

Wireless Honeypot Countermeasures

Laurent Oudot 2004-02-13

Wireless technologies have spread quickly in recent years and are now widely deployed in corporate environments as well as at home. The human dependency on those technologies has increased to the point where one can find wireless devices almost everywhere, from network devices to laptops, cameras, and so on.

Though these devices support standard security options and protocols useful to thwart common attacks (ciphering, authentication, etc), many kinds of attacks are still possible but are dependant on the real level of security present and the skill of the attacker.

Sometimes, even in companies, blackhat people find open networks with poor or no security in place. Then they can deeply penetrate such easy targets to steal information or bounce anonymously elsewhere over the Internet. These threats come through the external physical barriers (from a parking lot, walking down the street, through windows) or inside your own environment (via zealous network seekers with PDAs or laptops, wireless cards and scanning software).

This paper will introduce honeypots as a countermeasure for wireless environments (more specifically, WiFi-related technologies). So, let's prepare to feed greedy blackhat people with waves of honey to defeat our happy attackers.

1.0 Introduction to wireless honeypots

The Internet is full of excellent resources that describe wireless technologies, wireless threats, wireless security offerings and honeypot technologies. This paper won't cover those points, but will instead focus on the core of the subject: wireless honeypots. In reading this paper, one can suppose you know what wireless networks are, that wireless security issues certainly exist and that there are security resources called honeypots to help mitigate this threat.

We will first describe what a wireless honeypot could be, and then move on to addressing our related goals. Then we will focus on theoretical aspects and design possibilities, before looking at two easy technical examples. And before we conclude, we will introduce some of the limitations for such architectures.

2.0 Definition

If you glance at the web site of Lance Spitzner, leader of the HoneyNet Project, you'll read the definition of a honeypot : "*A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.*" [[ref 1](#)]

So, a wireless honeypot could simply be a wireless resource that would wait for attackers or malevolent users to come through on your wireless infrastructure.

3.0 Goals

Honeypots are still young technologies, and though wireless networks are commonly deployed in the wild, some people ask: why should we use wireless honeypots? Depending on your networks and your security needs, you might be interested in the benefits of wireless honeypots. In the blackhat community, most skilled or curious individuals enjoy penetrating wireless networks because they seem to be:

- safe: you are not really connected while attacking and you could run away in case of detection (ex-athletes are prohibited);
- easy: there are still a huge number of open or non secured access points everywhere (hotspots in hotels, airports, public areas, SOHO wireless networks, etc). With very inexpensive devices now on the market the number is quickly growing;
- still relatively new: wireless networks are considered "fun" to attack;
- stealthy: it's a perfect venue for corporate hackers and evil cyber-terrorists. Such miscreants can randomly use open APs to anonymously launch attacks, worms, and so on, yet nobody will be able to catch them.

Yet most of the time for managers, wireless attacks are considered less dangerous than Internet attacks because the attackers have to be physically near the network devices. Many believe that this happens infrequently, however security should really be a more serious issue in most companies -- particularly due to the relative ease with which a wireless network can be cracked.

Wireless honeypots could help to reveal real statistics about such attacks on your infrastructure, such as the frequency of attacks, the attacker's skill level, his goals and methods. Wireless honeypots can also help with protecting your networks while the attacker expends significant effort on fake targets; thus with honeypots blackhats will lose time in their discovery of your network.

4.0 Theory and design

What can be done in a wireless environment to fool the bad guys? To answer that question, just think about the kind of threats you want to deal with and develop an action plan.

4.1 Wireless activity

First, attackers will try to scan and/or listen to wireless networks, so you may be interested in sending out fake packets, asserting the presence of wireless networks (see FakeAP hereafter). Or, you may be interested in deploying fake wireless resources dedicated to some honeypot infrastructure. A very interesting option would be to simulate traffic through the waves of your honeypot, but at this time no automatic or easy-to-use public tool has been released. One could use something like automated scripts simulating network sessions between an Access Point and its clients, as we'll see below, or use tools that replay recorded packets such as tcpreplay.

Folks from the French HoneyNet Project sometimes use Perl scripts that automate dialogs between clients and servers with random sessions and commands. This idea was first published in June 2003 during the SSTIC in France, by students from ENSEIRB doing some research on UML and HoneyPots [[ref 2](#)].

