Honey Tokens, The Universe and Everything

By Tom on July 17, 2006 4:25 PM | 0 Comments | 0 TrackBacks
Let's start off with a lovely definition of what a honey token actually is, from none other than Wikipedia.

"In the field of computer security, Honeytokens are honeypots that are not computer systems. Their value lies not in their use, but in their abuse. As such, they are a generalization of such ideas as the honeypot and the canary values often used in stack protection schemes. Honeytokens can exist in almost any form, from a dead, fake account to a database entry that would only be selected by malicious queries, making the concept ideally suited to ensuring data integrity--any use of them is inherently suspicious if not necessarily malicious. In general, they don't necessarily prevent any tampering with the data, but instead give the administrator a further measure of confidence in the data integrity."

Aside from sounding like something you add milk on and eat in the morning for breakfast, honey tokens are an important part of information security, and are often overlooked. People (and organisations) set up honeypots and completely disregard what a simple honey token can do.

Okay, let me explain a little more. A honeypot is a box that is setup to lead a hacker away from the real target. It is also used for collecting research, i.e. method of rooting the box, what tools/techiques were used, etc. The box has no information on it and is worthless in terms of being a valuable asset to the organisation (or individual). However, I personally see no point in implementing honeypots if your motive for doing so is to have an early warning system to catch hackers. Sure, the honeypot is part of your network, however... what if the hacker doesn't even touch the honeypot? You assume your network perimeter is safe solely on the reason that the honeypot(s) is fine? That's silly. A honeypot is easy to spot from miles away, especially if it's running any kind of VMware - two words for you - TCP fingerprinting.

Wouldn't it be a better idea to have an early warning system alerting of hackers on the very core of your information systems and network? Of course it would! If say, you use a honey token as a form of IDS on the core database of your network then it's going to go completely nutty if it is upset. I think it would also be safe to say that if your core database has been compromised, so has your whole network. Or, it's just a matter of time before all the boxes on the network are compromised. Either way, if you look at implementing a honeypot as a sole form of IDS, then you're very much fucked.

Don't get me wrong, I love honeypots. However, relying on them entirely is a bad bad idea. You need to go with the whole package. An IDS like Snort with some *custom* rules, an IDS/IPS like portsentry (which is able to rewrite the routing table and block the attacker out), a few honeypots and a few honey tokens, all scattered about your network. That's how you win the fight against kiddies. Okay, Mr IT Manager, I hear you say... "Well our honeypot is implemented on our gateway, the *only* way into our intranet over the VPN for example." Okay Mr IT Manager. Do you have dial in lines? Is your PBX secure? Do you *gasp the thought* have wireless access points in your offices? There is *always* more than one way in. You need to think outside the box, think of the bigger picture and react to it.

Right. Now my little rant about why honeypots on their own suck as a form of IDS/IPS is over and why honey tokens rock, we can move on.

Imagine you're a big telesales company in a call centre. These companies I despise - almost as bad as the spammers

like the Data Protections Act, however... we shall ignore that for now. Imagine every employee has unrestricted access to that database. For example, customer Mr Smith may ring up and the employee will need to get his details up on the computer to deal with his enquiry.

Now, imagine you're a 16 year old teenage girl who just got a job at this place and you are looking at a screen with "NAME:" and "SURNAME:" fields waiting for input. Sooner or later you're going to get curious. We are human after all. If you're that teenage girl, sooner or later you're going to end up putting "David Beckham" into those boxes, or any other celebrity for that matter. This is no other than abuse of the system. Of course it isn't if David Beckham actually does ring up... but somehow I doubt he'd have an account with some telesales company in a call centre in Slough for starters anyways. But hey, prove me wrong. Anyways back to the story. We want to be able to be made aware of this abuse as soon as possible (in real time would be good!) Audit/log files are no good, who the hell is going to read through a gigabyte of SQL requests for that day. We all know we should, but just like people checking IDS logs regularly, it ain't gonna happen.

Now, how do we detect this abuse in real time I hear you ask? Hang on... I'm coming to it... and I think you can guess how anyway. By using a honey token of course!

Re-wind the story back to where the teenage girl working there is thinking about looking up some celebrities. Of course on a normal system with no honey token in place, "David Beckham" will just result in a "NOT FOUND" (0 results) back to the user from the database. However, this is where the honey token comes in. Let's create a "David Beckham" record. Let's give him some fake details. However, this is where we are going to be clever. We are going to extend the use of the honey token even further. We have a phone line that isn't used at all back at head office. Put that number in. Now, we have two honey tokens in one. Not only will we be alerted when a "David Beckham" is found via the database by our spotty teenager, but now, if she sells on the information to some friends down the pub later that night... when that phone goes that never rings... you know she has done so! Not only do you know she's accessed the database, but you know the extend and scope of what she has done. She has initially done abuse via the database, but also attempting to phone the number that she thinks is David Beckham's. The phone call also confirms that the database abuse took place and all you need to do is grep the database logs for it. You can then trace this number, whatever, etc... and hand her a P45 in the morning.

Right, that is just one use of a honey token. Let me share my initial thoughts of using a honey token when I think with my paranoid brain.

Wouldn't it be great to be able to tell if someone was sniffing your connection, reading your e-mail, etc.? Yup, it would be lovely. That's why I have come up with the following idea. (I haven't seen this mentioned by anyone else in the security industry as using honey tokens in this way so I will take credit for this... but give me a shout if I'm wrong).

