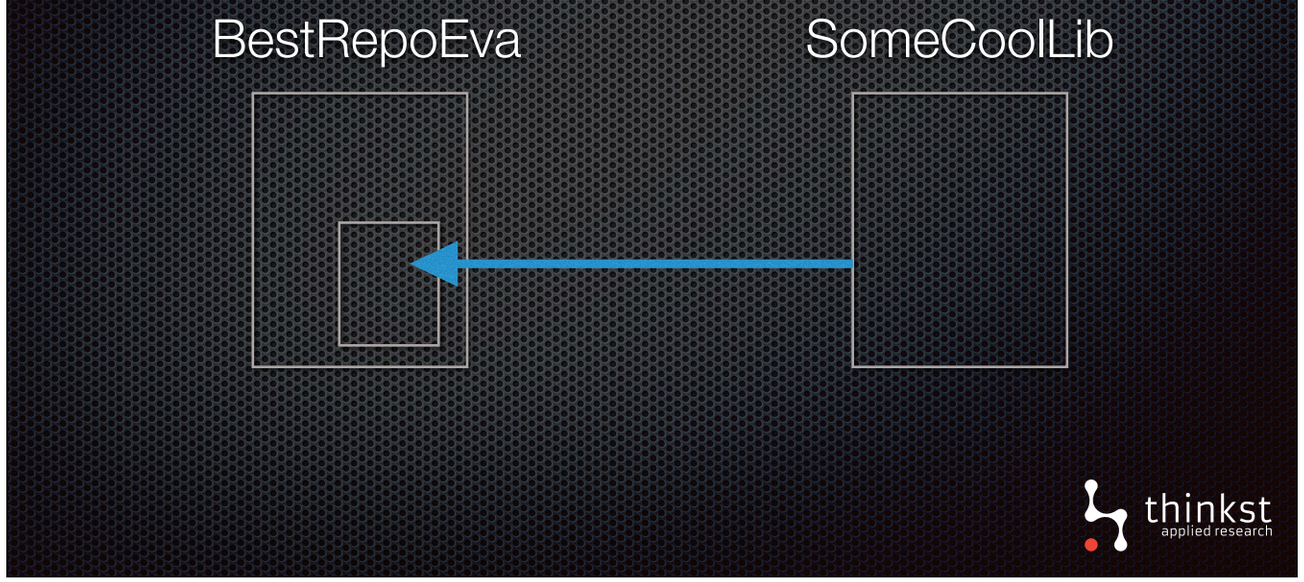

- That’s a quick token that works quite nicely to report when a clone webpage goes live.
- This technique is a good example of where many logging setups would have more difficulty detecting this event, but tokens can easily flag on it.




- Let's now look at source code and, specifically source code repos - most organisations have them and they contain source code from current projects as well as old or completed projects.
- Usually, access to the repos are limited but sometimes this access is not governed properly, or someone gets credentials they shouldn't have and is able to pull sensitive source code.
- Being alerted on this pull would be useful in being able to act quickly and stop any leak of the code.


- Let's look at Subversion, a popular versioning system.

# How it works (simply)




- Subversion has an idea of externals definitions which are simply local directories mapped to a url of a versioned directory.  This let's you include remote repos into other svn repos.
- Using this idea, we simply replace the externals URL with a tokened URL.  When a repo is pulled the token is hit and you get an alert.

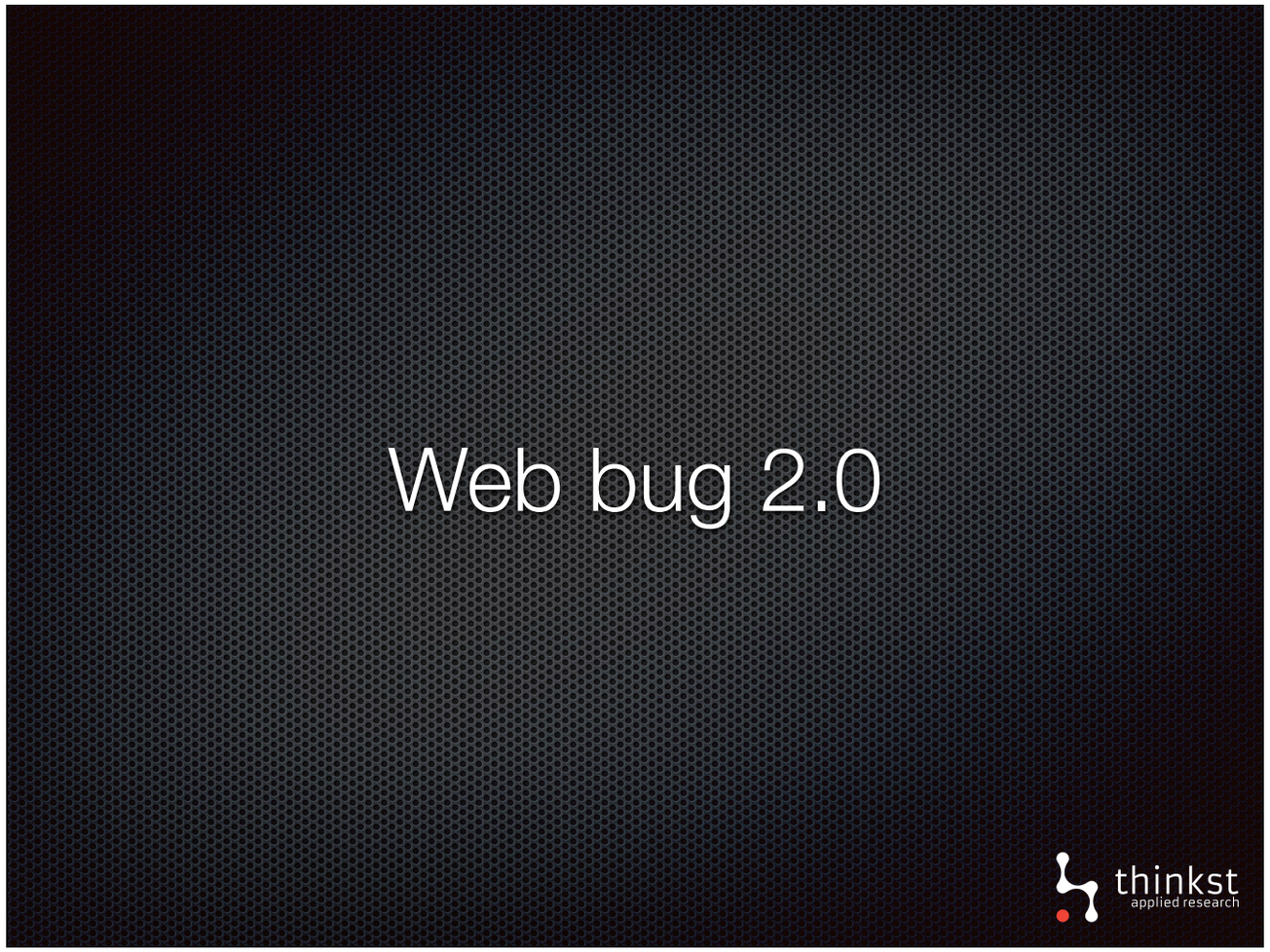
# Demo: Unauthorized clone of Prototype SVN Repo

thinkst
applied research

- Quick demonstration of triggering SVN token by cloning a repo.


- Let's recap: We have a subversion repository that no one should be accessing, whether it is a completed project, or a fake project left for intruders.
- We added an externals link that triggers on any pull of the repo.
- If someone comes along and tries to clone the repo, we are alerted and can act accordingly.


- Many of you may be using GIT instead and want a similar solution.
- There is, and there isn't, at least for what we have found.
- Simply, git uses submodules which are similar to externals in subversion but they aren't pulled automatically.
- So you can add a submodule and if anyone ever runs git submodule update you will be alerted.




- Let's now relook at our simple web bug.
- We showed that it is simple a URL that, when hit, triggers an alert. Typically, web bugs serve a 1x1 gif, ours does too.
- This has its uses (for example, you can put it on a page and no one will notice), however, why not let us serve an image?
- So our scenario is as follows: we want to place an image on a page that actually gets rendered and is visible to the user. At the same time, we want this image to trigger an alert.

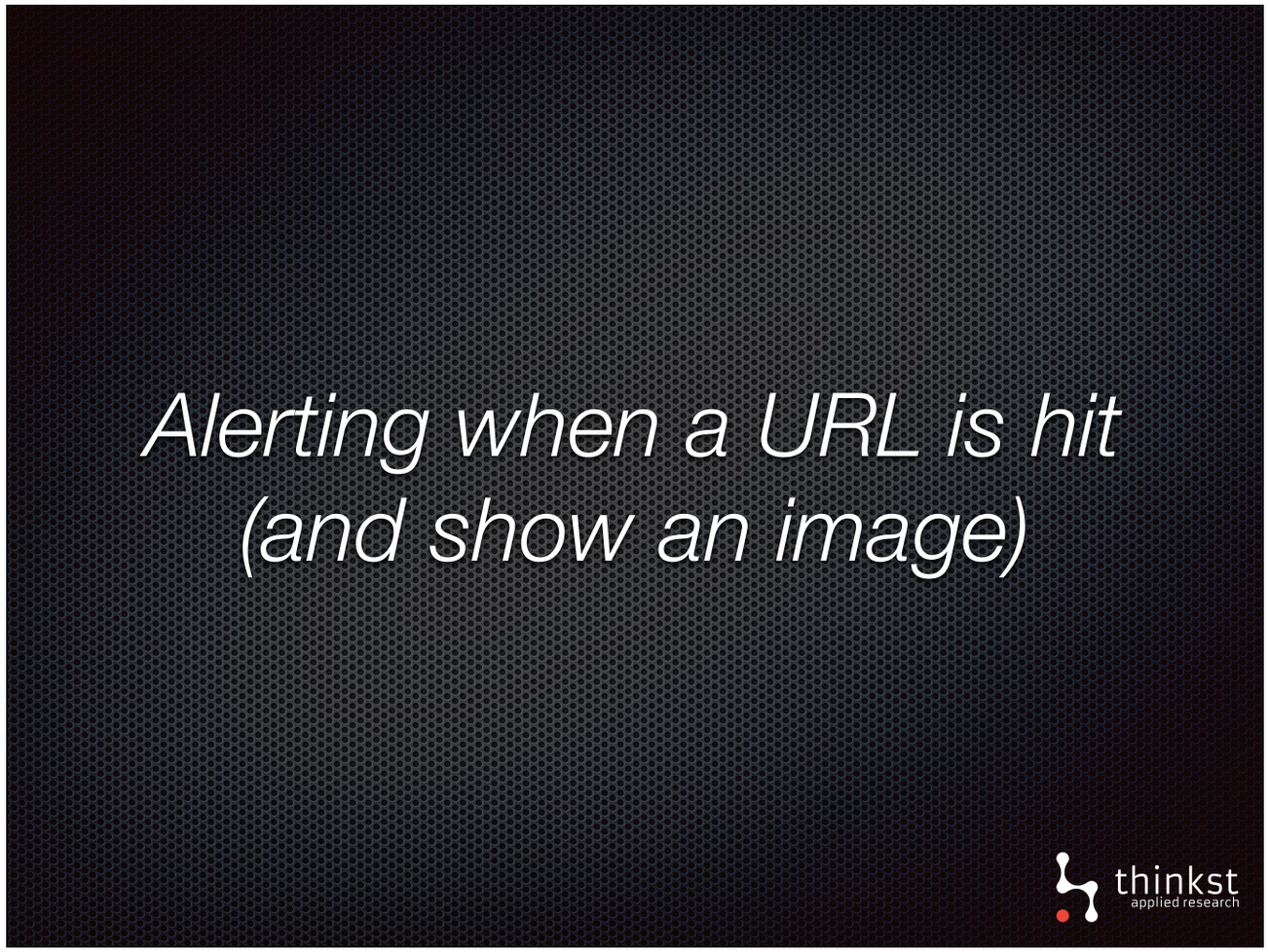
# Tokening an Image

thinkst
applied research

- Canary tokens offers you the ability to upload an image that get's served in place of the 1x1 standard gif.
- When your tokened url is hit, the image is returned as well as firing an alert.
- Let's take a quick look at this process.

# Demo Image Tokening


- Quick demonstration of generating a tokened image.

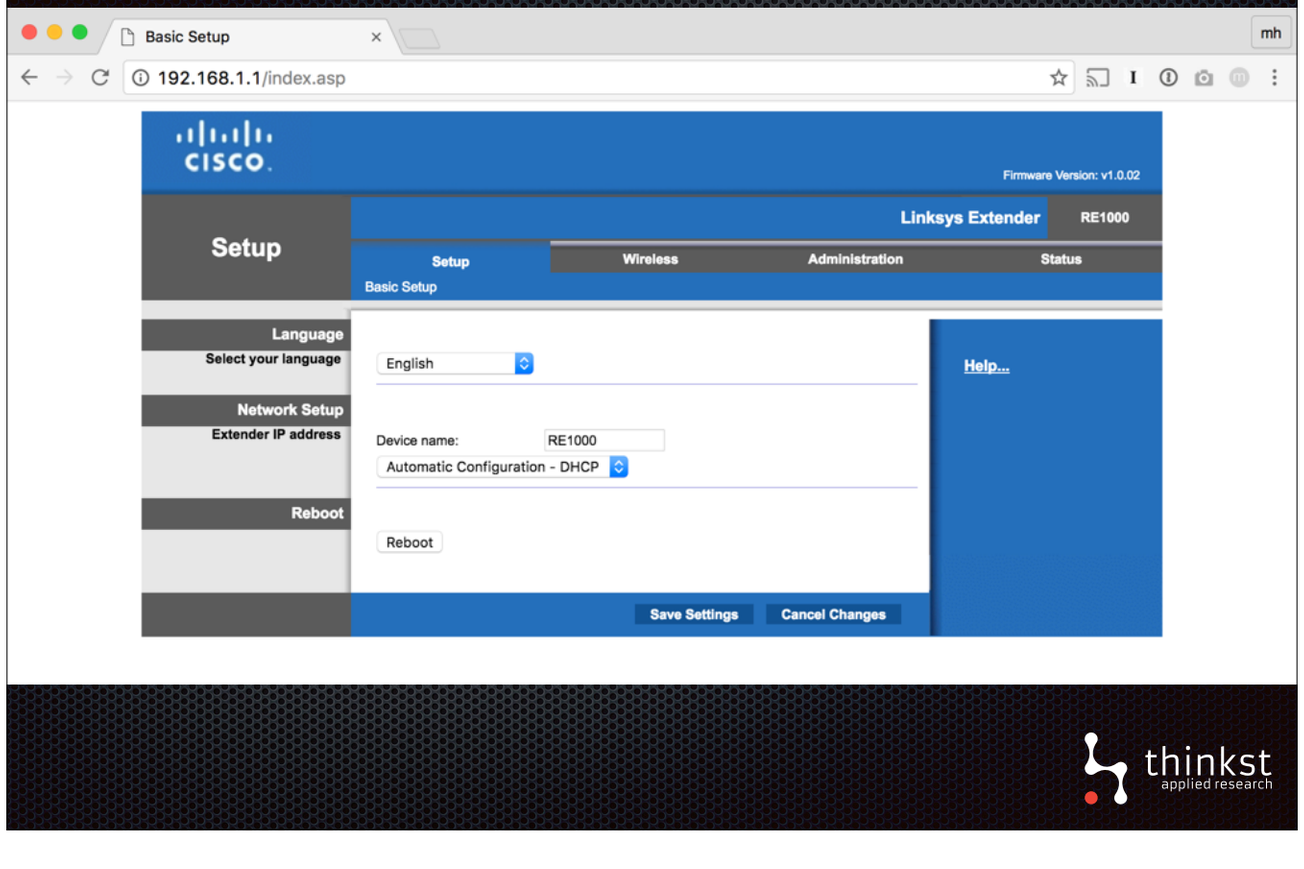



- So what have we done here, and how does it help us?
- Just like our simple web bug, when a URL is hit, you get an alert.
- However, you can now return an image to display so that the url in fact has some meaning behind it instead of only firing an alert.
- So what we have done is tokened remote images.
- Our next demonstration will help us better understand this.