The following two examples offer such automation, generating random sessions and commands that simulate wireless traffic:

```
#!/usr/bin/perl
# initiated by Michaël HERVIEUX, Thomas MEURISSE
# example of script to simulate an automatic FTP session
# feel free to modify it and add random activity
# launch it from your clients (use cron, etc)
use Net::FTP;
$ftp = Net::FTP->new("192.168.16.98");
if ($ftp == NULL)
{
    print "Could not connect to server.\n";
    exit(9);
}
if ($ftp->login("barbul", "StEugede"))
{
    $ftp->cwd("/home/rpm/");
    $ftp->get("Readme.lst");
    $ftp->quit();
}
```

```

}
else
{
    print "Could not login.\n";
    exit(7);
}

```

```

#!/usr/bin/perl
# initiated by Michaël HERVIEUX, Thomas MEURISSE
# example of script to simulate an automatic SSH session
# feel free to modify it and add random activity
# launch it from your clients (use cron, etc)

use Net::SSH::Perl;
my $ssh = Net::SSH::Perl->new("192.168.16.98", protocol => 2);
$ssh->login("misc", "m4gRul3Z");
$ssh->cmd("who");
$ssh->cmd("uname -a");
# ?

```

Simulating traffic can be a more important issue on a wireless network dedicated to honeypot activity than on a wired one, because attackers need to see traffic in order to perform some of their attacks. Bypassing 802.1X, bypassing MAC address filtering, cracking malformed WEP keys, looking at beacons, looking at SSID in the frames used for connection by clients, and so on all require existing traffic to be analyzed.

4.2 Wireless architectures

You will first need at least one device that offers wireless access. If you choose to use a real Access Point, then you can safely plug it on a wired network (with at least one computer) with visible resources playing the role of targets on this fake network, and invisible resources to record data and detect intrusions (data capture). To monitor wireless-specific layer 2 attacks, one can use data capture on a wireless invisible client in mode Monitor, using software such as Kismet. An example architecture is shown below in Figure 1:

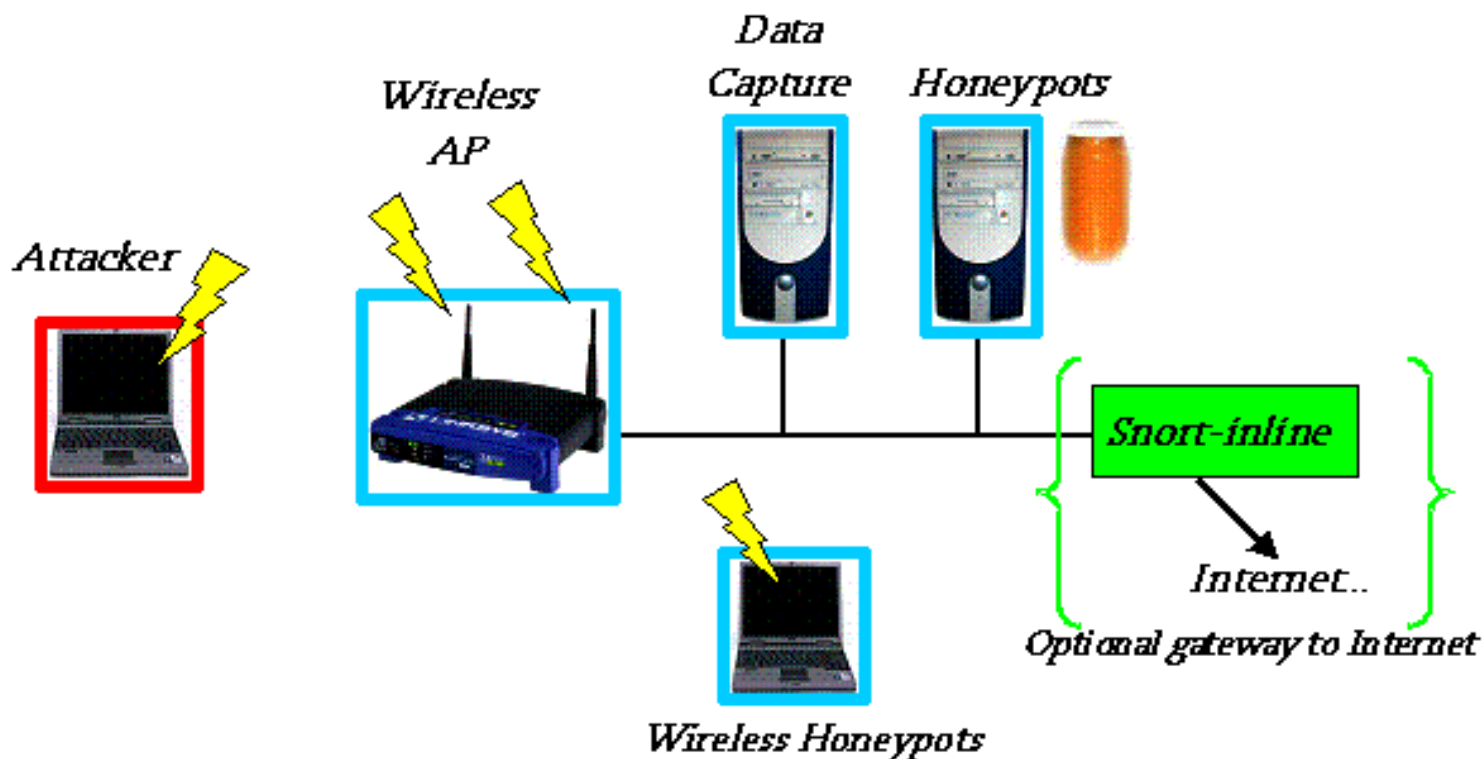


Figure 1: sample WiFi honeypot architecture

If you want to offer Internet access on the honeypot network, to improve the realism and interaction of your network, you should be careful and filter the outgoing network traffic to forbid attacks using a kind of Intrusion Prevention System, like snort-inline from the Honeynet Project. Most of the time, people don't want to make an Internet connection available to a wireless honeypot because of the related risks. Notice however that doing so can be used to understand blackhat activities: where do they want to go on the Internet? How do they try do go on the Internet?

For example, if you only propose free DNS traffic and require authentication for other services (a classic hotspot configuration), you could catch skilled attackers trying to bounce to the Internet with tools that encapsulate traffic over DNS. Such tools would reveal you the remote IP of the server they use to freely access the Internet in their unauthorized tunnel sessions (Nstxd server, for example) which could eventually be used to sue them. If blackhat people were aware of such risks, they would hesitate before doing illegal actions and the impact of wireless crimes would be reduced.

Another option could be the use of wireless clients on such architecture. Usually, people deploying honeypots propose servers, but clients can be used to improve the realism or to monitor specific attacks. More specifically, on a wireless environment, clients can be used to simulate wireless traffic and also monitor layer 2 attacks and probes. In fact, some attackers listening to the

wireless network traffic will recognize the presence of clients. Sometimes, those clients are not well configured and badly protected (such as laptop used from home and brought to a company) and become interesting, easy targets. As an example, an attacker could try to use a Rogue Access Point with a stronger wireless signal than the official wireless AP. A typical client will then automatically connect itself to the attacker's rogue access point and specific, evil actions can then be tried by the attacker: man in the middle attacks, denial of service, infection with a new worm that spreads itself on the rest of the legitimate network after the client reconnects itself, and so on.

To look at easier solutions, one can also turn a wireless card in Master mode to simulate an Access Point, so that the honeypot is concentrated on only one computer. This is really cheap and easy to manage. Even if the honeypot is compromised, you should not have any problem if it's disconnected from your real network. Moreover, this computer could be either a high-interaction honeypot or a low-interaction honeypot. As an example, you could use a wireless computer (a laptop for testing) with Honeyd, as will be explained.

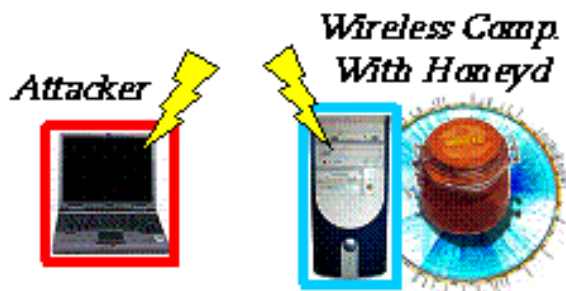


Figure 2: simple wireless client in Master mode, with Honeyd

Another possibility would be to modify a wireless Access Point directly and transform it as a honeypot. I've begun work on that and my results should probably be released when I have enough time; to do such a funny thing, I rebuilt my own firmware on my favorite AP which is the cheap WRT54G from Linksys, and its sources have been publicly released under GPL! By slightly modifying some classic tools such as Honeyd, one can compile MIPS binaries that would work on this AP (running Linux 2.4.5) and create a very geek, customized, wireless embedded honeypot. Though that's a personal and somewhat funny solution, I suppose that official commercial products will ultimately propose wireless honeypots as well (embedded or not).