You need to be hosting your own web server, or... have access to the log files. The first option would be better.

Now, let's picture the scene. I send an e-mail to Bob to tell him to buy X amount of shares as I predict something is going to happen on the stock market very soon. Bob gets this e-mail, and buys them for me (isn't he nice?). However... this message is going over an insecure medium, yes... that's right, the Internet. If you didn't realise it was insecure, unplug your modem now, heh. Anyways, what I am trying to say is that at any point the message could have been sniffed in transit from me to Bob. Just do a traceroute/tracert in Unix or MS-DOS and you will see the journey your packet goes. I have no way of telling if someone else read my e-mail and also bought some shares! Bastards.

However... by using a honey token, I will do.

What we do here is create a random document, or even an image on our web server... it doesn't even have to be valid. For example, use "touch" in Unix and just create the file; bah.gif/bah.html. I suggest making it an image for reasons you will soon see. Now, re-wind back again to where I am about to e-mail Bob. I am just about to finish my e-mail... however, I want to whack the honey token in. I do this by adding the HTML <IMG SRC="http://www.myserver.com/bah.gif" height=1 width=1> in the e-mail somewhere. This won't show as it's absolutely nothing, also, it'll be 1x1 pixel in size, so you won't even see it. Thing is, when Bob reads the e-mail now, his client will visit http://www.myserver.com and try and get bah.gif. We can see this by tail -f'ing the httpd access logs in real time and see when he reads it. We will see his IP, browser details, etc... and by WHOISing the IP and seeing the netblock owner - we can approximately guess this is him. See below for an example.

81.151.123.179 - - [17/Jul/2006:21:58:25 +0100] "GET /bah.gif HTTP/1.1" 304 - "http://www.yahoo.com/myemailbah?action=readmail&mail=3" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)"

We see the netblock is owned by BT and that client is using Yahoo mail. Now, if we know we just e-mailed @yahoo.com (being Bob) and we also know he is on BT... it would be safe to assume this is him.

But wait, wait a minute... what's this I see 35 seconds later!

This isn't Bob! A different IP that isn't owned by BT... a different browser... surely he couldn't have installed a newer .NET framework in under 35 seconds???! Plus, it didn't come from Yahoo mail (the referral), although this means nothing except that the e-mail isn't being read from inside Yahoo... it's being read by someone packet sniffing at the ISP or at his computer. We can be 100% that we have a new friend in our communications because of the second request for the file on our web server and we didn't use that bah.gif for anything else - we just created it for this e-mail! The reason I said make an image as a honey token earlier was because we can have it automatically load when someone opens the e-mail, else we would have to rely on them clicking on a link in the e-mail... so go with image every time.

The second example is almost like the first in a way... we are using a web server instead of the database and the phone as a form of alerting us. You can even extend this idea much like the first one. For example, create a user account on your network but NEVER log into it. Send yourself an e-mail with "USER DETAILS: username/password" and leave it sitting in the inbox. If that account ever gets logged into... (the IDS will pick it up with a rule) you can be sure as hell someone just read your e-mail.

Just think of a honey token like this... Remember when you were younger and if you were smart enough when you left your room you'd put a piece of paper lodged between the door (I know I did...). Therefore, the door was holding this paper in place between itself (obviously) and the wall. If someone went into the room while you were away, the paper would drop to the floor. You would use a small bit of paper (or anything really) so it wasn't noticed if someone opened the door. Else they'd just put it back to how they found it! However, as it's so small (just like our 1x1 pixel image) they won't even notice they changed something... whereas they would if they knocked a vase over. Then you'd come back... see the paper on the floor and not in the door and you'd know someone had been in your room! Either that or the wind/gravity took it out of place, heh.

Hopefully this article has opened your eyes up a little and given you a little more information on honey tokens and what you can do with them, whatever background you come from. From the security manager to the paranoid user - hopefully you will use a honey token or two in your e-mails and for general communication over the Internet.

Categories:

- Network Security

No TrackBacks

TrackBack URL: http://www.tomneaves.co.uk/cgi-bin/mt/mt-tb.cgi/18

Leave a comment

Name
Email Address
URL
Remember personal info?
Comments (You may use HTML tags for style)
Preview Submit

Tag Cloud

- academic
- botnets

- cookie stealing
- device configuration review
- dns
- gsm
- port knocking
- satellites
- secure apache
- sms
- snmp
- web proxy
- wireless

Recent Entries

Cookie Monster - A Cookie Analysis Tool

Is it possible to predict cookie values? I've got a feeling it just might be possible. Cookie values are assigned…

By Tom | Comments (0)

fwknop released - Single Packet Authorization and Port Knocking

Port Knocking came about in around 2003, but it has various weaknesses. There are plenty of implentations though (some quite…

By Tom | Comments (0)

onesixtyone 0.3.2 released - An Efficient SNMP Scanner

The SNMP protocol is a stateless, datagram oriented protocol. An SNMP scanner is a program that sends SNMP requests to…

By Tom | Comments (0)

PorkBind v1.3 released - Nameserver (DNS) Security Scanner

This program retrieves version information for the nameservers of a domain and produces a report that describes possible vulnerabilities of…

By Tom | Comments (0)

Surf Jack released - Cookie Session Stealing Tool

A tool which allows one to hijack HTTP connections to steal cookies - even ones on HTTPS sites! Works on…

By Tom | Comments (0)

Long Way Down

…

By Tom | Comments (0)

Powered by Movable Type Pro

Links

- Home