# Device Admin Pages

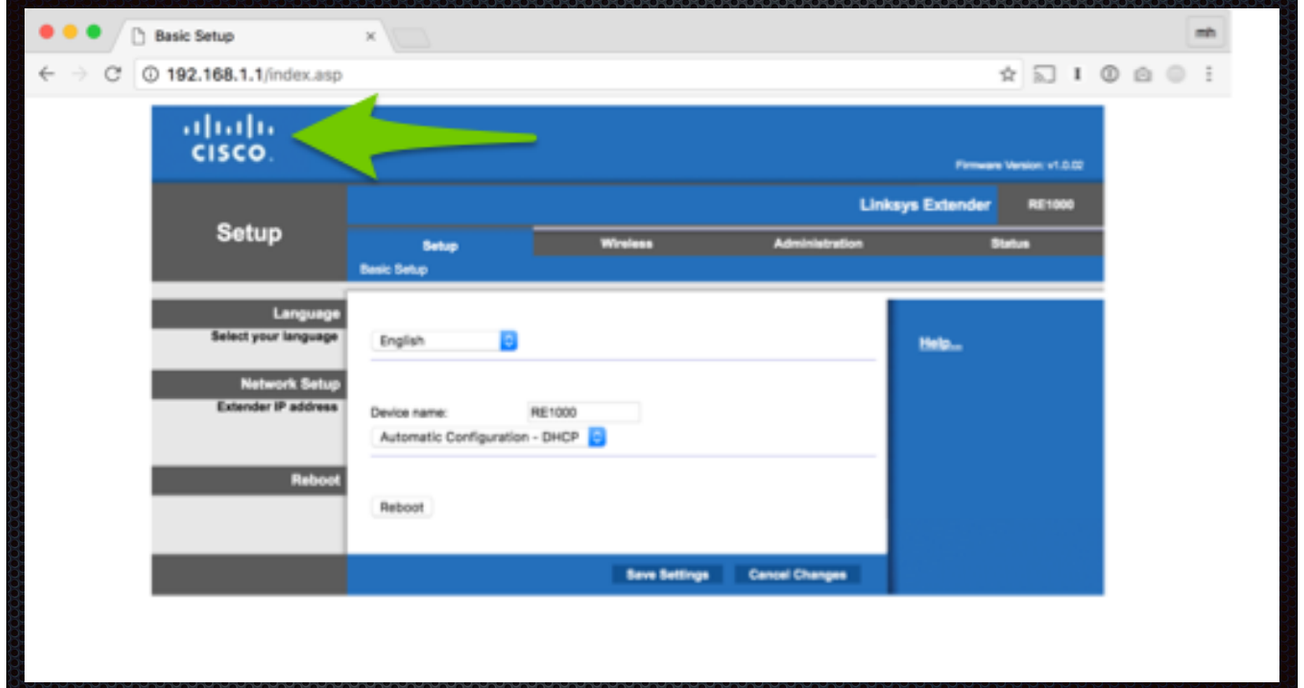
thinkst
applied research

- Device admin pages... we've all come across them on numerous occasions.
- Often people have these device admin pages that are rarely accessed and it would be great to know if or  when they got hit.
- A good example is an ADSL router config page or a wifi extender.






- Now, we want to know if someone unexpected logs into our router,
- The issue we have is that traditional logging would be difficult to integrate here.

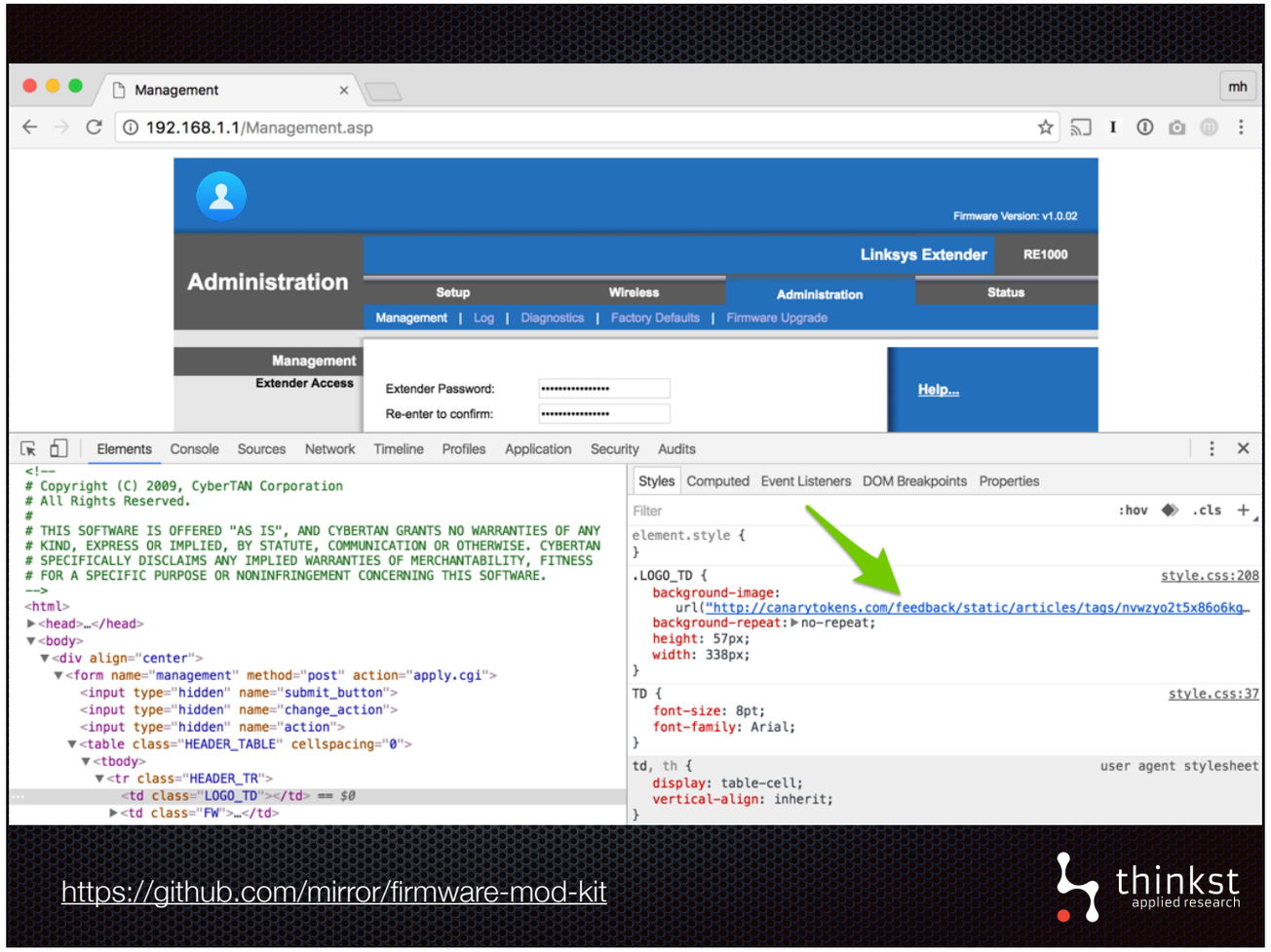



- What we do have though is a config page that is simply an html webpage with a logo.
- All we need to do is replace the logo with a tokened image.
- Now whenever someone browses to it, we'll get an alert.

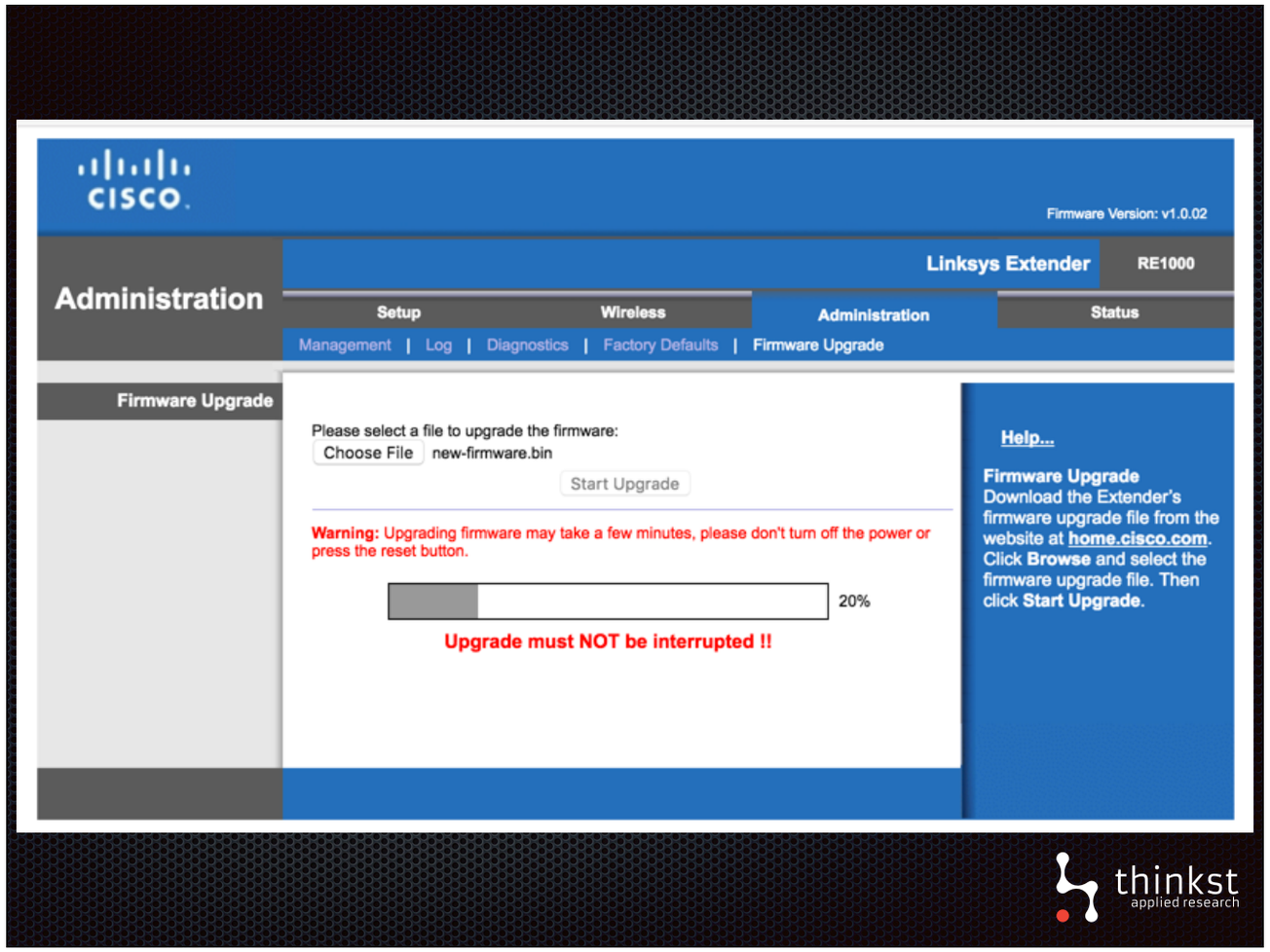
# Composable Tokens:
# Embedding a tokened Image

thinkst
applied research

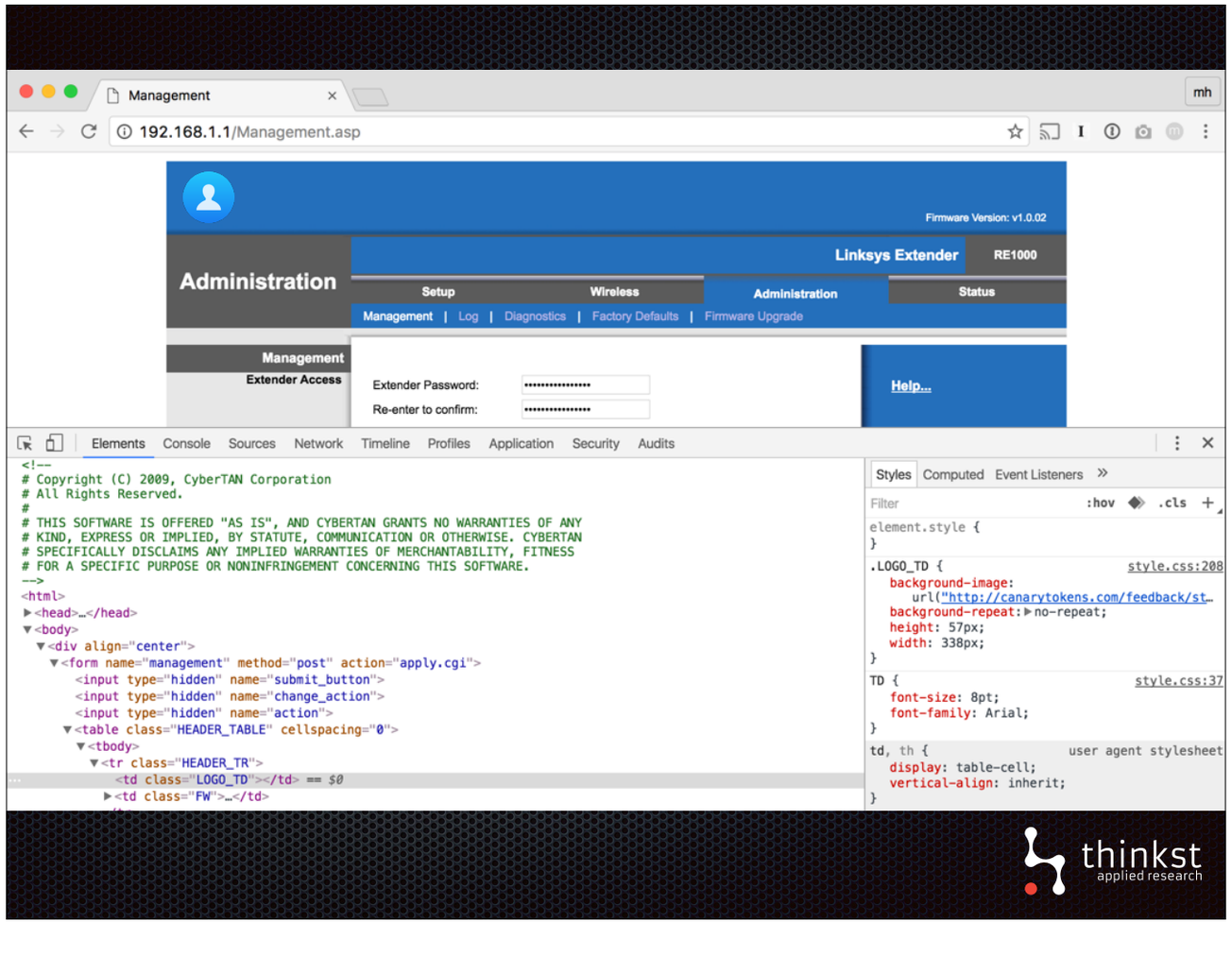
- In our previous demonstration, we tokened an image.
- Now, we are saying we want to replace a logo with a tokened image.
- This is a perfect example of being able to create new tokens using existing ones.



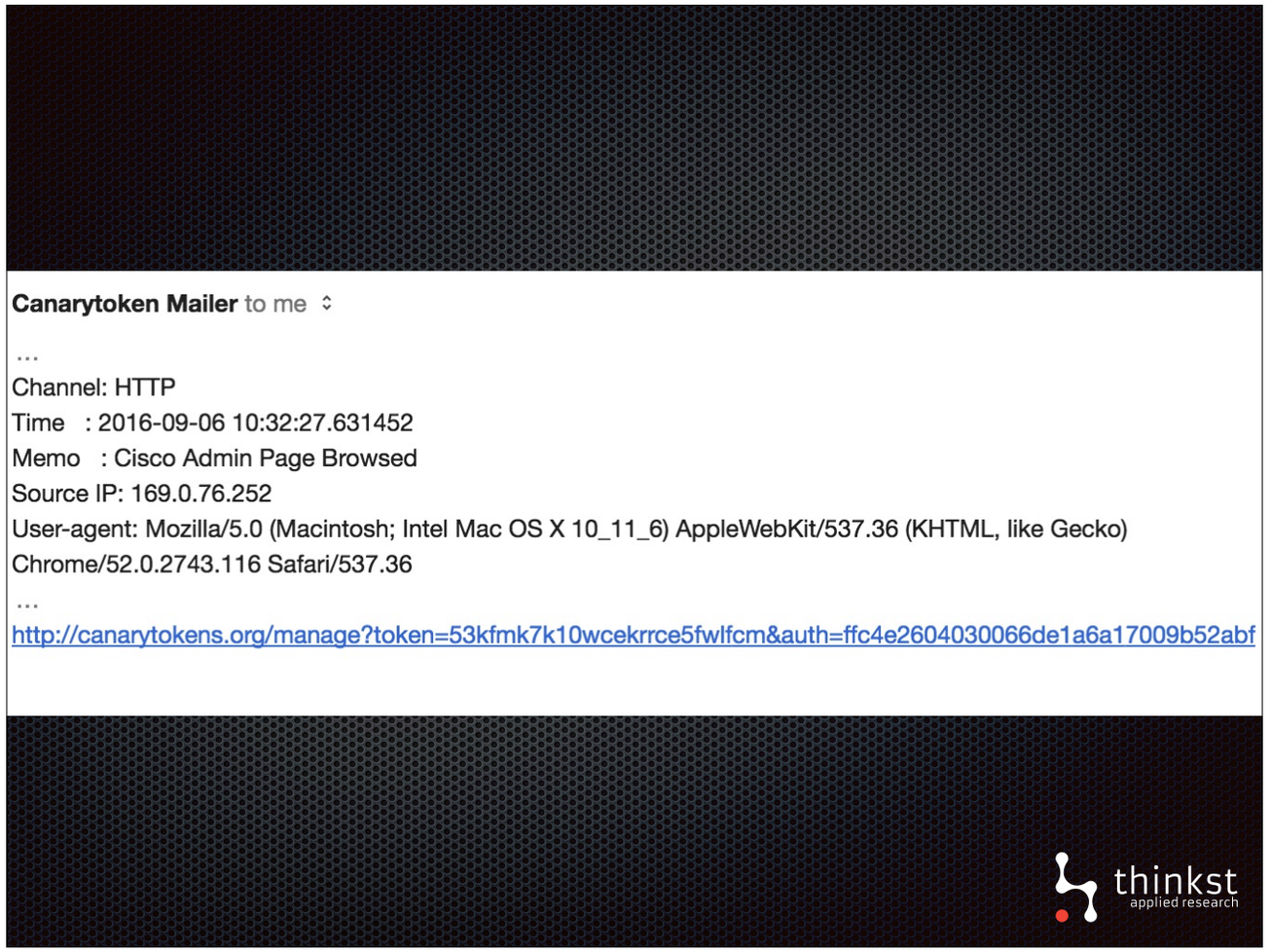

- The basic idea is that you download a copy of the firmware form the vendors site
- Unpack the firmware using  a tool
- Modify the page by replacing the logo with your own tokened image and rebuild the firmware image