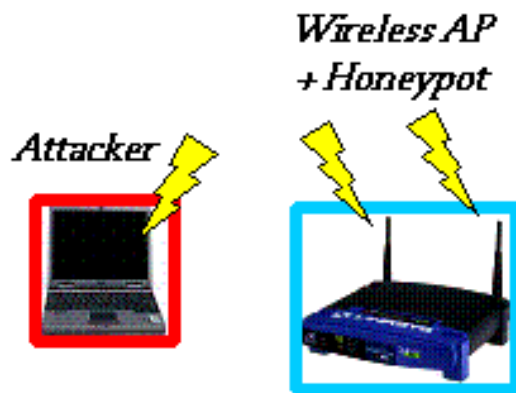


Figure 3: modified access point, hacked firmware and Honeyd

An additional, and rather evil possibility could be the use of a rogue access point, passively waiting for incoming unauthorized wireless clients, to automatically counterattack them. If you are interested in Evil Honeypots discussions, you should definitely come to next CanSecWest Conference organized by Dragos Ruiu [[ref 3](#)].

5.0 Practical examples

Here are two easy examples for creating wireless honeypots.

5.1 Honeyd

- Simulating a network behind the wireless access

If you look at the work by Niels Provos, author of the well known tool called Honeyd, you'll find an example configuration to set up a fake Internet routing topology that can be used on a wireless honeypot architecture [[ref 4](#)]. This is a simple configuration to show how easy it is to simulate a huge network on a wireless environment. Such architecture was used during a conference called Libre Software Meeting 2003, where unsuspecting end users connected themselves to a fake network without seeing it was not, in fact, a real one. With such an architecture, an outside attacker could think he has found a big network and would probably lose hours before understanding that it is not.

- Simulating a wireless AP

One other interesting possibility of Honeyd is the creation of fake TCP/IP stacks to fool remote fingerprinting tools such as nmap or xprobe, and this is an easy way to create your own fake services. For example, by copying well-chosen web pages used to manage an access point, one

could really simulate an AP. This technique can be useful to monitor attackers who would try to connect to the management interface using well-known default passwords, or who would try other opened services (such as attacks over SNMP, DNS, DHCP, TFTP, etc).

For example, here is a quick test that could be tried on a laptop with a wireless card turned in Master mode and Honeyd listening on it. Suppose you want to simulate a Linksys WRT54 Access Point with a web server used for administration. Just ask Honeyd to simulate this stack and web server, as follows:

```
create linksys
set linksys personality " Linux Kernel 2.4.0 - 2.5.20"
add linksys tcp port 80 "/bin/sh scripts/fakelinksys.sh"
add linksys udp open 53 open
add linksys udp open 67 open
add linksys udp open 69 open
set linksys tcp action reset
bind 192.168.1.1 linksys
```

By using a tool like nmap (-O for Os Fingerprint) a remote attacker could see :

```
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
```

And to simulate the web server used to administer the Linksys, honeyd will launch fakelinksys.sh to handle web requests. This fakelinksys.sh script could be as follows:

```
#!/bin/sh
DATE=`date`
echo "== Httpd break-in attempt [ $DATE ] ==" >> /tmp/linksys.log
while read request
do
    LINE=`echo "$request" | egrep -i "[a-z:]"`
    if [ -z "$LINE" ]
    then
        break
    fi
```

```
    echo "$request" >> /tmp/linksys.log
done
echo "==" >> /tmp/linksys.log

cat << _eof_
HTTP/1.0 401 Unauthorized
Server: httpd
Date: $DATE
WWW-Authenticate: Basic realm="WRT54G"
Content-Type: text/html
Connection: close

401 Unauthorized

Authorization required.

_eof_
```

On such architecture you should probably see incoming wireless attackers trying the default login password (linksys/admin), thinking they are really attacking such a device.