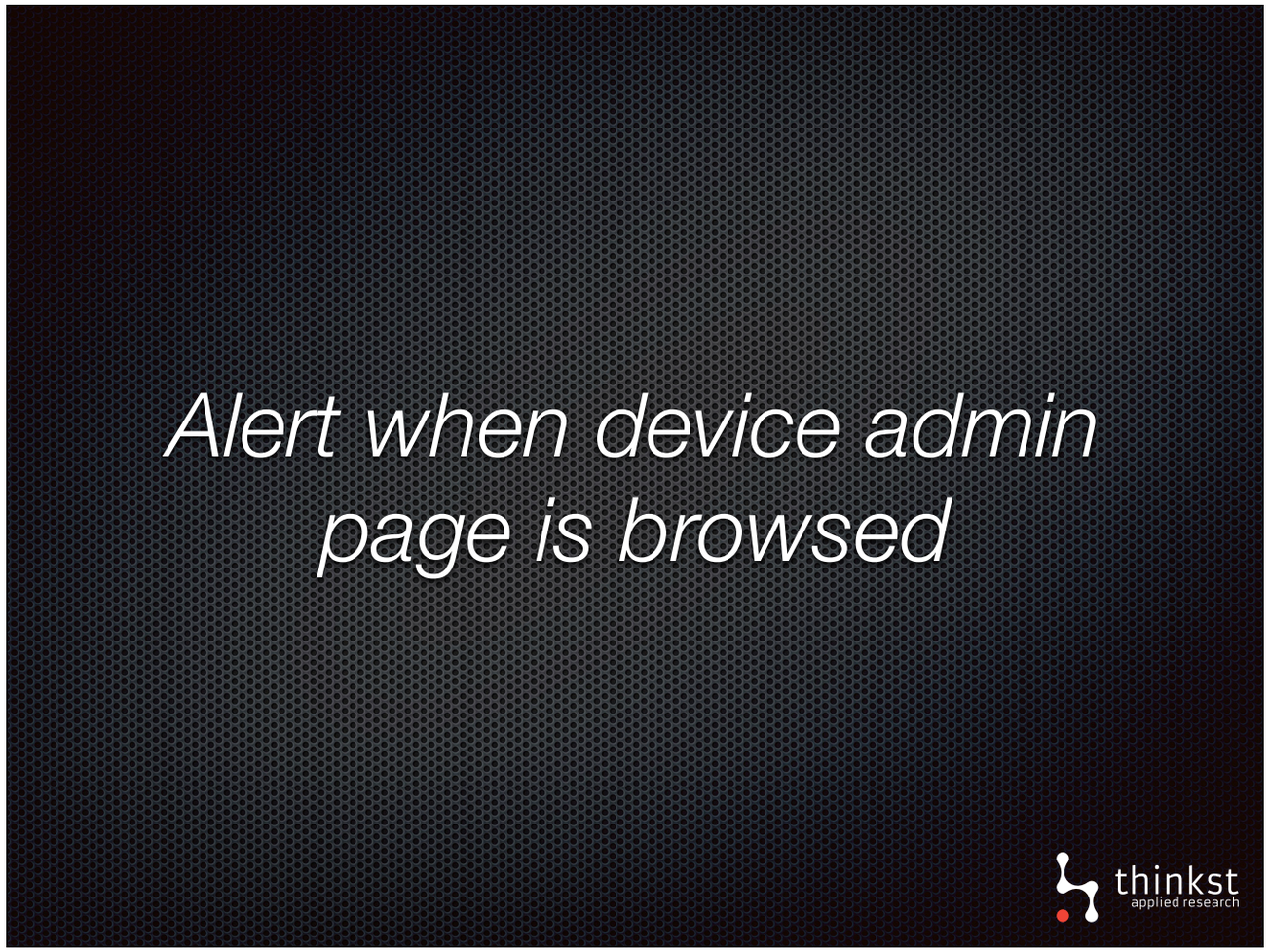

- Upload it by updating the device firmware with the repacked zip and wait for it to finish and restart.



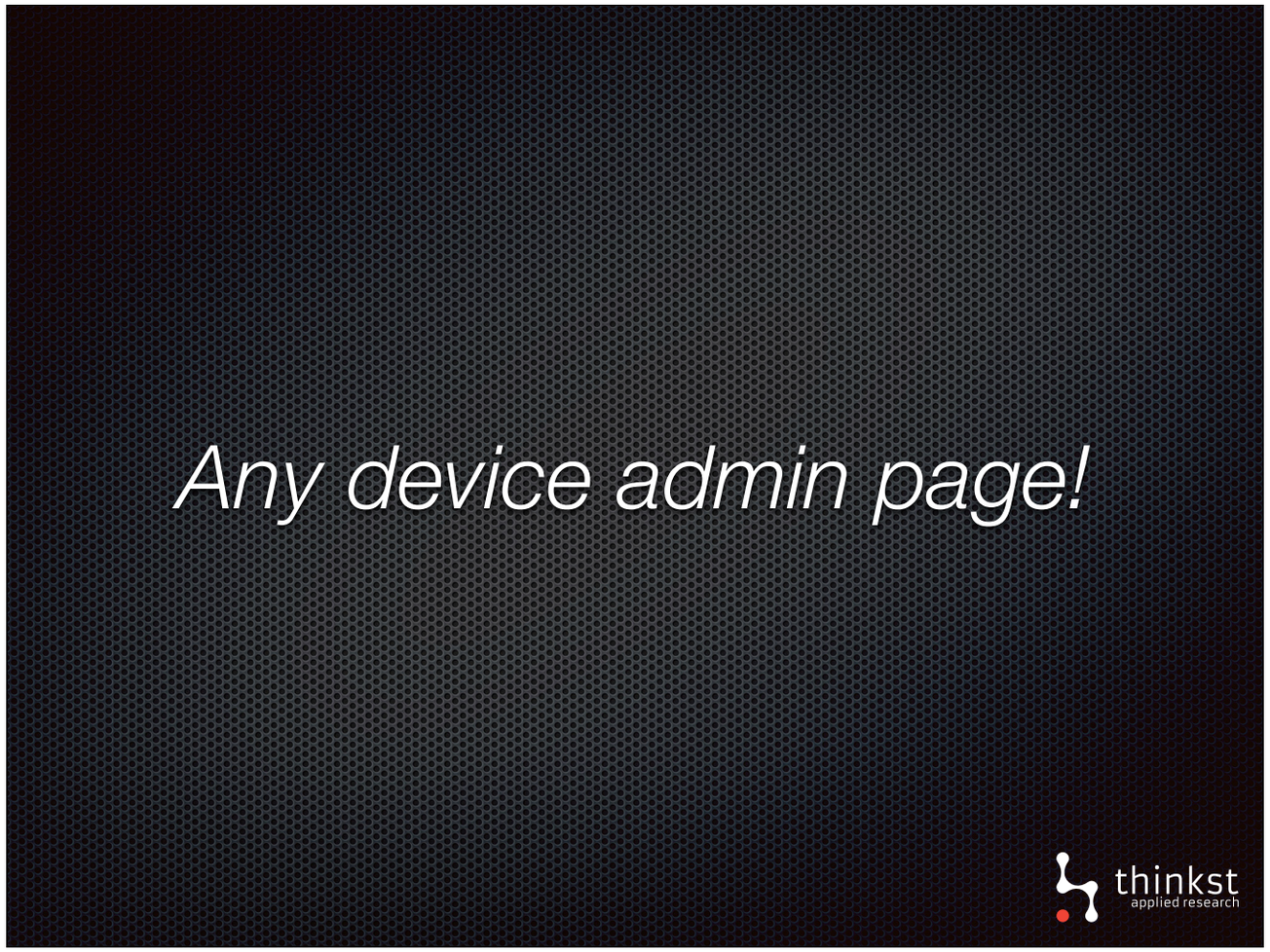

- Now when someone navigates to the admin page, the token is triggered.




- Allowing you to receive an alert informing you that someone is browsing your device admin page.




- Lets summarise this. We noticed that there was no easy way for us to be alerted if someone was accessing our router's config page.
- So, we used a previous technique of tokening an image, and used that to replace the device logo on the config page.
- This means that whenever the page is loaded, our image also gets loaded and an alert is triggered.
- You'll also get alert when a legitimate user browsers to the page, but using this will be most effective on devices where the admin page sees low traffic.




- We showed you an example of an embedded device, but there are tons of different places this could be used.
- Sys admins for example use a bunch of tools and servers and many of these come with their own admin pages.
- Tokening any off limits admin page helps to add to your chances of catching intruders moving around your network.


- Let's now take a look at the cloud. It's a not-so-new technology that everyone is using nowadays.
- What is newish (and all the cool kids are doing it too), is cloud based attacks and defence.

# Instagram Hack Reveals The Risks Of Bug Bounty Programs

After a security researcher dove deep into its systems, Facebook says it plans to review its bug bounty guidelines.

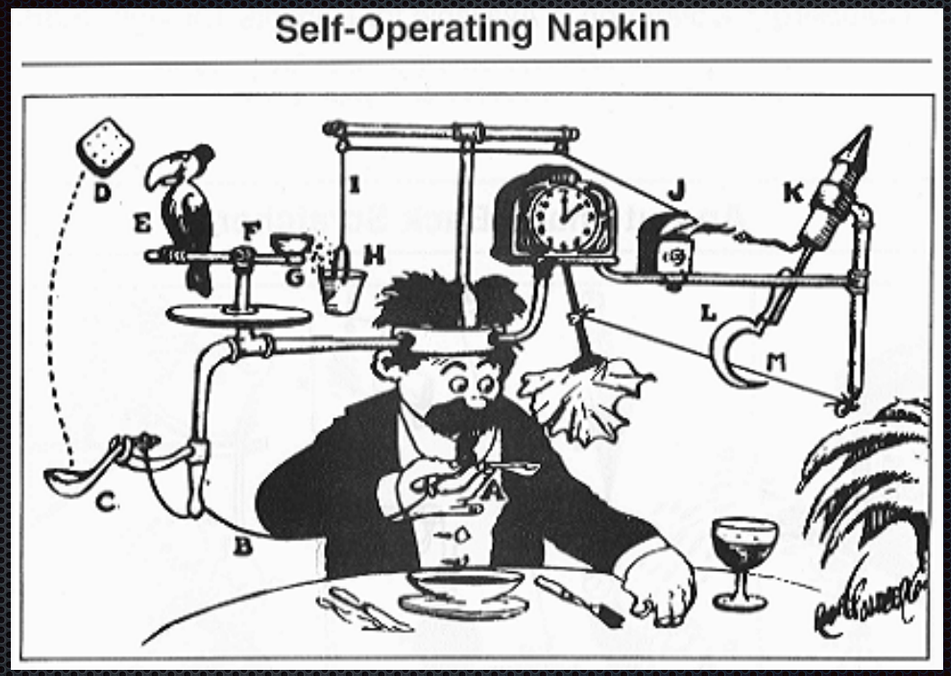
[Photo: © User:Colin / Wikimedia Commons / CC-BY-SA-4.0]

http://www.fastcompany.com/3054875/elasticity/instagram-hack-reveals-the-risks-of-bug-bounty-programs

thinkst applied research

- Late last year, lots of questions were raised ( and opinions voiced), over controversy involving Facebook's bug bounty program.
- Wesley Weinberg pushed the bounty's rules and managed to gain access to essentially everything in Instagram, including all your photos and messages.
- When he wanted to publicise the findings and claim his million-dollar-bug-reward, Facebook cited violations to the rules and drama ensued.
- Disregarding the politics, there were huge security flaws exposed in Amazon Web Services. Specifically, security around s3 buckets.


- Now Facebook is a big company
- The are exceptionally  smart security wise, they have a large security team and smart people working in it.
- Yet, there was an attacker browsing their s3 buckets  and they didn't know.
- If the buckets could be tokened, they would've known before any sensitive data was stolen.
- Essentially, we are looking at tokening files and folders for the cloud.
- But how do we do this?



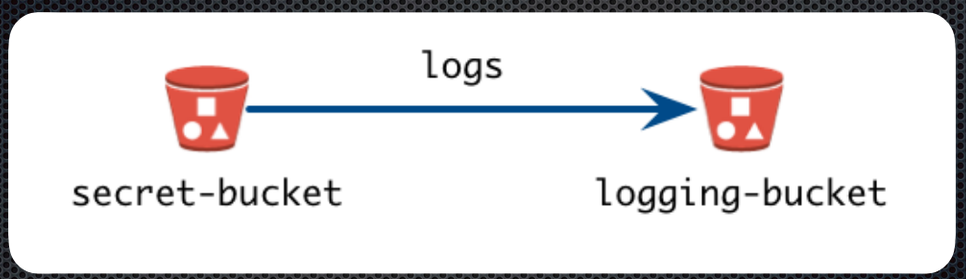

https://upload.wikimedia.org/wikipedia/commons/a/a9/Rube_Goldberg%27s_%22Self-Operating_Napkin%22_%28cropped%29.gif

thinkst applied research

- So to make this happen we are going to build a Rube-Goldberg machine.
- For those of you who haven’t witnessed the fascinating simplicity of the complex designs, I'd suggest you google and enjoy.
- Basically, a  Rube-Goldberg machine takes a simple task and deliberately performs it in a complex way, usually using chain reactions.


- You don't need to worry though, our tokening isn't that complicated, we focused more on the chain reaction idea.
- From your side it is easy, we provide a script, prompt you for a few inputs, and the rest is taken care of.

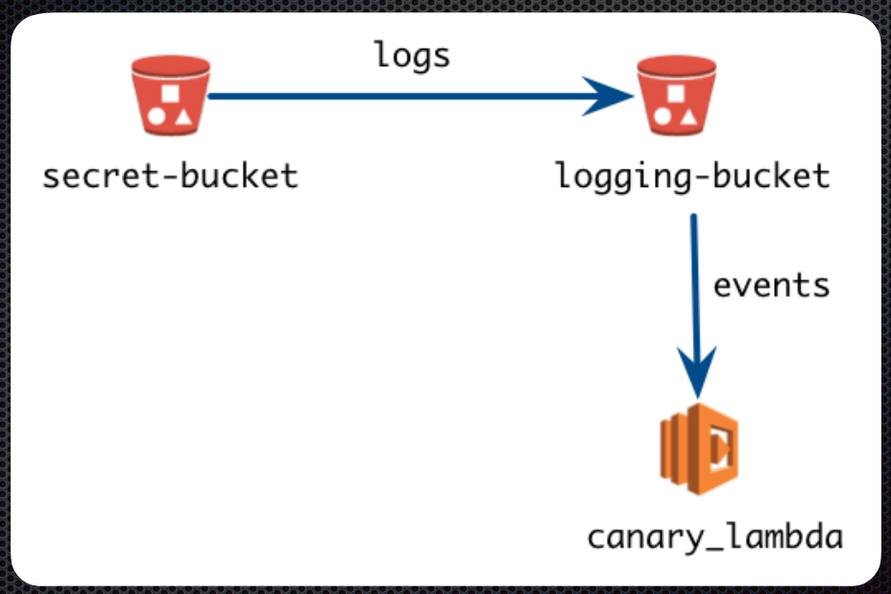
# Server Access Logging




thinkst
applied research

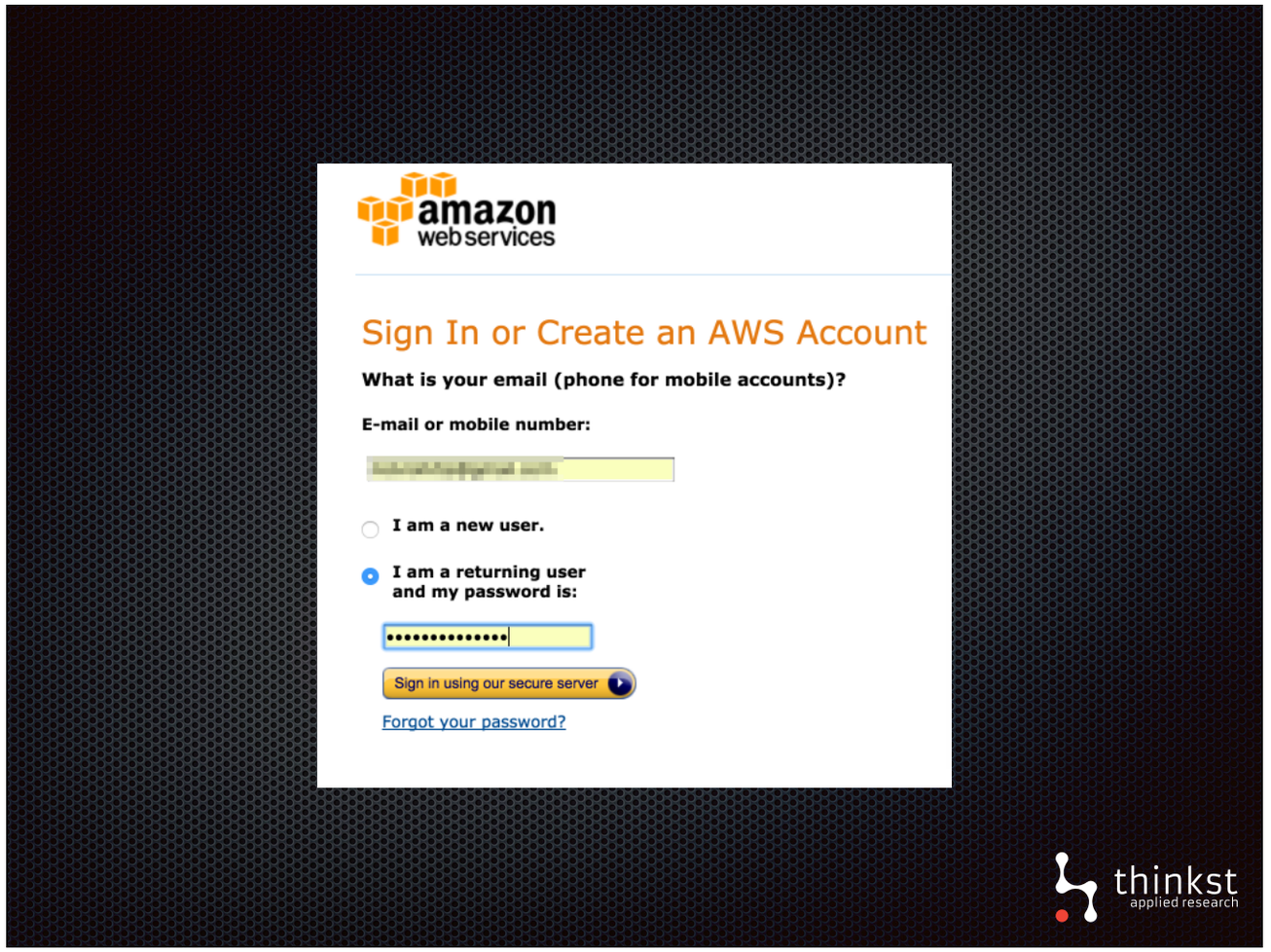
- Back to our S3, let's say we have secret_bucket that no one should be accessing.
- Our idea then is to monitor access to our secret_bucket and when it is accessed, get notified.
- In order to make this happen we need to create a second bucket, lets call it logging_bucket.
- We then enable logging on our secret_bucket and point the logs to our logging_bucket.
- Now, whenever someone accesses secret_bucket, logs are created in logging_bucket.
- We then monitor logging_bucket for any changes, if there is, we know someone is accessing secret_bucket.
- We must mention that AWS architecture dictates that logs arrive on a best effort basis meaning they sometimes take a few minutes and sometimes take an hour or more to arrive.  However a couple hours is still better than weeks/months later via some public announcement

# Server Access Logging

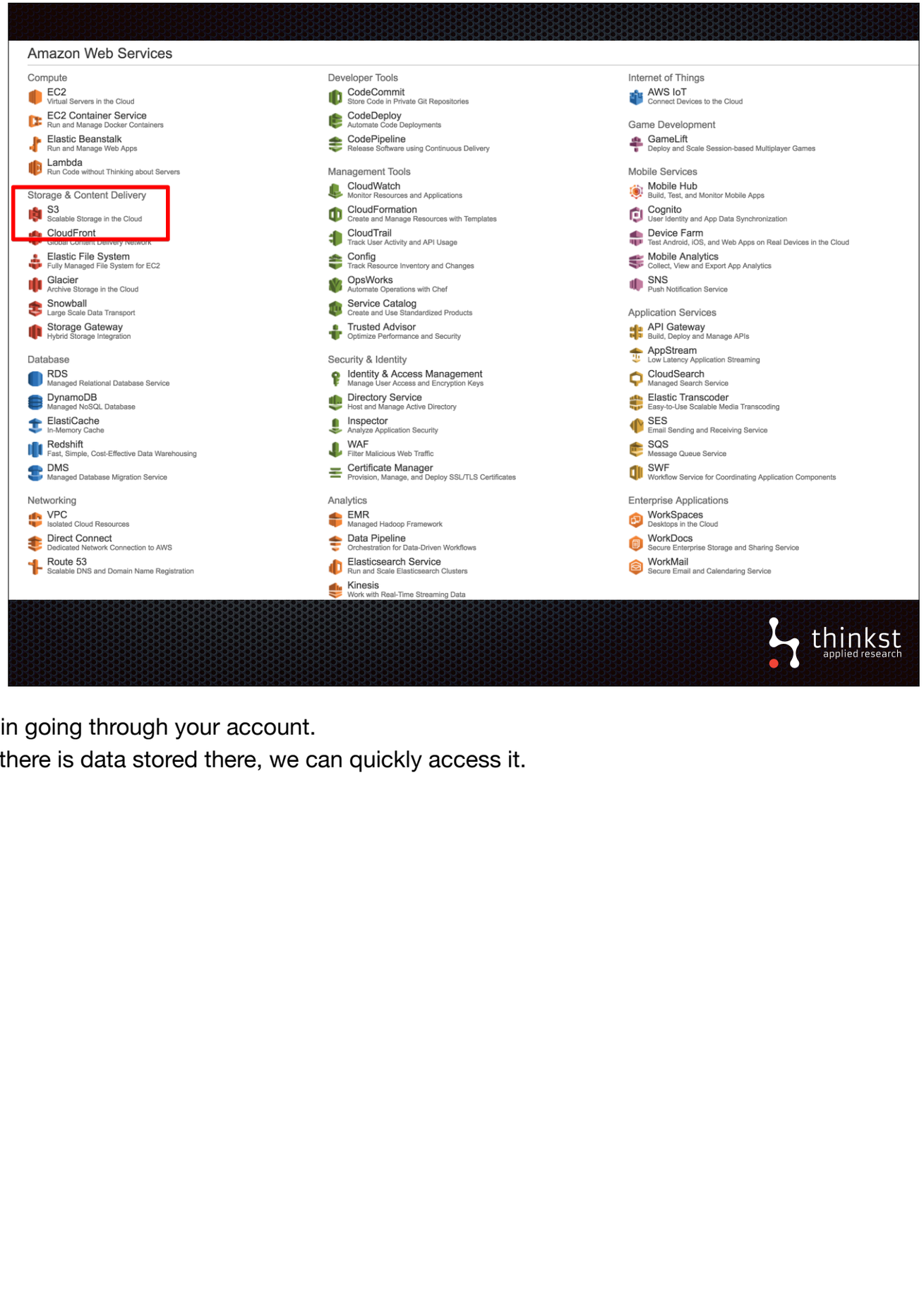



thinkst
applied research

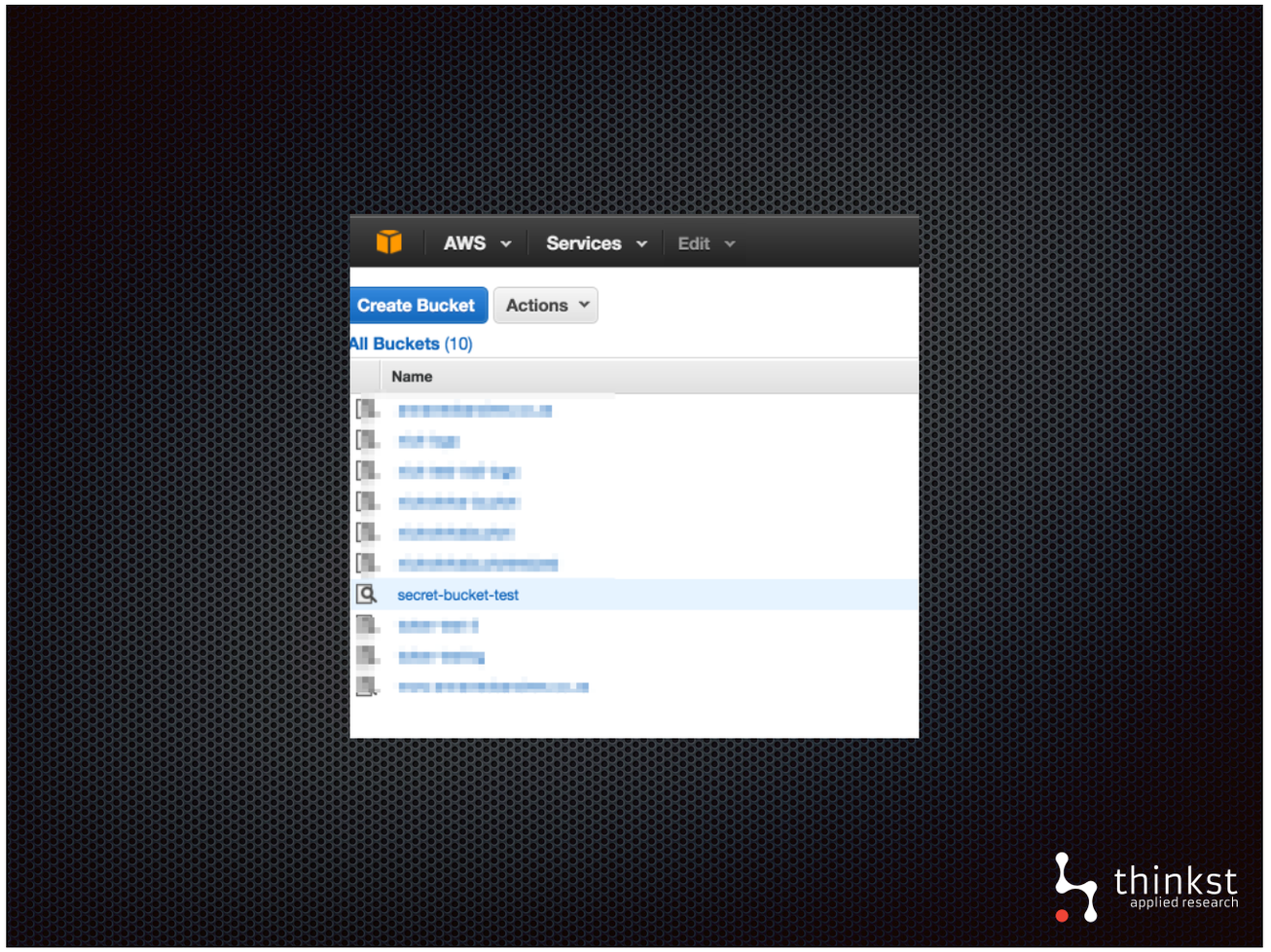
- So, as soon as there is a change in logging_bucket, an event is triggered which fires off an AWS computing process called Lambda - if you haven't seen AWS lambda, you should take a look, it is really cool. Basically, it lets you run your js/java/python code in the cloud whenever an even triggers.
- The lambda function parses the log entry and sends a request to Canarytokens.  This in turn sends you an alert.
- You could use Amazon's simple notifications service here if you wanted to, but it means you need to set it up yourself and manage it.
- Our S3 token takes care of that for you as well as condensing multiple logs into a single event meaning you don't get flooded with notifications.
- I'll now walk you through the process of triggering the S3 token.



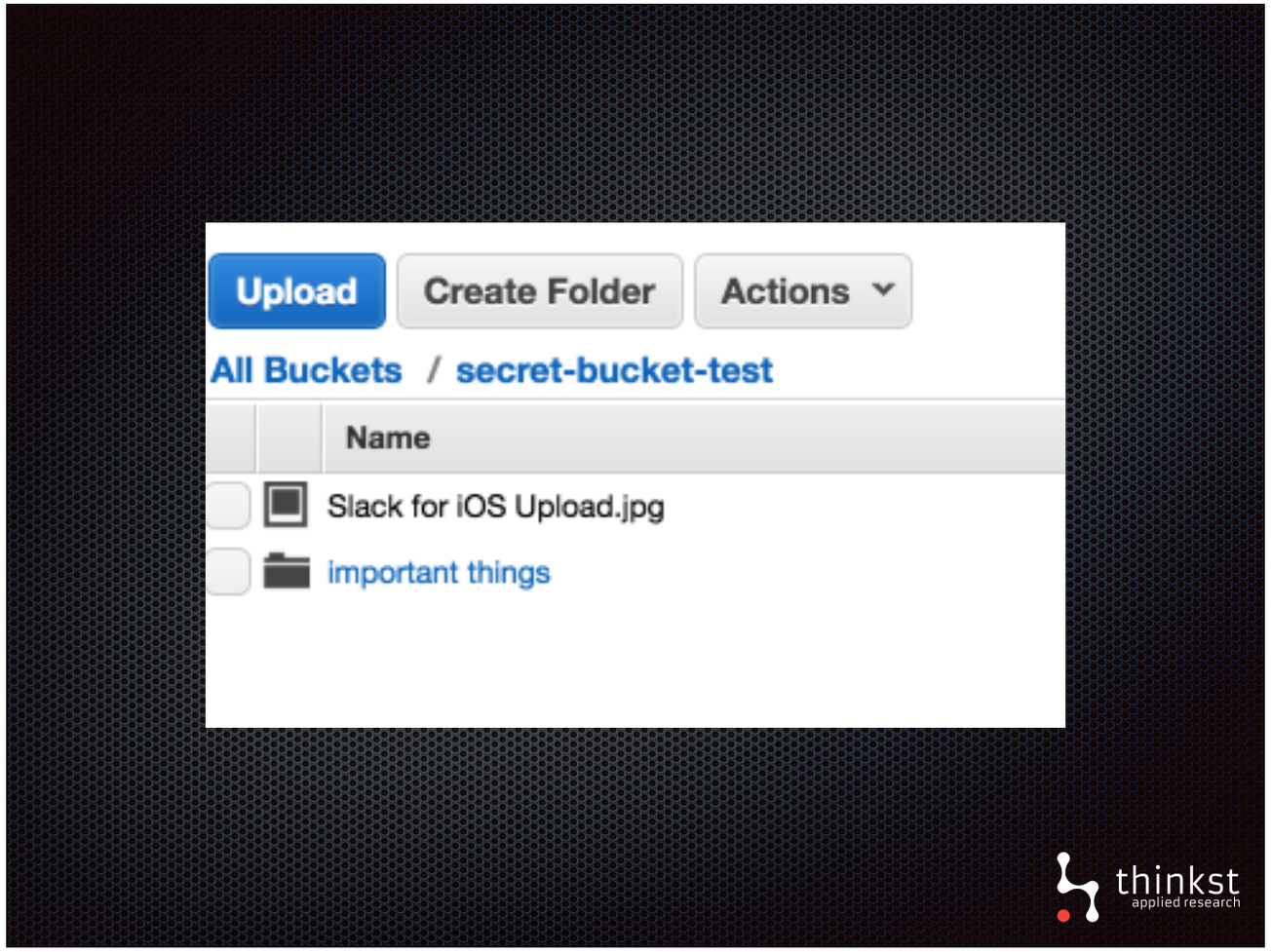

- It starts with someone gaining access to credentials and logging into AWS.



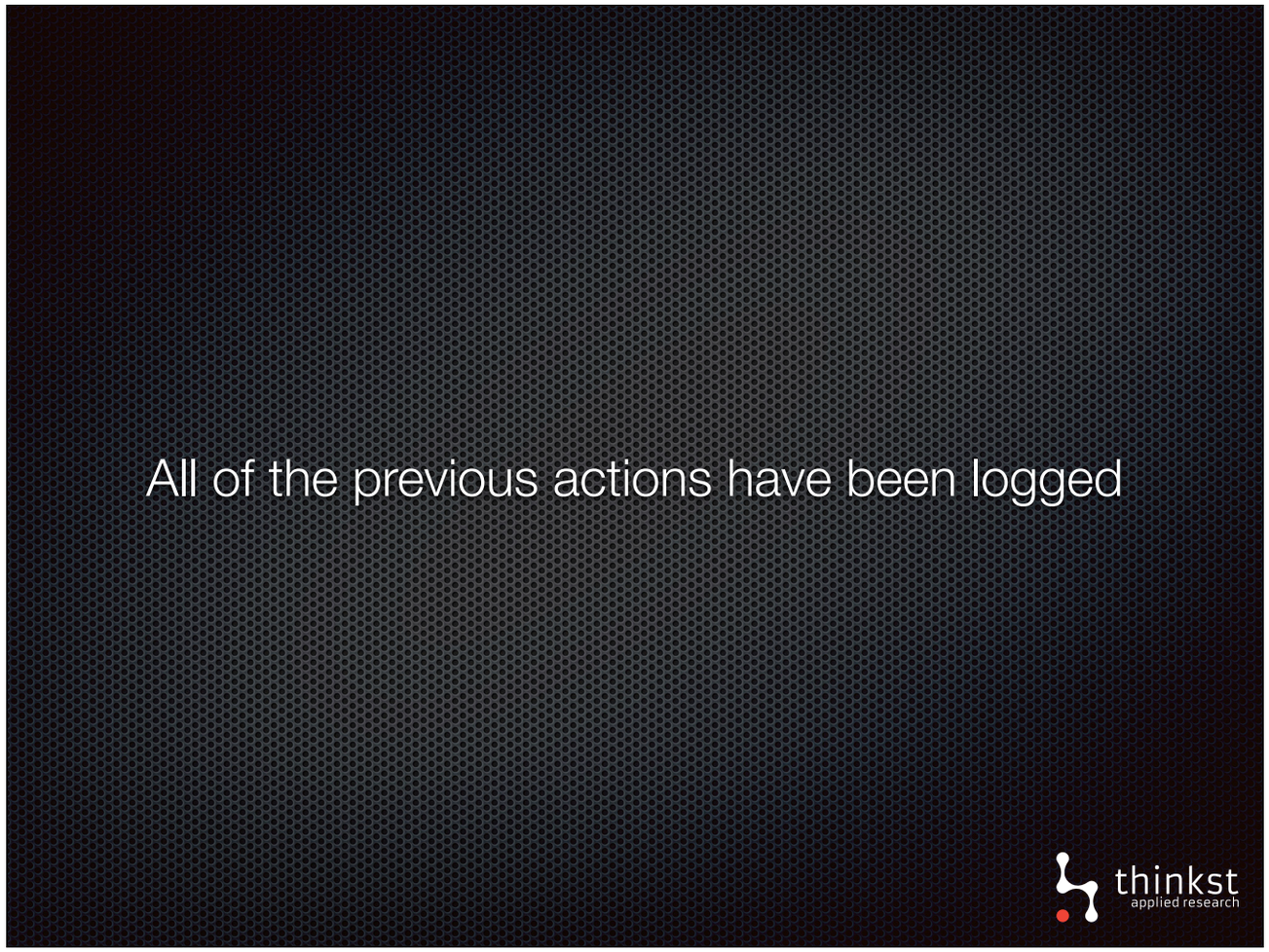

- Once logged in, the attacker will begin going through your account.
- S3 is a good place to start because if there is data stored there, we can quickly access it.



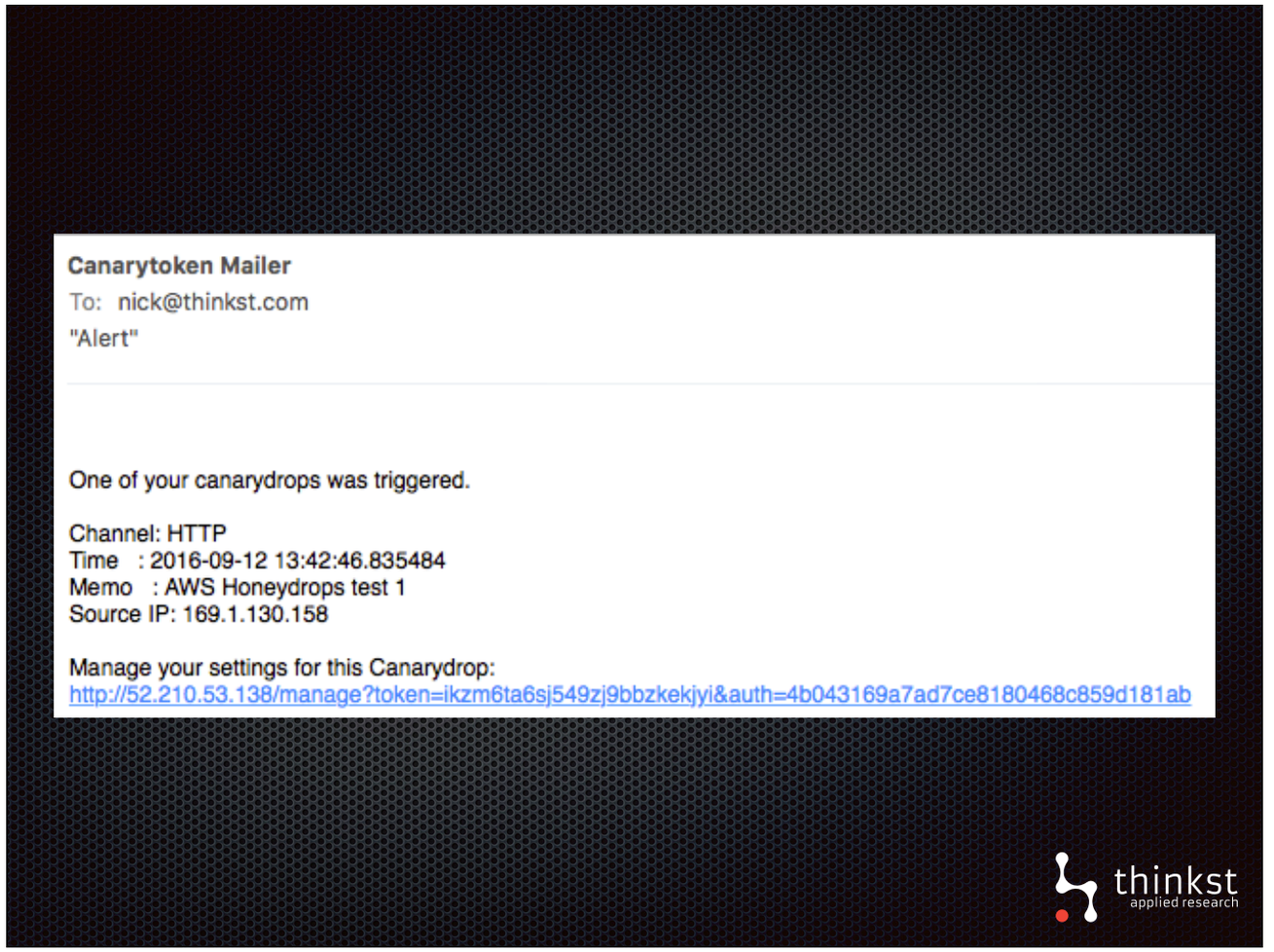

- Here we can see a list of all the buckets available.  Particularly, the secret-bucket looks interesting, let's see what is in it.




- Straight away I can see there is a folder and an image.  It is important to note that at this point already, the access to the bucket is logged, so we will be receiving an alert on this event even if the attacker does nothing further.
- Let's say the attacker decides to open the image, or browse the folder and view the contents in there, going through all our private documents, eventually having enough and moving on.

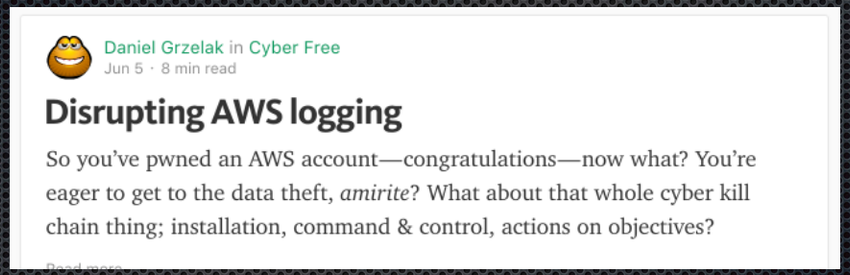



- At this point we have captured all of the actions of the attacker.  Starting from accessing the bucket to opening the image or browsing the folder and opening files inside it, we know what he/she was doing.
- AWS now will provide the logs which will get created in our logging bucket.  This may take up to an hour but once they come, our function will run and an alert will be generated.




- From this alert, you can go into your logging bucket and further examine what exactly happened.
- Either way, you know that credentials have been compromised (or you just have a nosey employee snooping around).

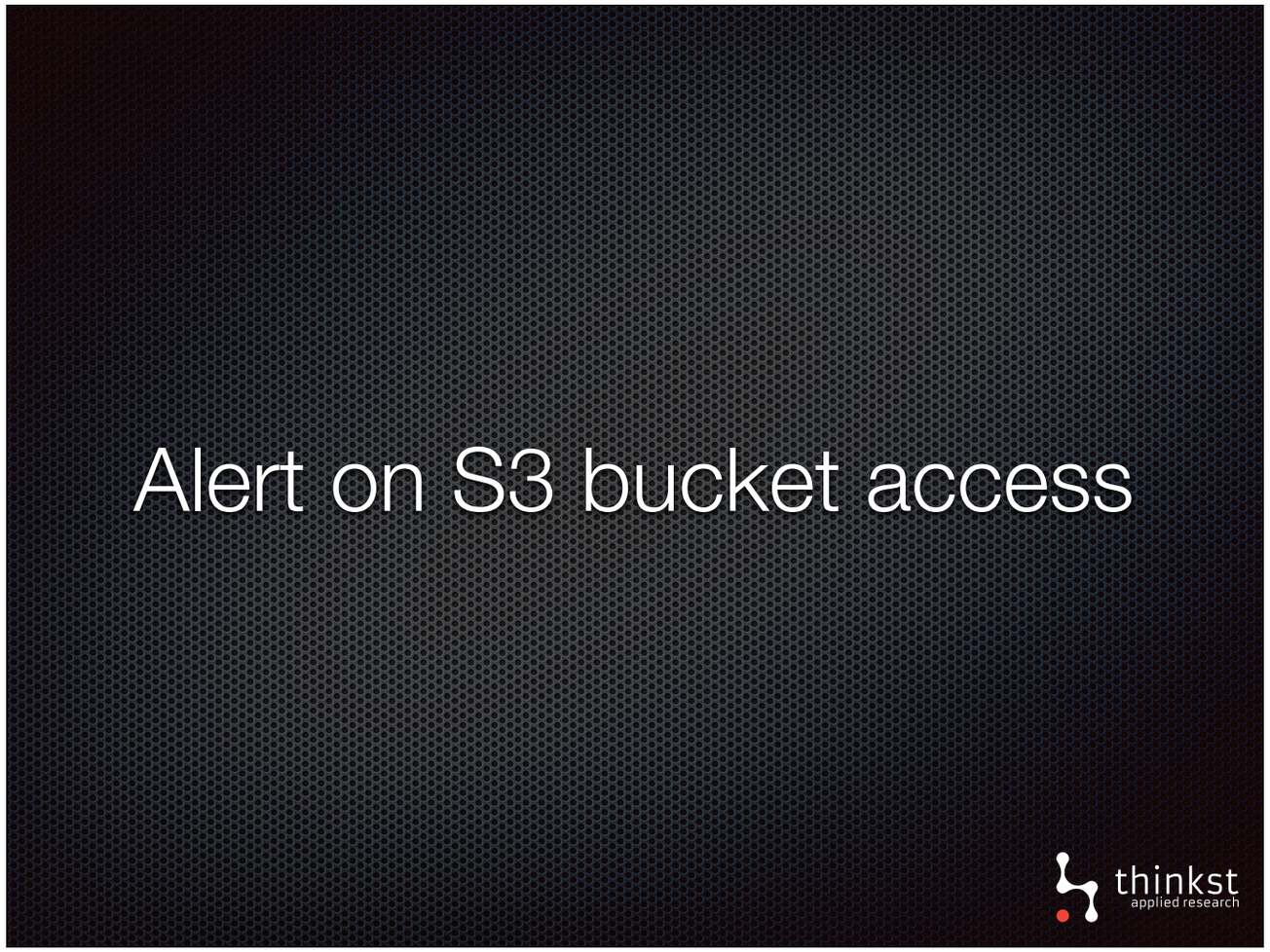
# AWS Cloudtrail




https://danielgrzelak.com/disrupting-aws-logging-a42e437d6594#.kx8xrp5nq

thinkst applied research

- As mentioned earlier, tokens aren't a replacement for logging.
- AWS Cloudtrail is a service that logs AWS API calls for your account.
- Logging API calls is extremely important in detecting unwanted access to your AWS account.  We cannot emphasise enough the importance in maintaining and monitoring your AWS logs.
- In fact, Daniel Grzelak recently wrote a few blog posts talking about owning an aws account, and the first thing he mentions is disabling logging as it is a dead give-away if not.




- Coming back to the Instagram hack. They had someone snooping around their account and viewing sensitive data in their S3 buckets without ever knowing.
- If they were alerted on the access, they could have stopped the attacker moving around.
- Our S3 token alerts you on access to a bucket. Put this on a fake bucket or a bucket that no one should be accessing, and any alerts will be important alerts.

# Physical breaches


- We've talked a lot about software tokens and how to determine if someone is moving around your network. But, what happens when someone is physically in a location they shouldn't be? What happens if you have a physical breach?
- As an example, your CEO is crossing the border and his phone is seized  or alternatively, you have an off-limits data centre, how can you determine if someone is inside?
- QR codes are our stab at tokening physical objects or locations

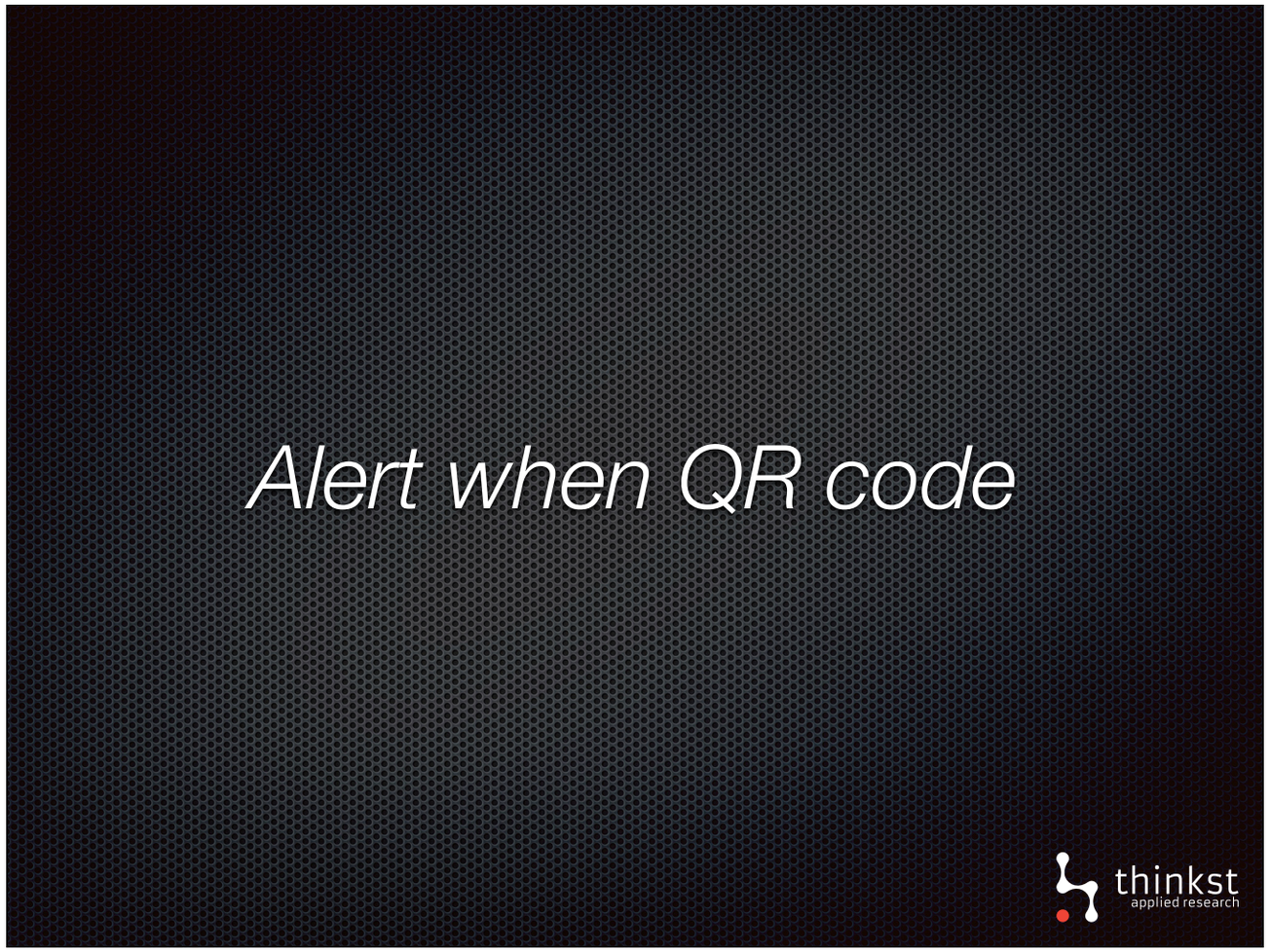

http://www.macworld.com/article/2059745/apple-promises-fix-for-keyboard-trackpad-woes-on-13-inch-retina-macbook-pro.html

thinkst applied research

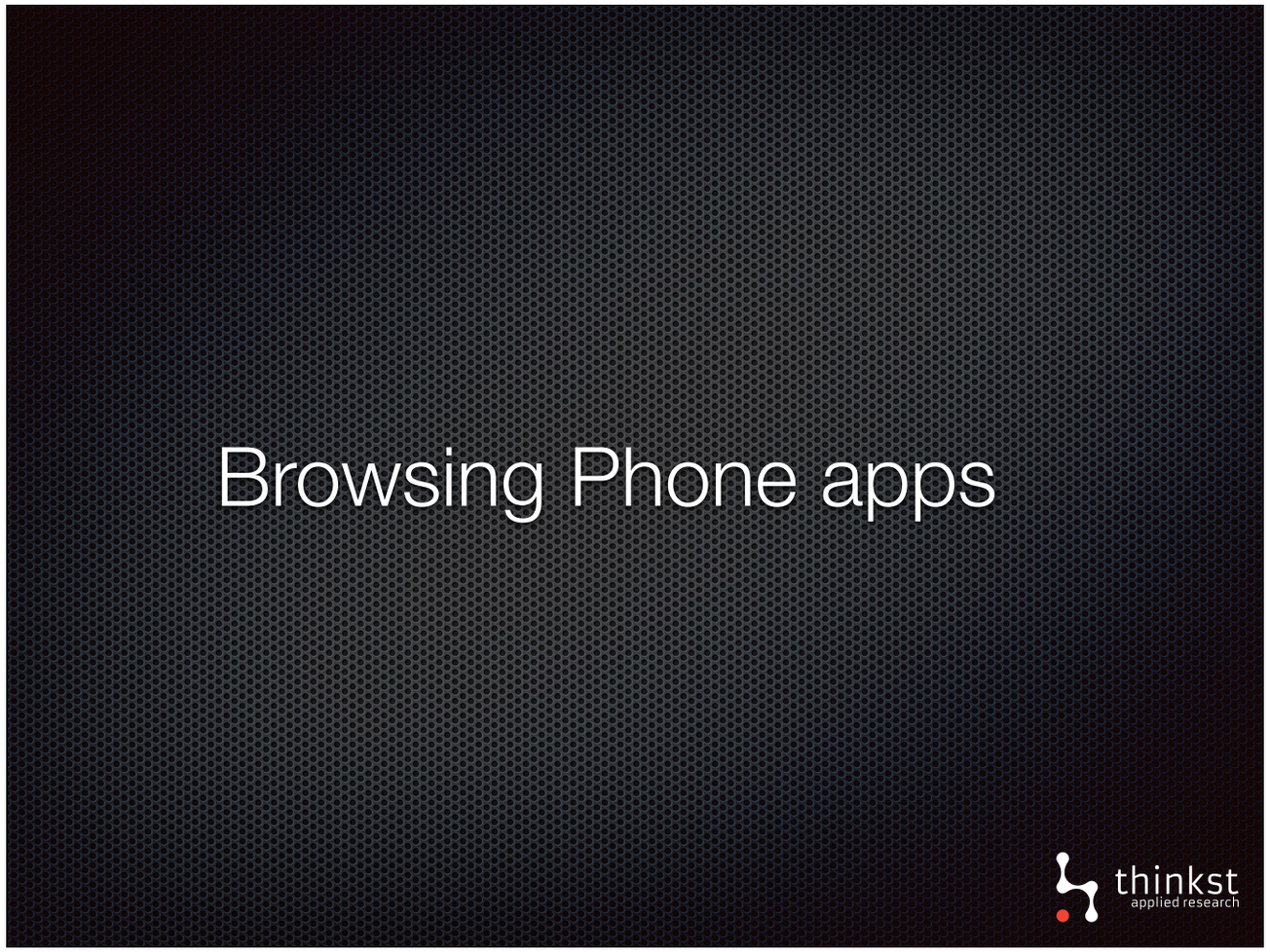
- You put a QR code on the back of the battery, if someone seizes the phone at a border and dismantles it, they will see the code.
- Or you put one on the wall of your datacenter or for any sort of asset tracking, if someone scans it, they will hit our tokened URL meaning you then get an alert which would indicate that someone is somewhere they shouldn't be.

# Demo: QR code

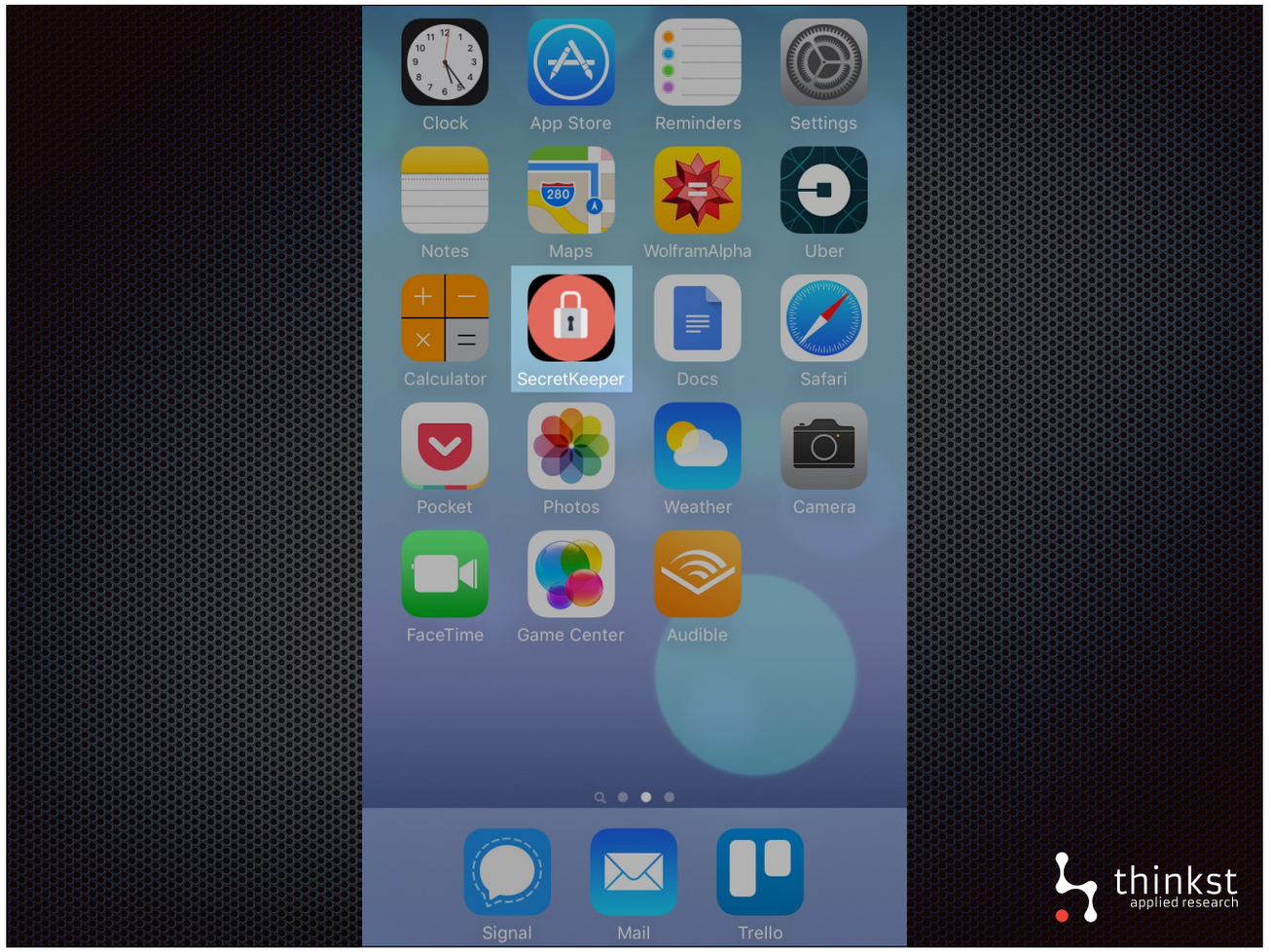

- Quick demonstration of triggering QR Code Token.




- Breaches of physical locations can be as important as network breaches.  Knowing when someone is somewhere they shouldn't be is critical to security.
- QR codes let you leave physical tokens lying around and alert you if ever scanned, showing you someone unfamiliar with the territory is exploring it.




- If instead looking at the phone battery, we turn the phone around and look at the apps in the front:  there's plenty of sensitive things to be had like emails and 2FA.
- If a phone is left lying around unlocked, or the owner forced to unlock it, someone could have a chance to go through it's apps.
- If for example this is a c-level exec's phone or sysadmins phone it could be useful for others to know, that someone snooping through to get at emails or 2FA details.




- One of colleagues Jason, built an iOS app called secret keeper.  It purports to store all of manner of secrets. But in actual fact, it's a tokened app that fires an alert whenever opened.
- With this proof of concept app, the work is already done and hooked up to our alerting. It only takes a few minutes to install and configure. It's not in the app store, but you get the source of the app and install it quite easily.

# Demo: tripping SecretKeeper App


- Quick demonstration of triggering iOS Tokened App.

**Date:** 2016 Sep 14 14:13:15 **IP:** None **Channel:** HTTP **Country:** Unknown

| Tor | |
|---|---|
| Known Exit Node | False |

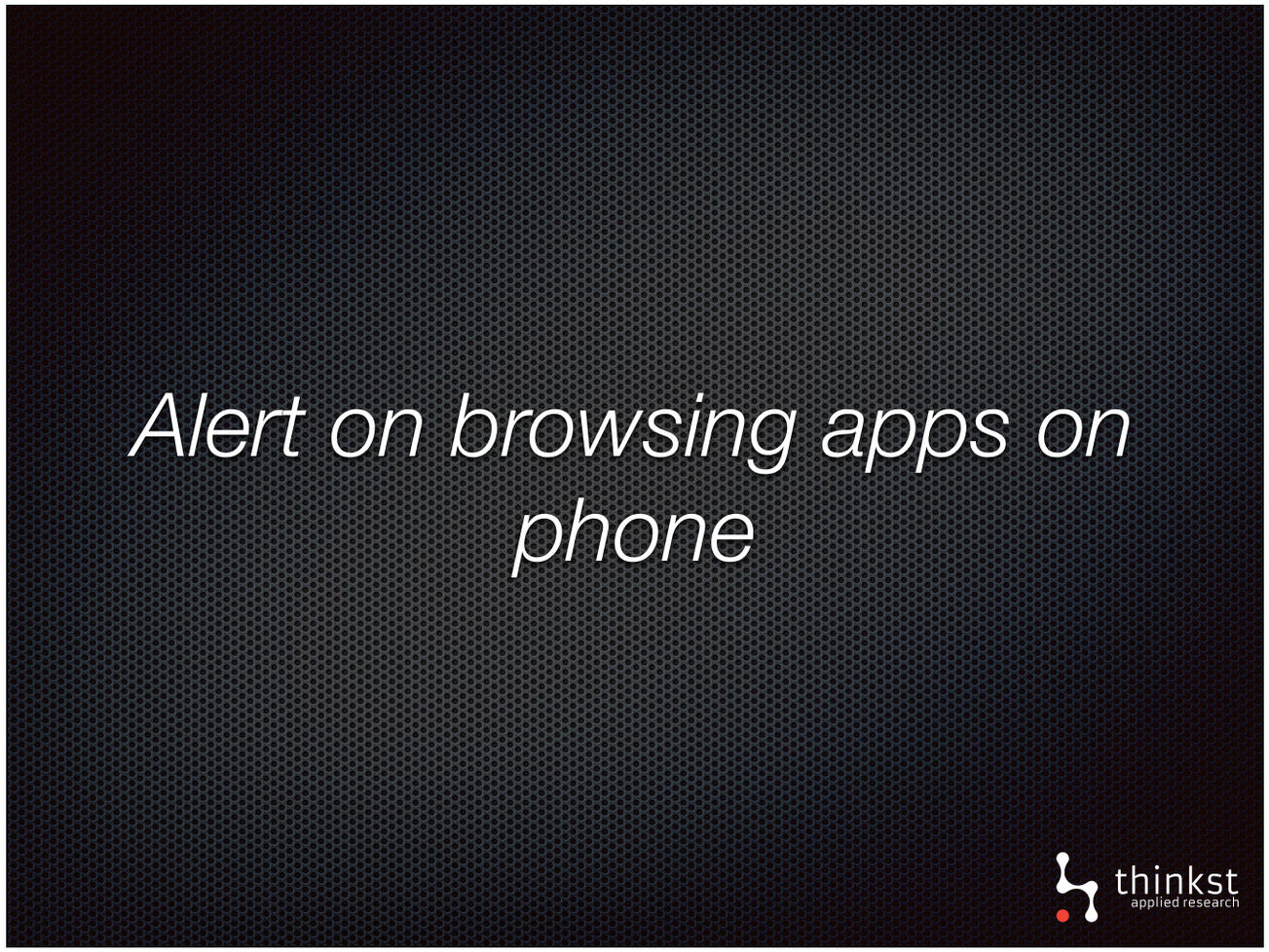
| Basic Info | |
|---|---|
| useragent | SecretKeeper/1 CFNetwork/758.5.3 Darwin/15.6.0 |

**Additional Info**

| secretkeeper_photo | |
|---|---|
| Photo |  |
| **iOS-App** | |
| loc | -33.9278254874291,18.4371831317379 |
| address_info | [redacted] Cape Town WC 8001 South Africa |



- The app also sends through a photo of the whoever opened and the location.




- To recap, what we just saw there - by installing SecretKeeper, when someone snooping through apps looking for info, we'll be alerted to it.


- We've spoken about applied tokens and how we can use them in different settings
- Now we want to show you how we are improving the information you can get out of them tokens, or use tokens to get more info out of somewhere.

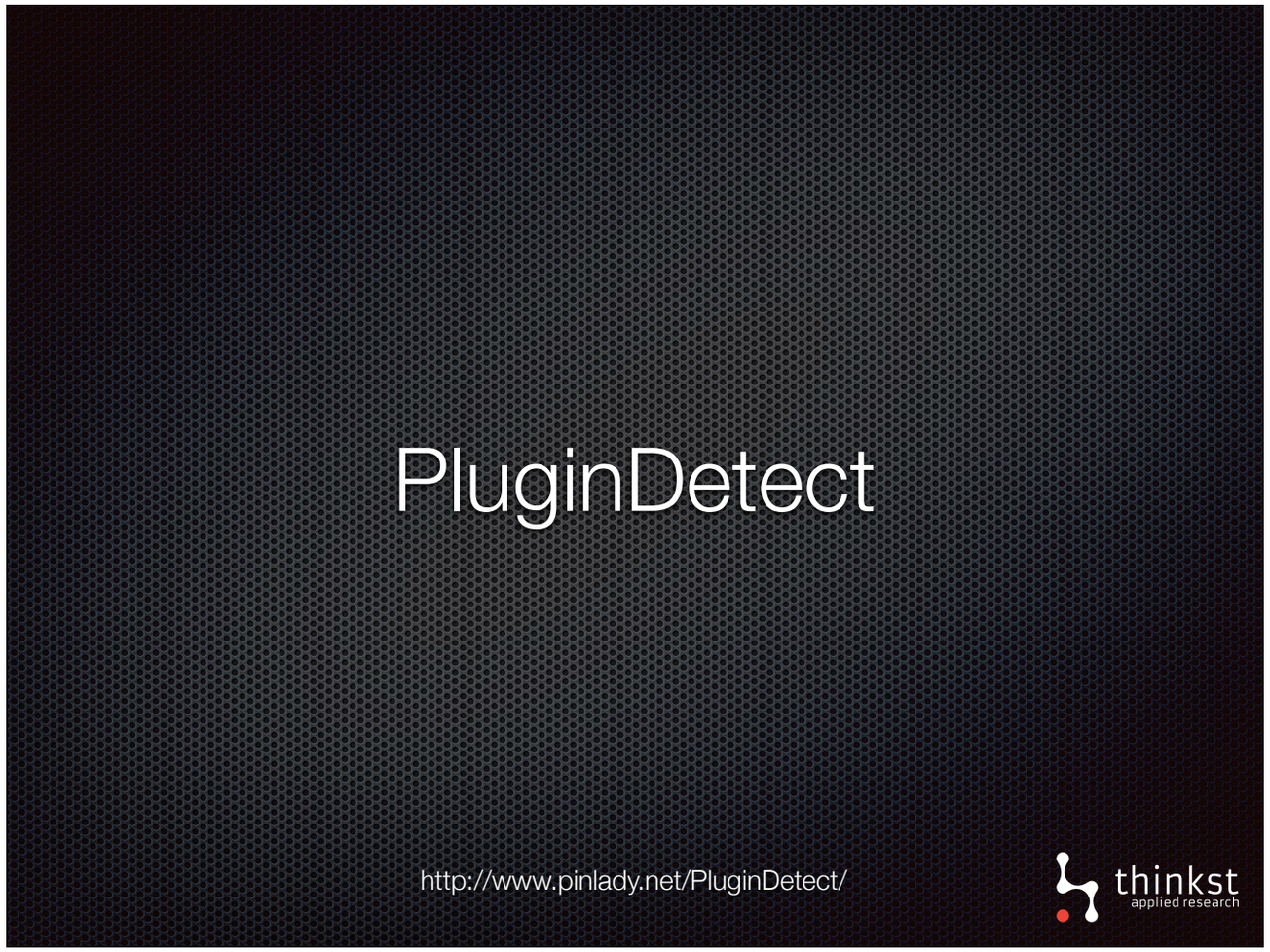
# De-anonymization

- In an attempt to help better understand the threat, we have included a few techniques that try to de-anonymise the attacker.  If they work, we show the extra information but they won't always work and this could simply  be because the attacker disables access to the information.
- But, when they do work, we can learn a little more about the incident which can help narrow down where it originated from.

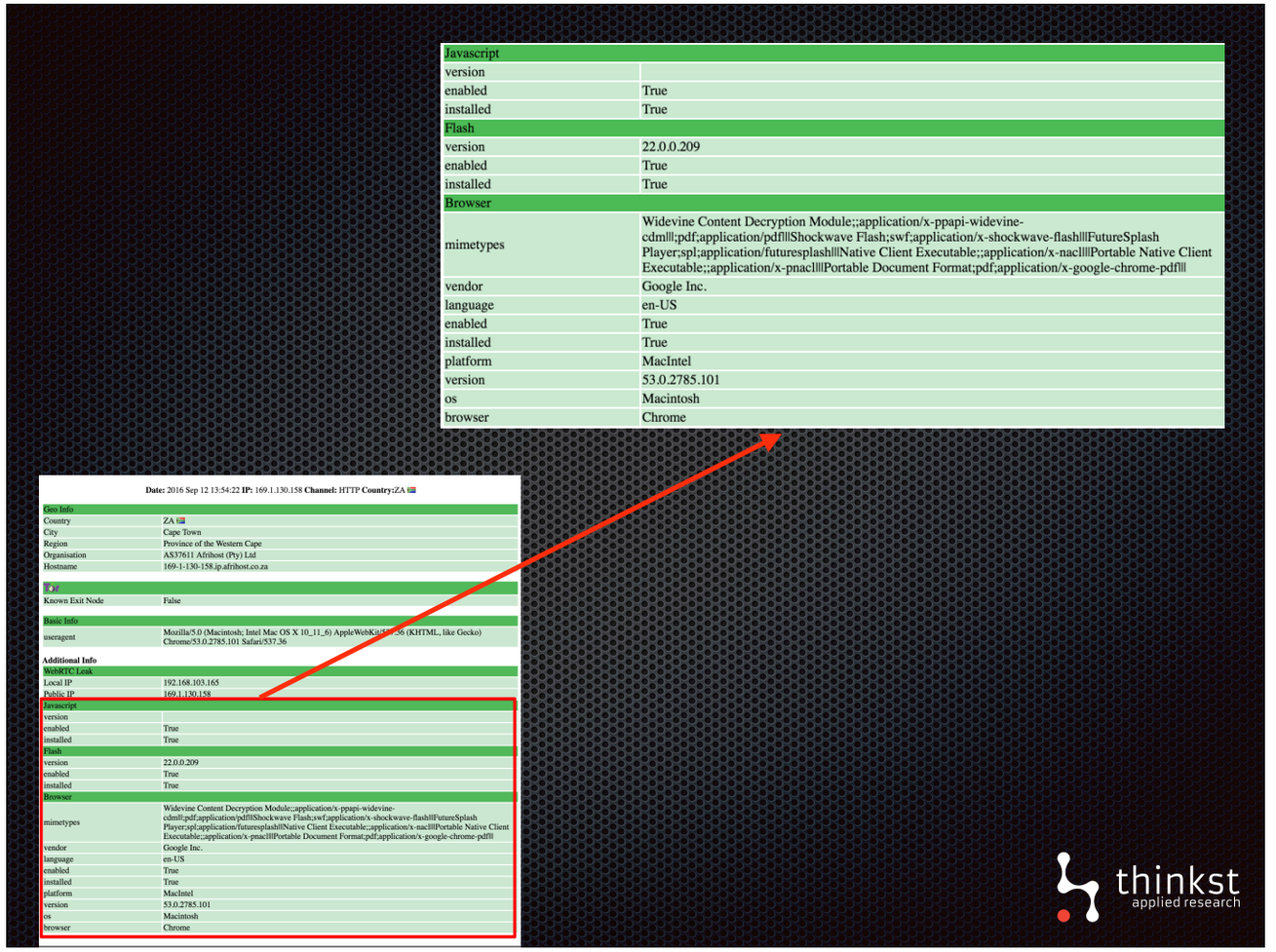
# Browser Scanner

thinkst
applied research

- We've extended the web bug to include details about the host browser.
- What this means is that you are able to grab some more details about who is tripping your token.




- This is achieved using PluginDetect which is an available javascript library.
- The  javascript enumerates plugins and sends it back to us.
- This extra information helps to get a sense of who triggered the alert and supplies audit info if you want to do forensics.



**Date:** 2016 Sep 12 13:54:22 **IP:** 169.1.130.158 **Channel:** HTTP **Country:**ZA

| Geo Info | |
|---|---|
| Country | ZA |
| City | Cape Town |
| Region | Province of the Western Cape |
| Organisation | AS37611 Afrihost (Pty) Ltd |
| Hostname | 169-1-130-158.ip.afrihost.co.za |

| Tor | |
|---|---|
| Known Exit Node | False |

| Basic Info | |
|---|---|
| useragent | Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.101 Safari/537.36 |

**Additional Info**

| WebRTC Leak | |
|---|---|
| Local IP | 192.168.103.165 |
| Public IP | 169.1.130.158 |

| Javascript | |
|---|---|
| version | |
| enabled | True |
| installed | True |

| Flash | |
|---|---|
| version | 22.0.0.209 |
| enabled | True |
| installed | True |

| Browser | |
|---|---|
| mimetypes | Widevine Content Decryption Module;;application/x-ppapi-widevine-cdm\|\|;pdf;application/pdf\|\|Shockwave Flash;swf;application/x-shockwave-flash\|\|FutureSplash Player;spl;application/futuresplash\|\|Native Client Executable;;application/x-nacl\|\|Portable Native Client Executable;;application/x-pnacl\|\|Portable Document Format;pdf;application/x-google-chrome-pdf\|\| |
| vendor | Google Inc. |
| language | en-US |
| enabled | True |
| installed | True |
| platform | MacIntel |
| version | 53.0.2785.101 |
| os | Macintosh |
| browser | Chrome |

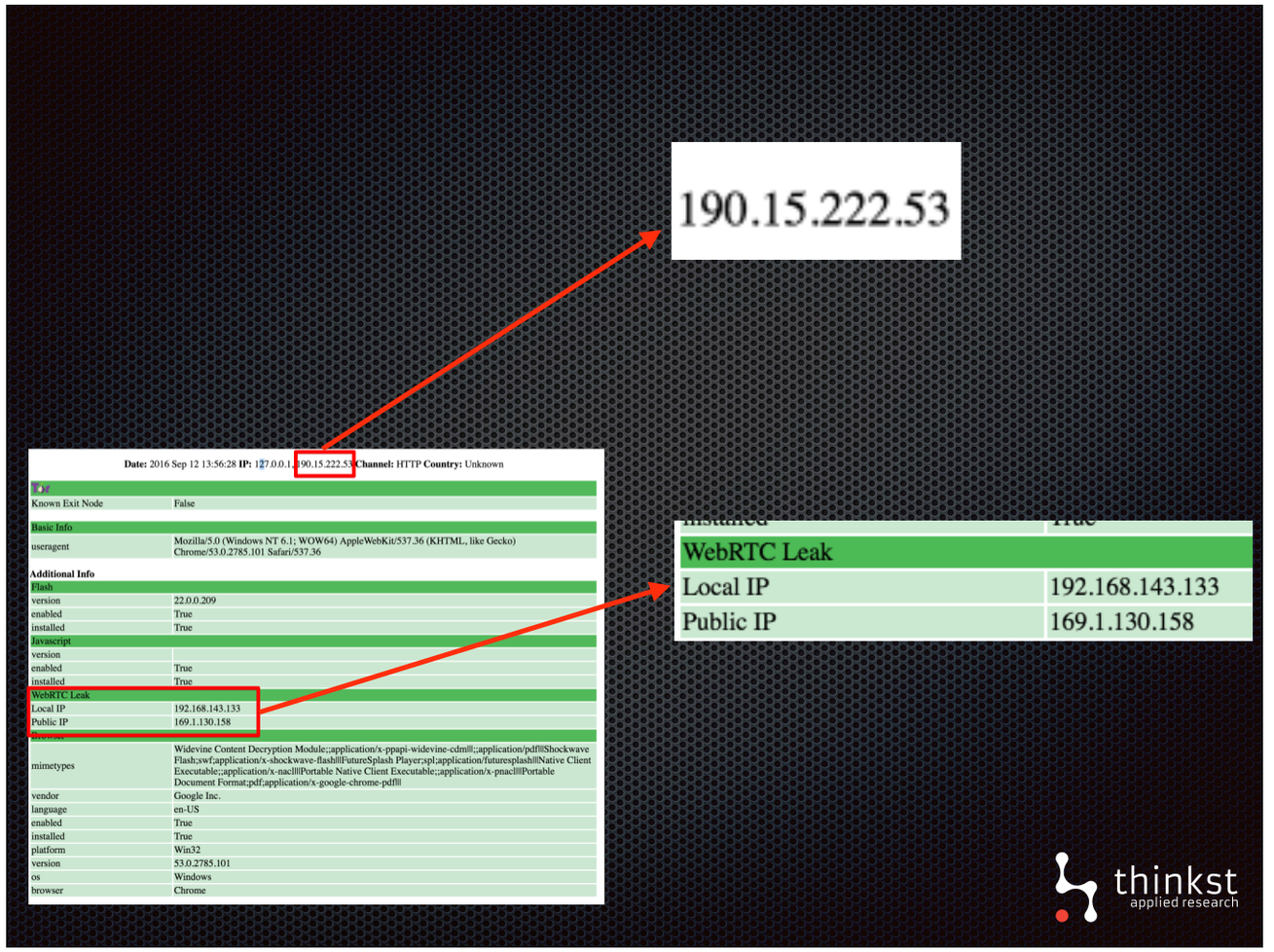
- Here we can see additional information about incident.
- We can see flash and js is installed and enabled, as well as specific browser information.

# WebRTC

https://github.com/diafygi/webrtc-ips

thinkst
applied research

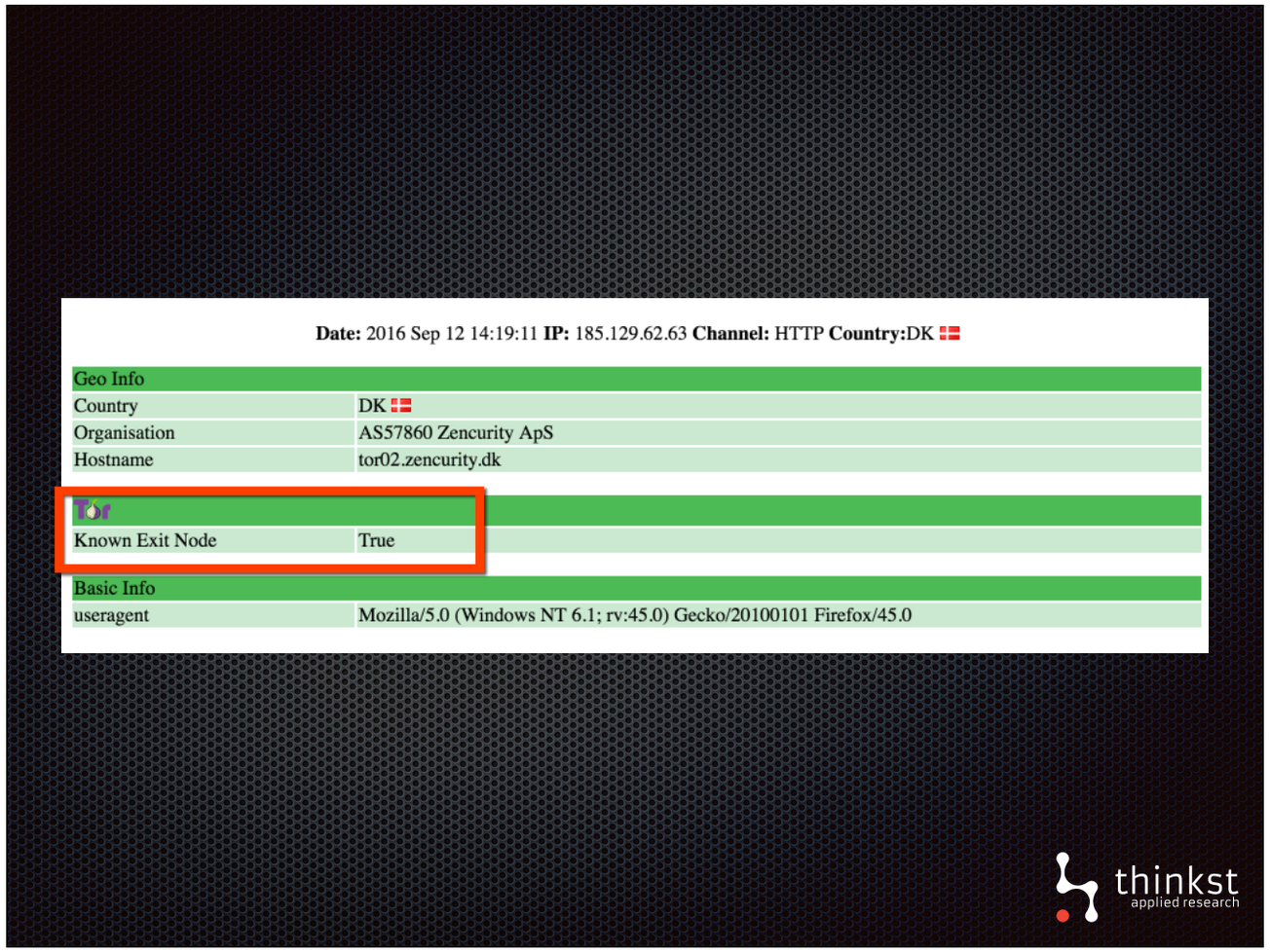
- WebRTC is an open framework for real time web  peer to peer communication
- Using a trick exposed by Daniel Roesler  we can use WebRTC APIs to expose local and public IP even if behind a VPN/proxy by making a javascript request to a STUN server.
- These stun servers are  used between peers behind a NAT to share public IPs so they can directly communicate.
- On windows using chrome or firefox, if WebRTC is enabled, when we load our web bug page, we make the calls and attempt to get the IPs. This won't always work, but if it does, it provides us with some extra usable information.




- Here we can see our IP was seen as 190.x as we were behind a proxy, but the WebRTC leak details show our provider assigned public IP as 169.x
- In the past, we have seen other attempts at obtaining a users local and public IP. These have included using flash or javascript to get the information. WebRTC is that current flavour and already has begun to be shutdown. When the next trick comes along, we'll add that to our scanner.
- It's all about getting every bit of information available to us to help you understand the threat that triggered your alert.

# Tor Relay

- Another quick check we can perform let us know if the attacker is routing through TOR.
- This is easily done: we check if the source IP is in the list of known public Tor exit nodes.




- As can be seen here, the IP has been identified as a public tor exit node.

# Token history

- Previously, hits on a token were isolated, meaning we didn't have an easy historic view of times the token was triggered.
- We've now added history to tokens, enabling you to view a list of previous hits.
- Now why would that be useful?
- Let's say you normally get admin config page hits from somewhere expected, your office for example. All of sudden you get a hit that comes from some external location.
- In the past it would be difficult to see the relationship between these hits.  Now you can quickly see that this is an outlier and needs immediate action.

**Map:**


**History:**

**Date:** 2016 Sep 15 10:23:32 **IP:** 194.42.227.53 **Channel:** HTTP **Country:**GB

Expand

**Date:** 2016 Sep 15 09:28:10 **IP:** 194.42.227.56 **Channel:** HTTP **Country:**GB

Expand

**Date:** 2016 Sep 15 08:36:57 **IP:** 197.214.117.130 **Channel:** HTTP **Country:**ZA

Expand

**Date:** 2016 Sep 15 08:33:02 **IP:** 10.243.58.30 **Channel:** HTTP **Country:** Unknown

Expand

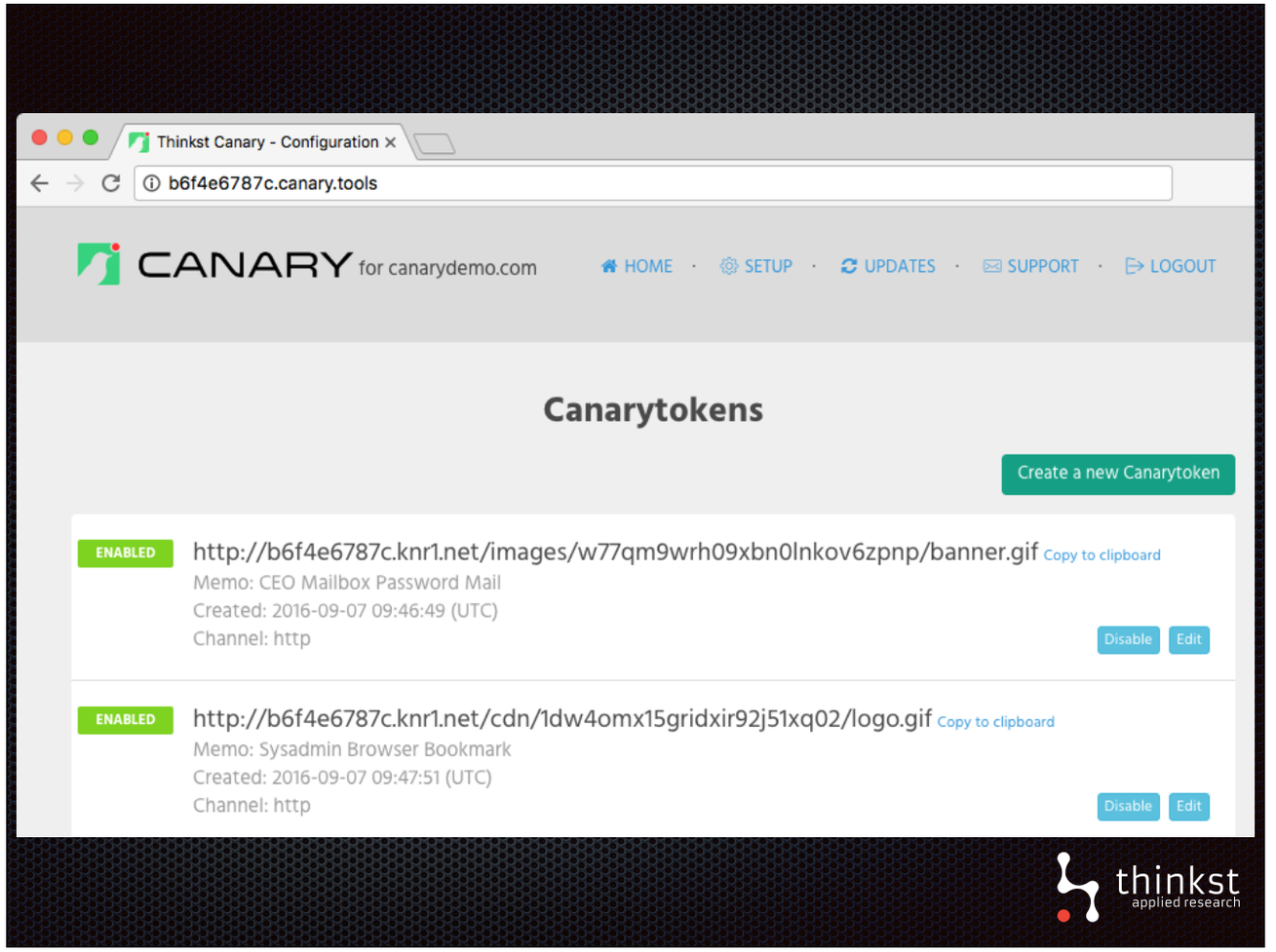
**Date:** 2016 Sep 15 00:06:29 **IP:** 169.0.76.252 **Channel:** HTTP **Country:**ZA

Expand

thinkst
applied research

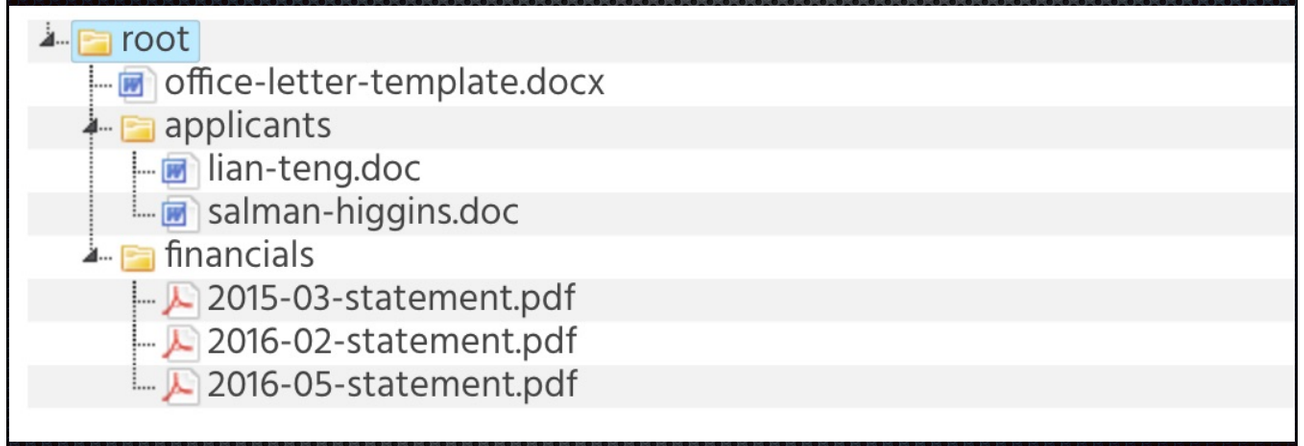
- As can be seen we have also added a map which lets you get a sense of the attacks at a quick glance.
- You can also view a list of the previous hits as well as details about the hits.
- Possibly giving you a better chance to deal with them.
- Our whole idea here is to give you more information. The more information you have, the better chance there is of resolution to your problem.


- Another way in which tokens can add information is when used with honeypots.
- Even though tokens have lives of their own scattered among real systems, they can still be used with honeypots like those early researchers and sysadmins did. we set about integrating canary tokens with a commercial honeypot we build.



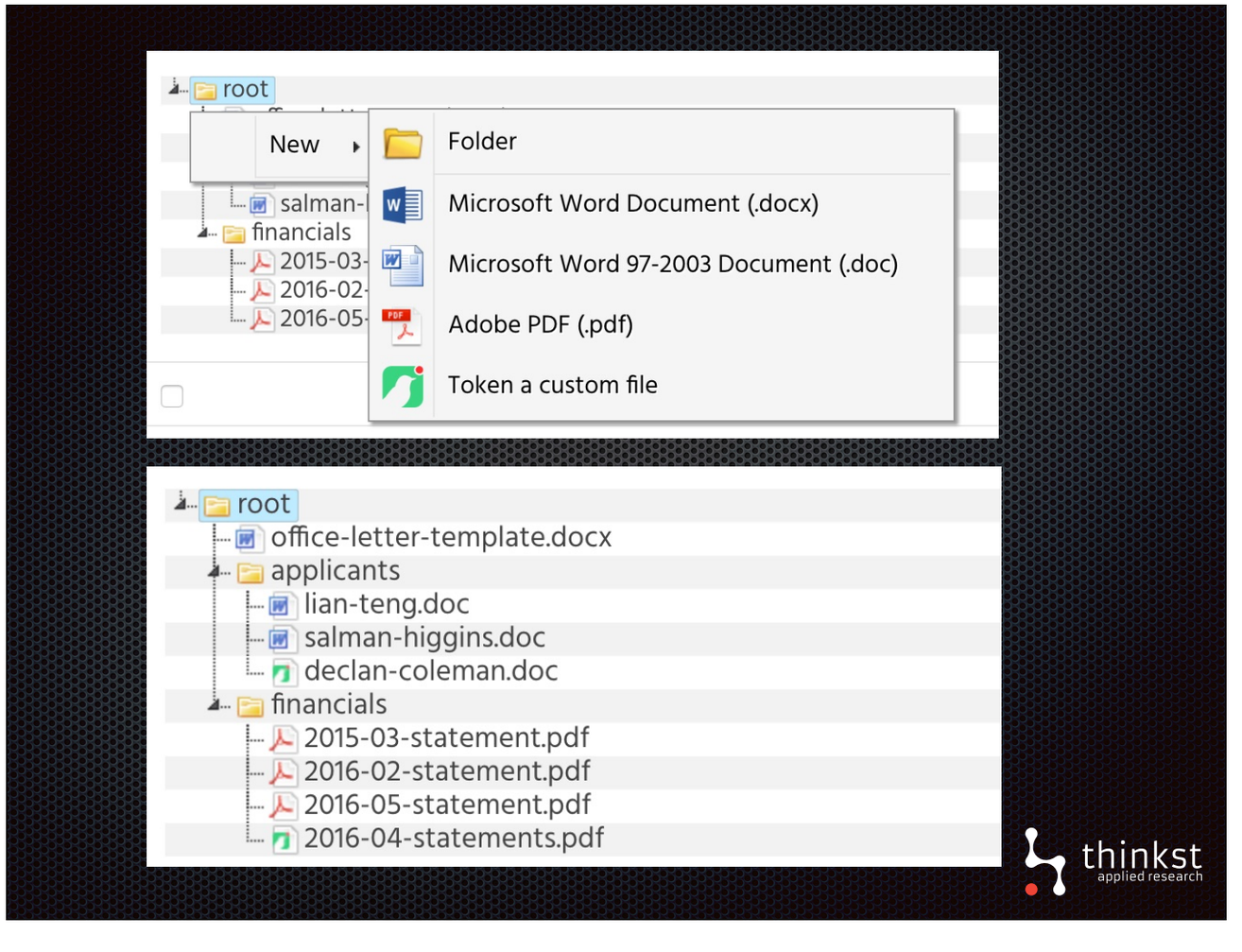

- In our integration tokens can be managed separately and alongside alerting honeypots.

## Honeypot File Share

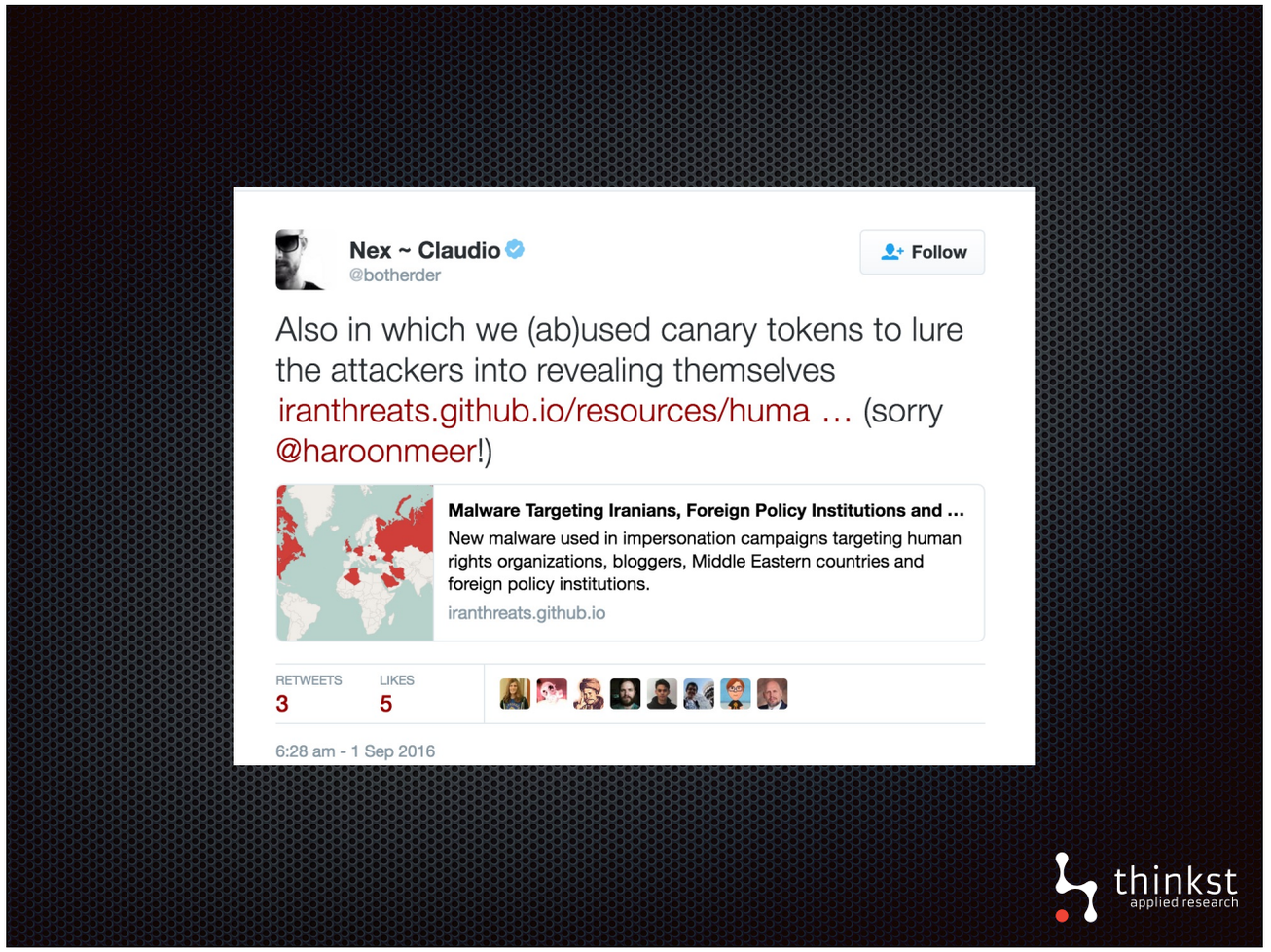



thinkst applied research

- And where it’s possible to customise a file share




- It’s simple to embed tokened files along side the files.
- We mention this because while we doing this, it turns out two other people were doing something quite successfully doing something similar with canary tokens.

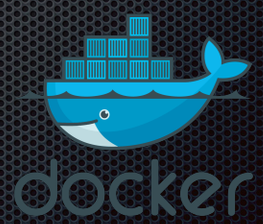



- Claudio Guarnieri & Collin Anderson were investigating attacks using malware on human rights activists' computers
- For those that recognize Claudio as being the creator of cuckoo sandbox for malware analysis, the next part won't be so surprising:
  - They setup a honeypot, and used canary tokens to generate tokened docs on it.
  - When the docs got pulled and opened, they managed to get a bit more information about their attackers.


- They had hits from a number of different IPs mostly VPN endpoints.
- But some of those hits weren’t, and they believe that this is an actual ip used by one of their attackers.
- This is sort of situation where the token uncloaking that nick talked about could possibly help in the future.
- It was a little surprising that they used our public hosted instance, rather than with their own custom domains to be a little more discreet.

# Canarytokens Implementation

- Public site
  - http://www.canarytokens.org/generate
- BSD-licensed
  - https://github.com/thinkst/canarytokens
- Docker images available for easy install
  - https://github.com/thinkst/canarytokens-docker
- Follow @thinkstcanary for updates




thinkst
applied research

- However, everyone's welcome to use the instance we host (and we really enjoy getting the feedback on how it's been used)
- As we mentioned before Canarytokens is open source, so you can use it as you see fit.
- There's also Docker images for ease deployment.

# Wrapping up

- Canarytokens eases building and deploying tokens
- Use defender's home ground advantage to deploy tokens in the way of a attacker
- Try it out and send comments, tokens and code

thinkst
applied research

- We've been talking about our revival of the idea of tokens with the Canarytokens project.
- It's eases the creation of tokens, so that they can be made to look valuable and deployed in places where attackers would snoop around.
- So like those two malware researchers, when there's a chance it could be useful, there's no need build any of the infrastructure - you can pick it up immediately from our hosted instance and you'll easily get started.

- So go ahead: use tokens to cut down those lengthy breach detection times and token all the things.

# Questions?

thinkst
applied research