5.2 FakeAP

If you remember the movie called War Games, the young adolescent was using a modem on the phone line to scan remote phone numbers and find open lines like BBSes. This activity was called wardialing, and by transposition in the wireless world, people talking about wireless scanners or wireless listeners as wardriving, or even warwalking. Wardrivers try to find open networks. A good first idea to delude those potential intruders would be to simulate as many fake networks as possible for them to lose time and patience. Targeting one network is an easy task, whereas dealing with a cloud of targets could be more difficult.

This proof of concept was done with a tool called FakeAP [[ref 5](#)], free software distributed under GPL by the guys from Black Alchemy during the Defcon X. This tool can send specific wireless network traffic to fool basic attackers. As a wardriving countermeasure, it generates 802.11b

beacon frames as fast as possible, by playing with fields like BSSID (MAC), ESSID, channel assignments, and so on. This trick is easily created by playing with the tools used to manage a wireless card (under Linux, that's like manually playing with: `iwconfig eth1 ESSID RandomSSID channel N...`). A remote, passive listener should then see thousands of fake access points! To quote the web site of the authors: *"If one access point is good, 53,000 must be better."* The idea behind this simple tool was quite good when it was first released, and you could even detect NetStumbler users by looking at 802.11b probe requests/responses. Whereas now, most updated tools can advise the attacker that the detected access points are unusually strange, such as these cases where no traffic is generated on the found networks. Figure 4, below, indicates a NetStumbler scan on one of these honeypots:

MAC	SSID	Name	Ch.	Vendor	Ty.	En.	SN	Sign.	No
0000CE992FA4	TrackingHackers		10		AP		30	-56	-86
0000CE1DD8D2	CanSecWest		10		AP			-54	-86
0000CE69BFB0	Barbus		10		AP			-54	-89
0000CB9C890	CanSecWest		10		AP			-56	-88
0000C193CF9	Moutane		10		AP			-54	-90
0000CE991A75	SSTIC		10		AP			-54	-87
0000CE3EC028	SSTIC		10		AP			-54	-91
0000C1BDF30	SSTIC		10		AP			-56	-85
0000CE43B002	Rstack		10		AP			-53	-89
0000CBF3100	Moutane		10		AP			-54	-91
0000CE082274	Moutane		10		AP			-56	-86
0000C2CA061	MiscMag		10		AP			-55	-88
0000CEFB05A3	MiscMag		10		AP			-55	-91
0000CE2EBED5	Moutane		11		AP			-57	-89
0000C72DA69	Moutane		11		AP			-58	-87
0000CED52D36	Moutane		11		AP			-60	-87
0000CF5FCF18	Barbus		11		AP			-60	-87

Figure 4: NetStumbler scan on a FakeAP honeypot

6.0 Limitations

If you think about deploying honeypots to fool attackers, you will have to perfectly simulate reality (a common honeypot theme -- ie, 'what is the Matrix?'). Many counter papers have recently been released on the Internet because blackhat people want to prove that they are not afraid of honeypots and that they are stronger than their creators. This public game between the good guys and the bad guys will surely improve honeypots technologies, albeit passively, and new paths of research have been drawn to resolve the stealth problems.

Wireless honeypots suffer from the same stealth problems that classic honeypots do, and also from specific, additional ones related to this environment. Remember that skilled attackers may

be afraid of "too open" networks. So, the rules of the game are easy:

- the better you simulate reality, the more you'll catch skilled attackers (but in this case, intrusions rarely occur);
- the less you deal with stealthiness, the more you'll see successful attacks (but they are often done by 'kiddies' or inexperienced attackers).

Therefore, depending of your goals, you might create honeypots with or without these options:

- Beacon transmission;
- WEP (or more generally, ciphering, that can be cracked more or less easily);
- MAC filtering;
- 802.1X authentication;
- Wireless traffic between clients and AP;
- Wireless clients with auto-connect mode enabled;
- Wireless networks using well known standards (802.11b, 802.11g, 802.11a?).

7.0 Conclusions

Though we cannot cover all the practical and technical aspects of wireless attacks in one single document, this paper should help you with creating your own wireless honeypots. This new kind of security resource could easily become an effective way to monitor wireless intrusions attempts and to understand a blackhat's goals and their corresponding tools. Whether these people are corporate attackers, bandwidth borrowers, or cyber terrorists, they will be discovered.

To conclude, one should note that there are very real examples of well known wireless honeypots already deployed: the Science Applications International Corporation (SAIC) created one of the first huge wireless honeypots in Washington DC in order to catch WiFi hackers [ref 6], as shown in the Figure 5 map, below.

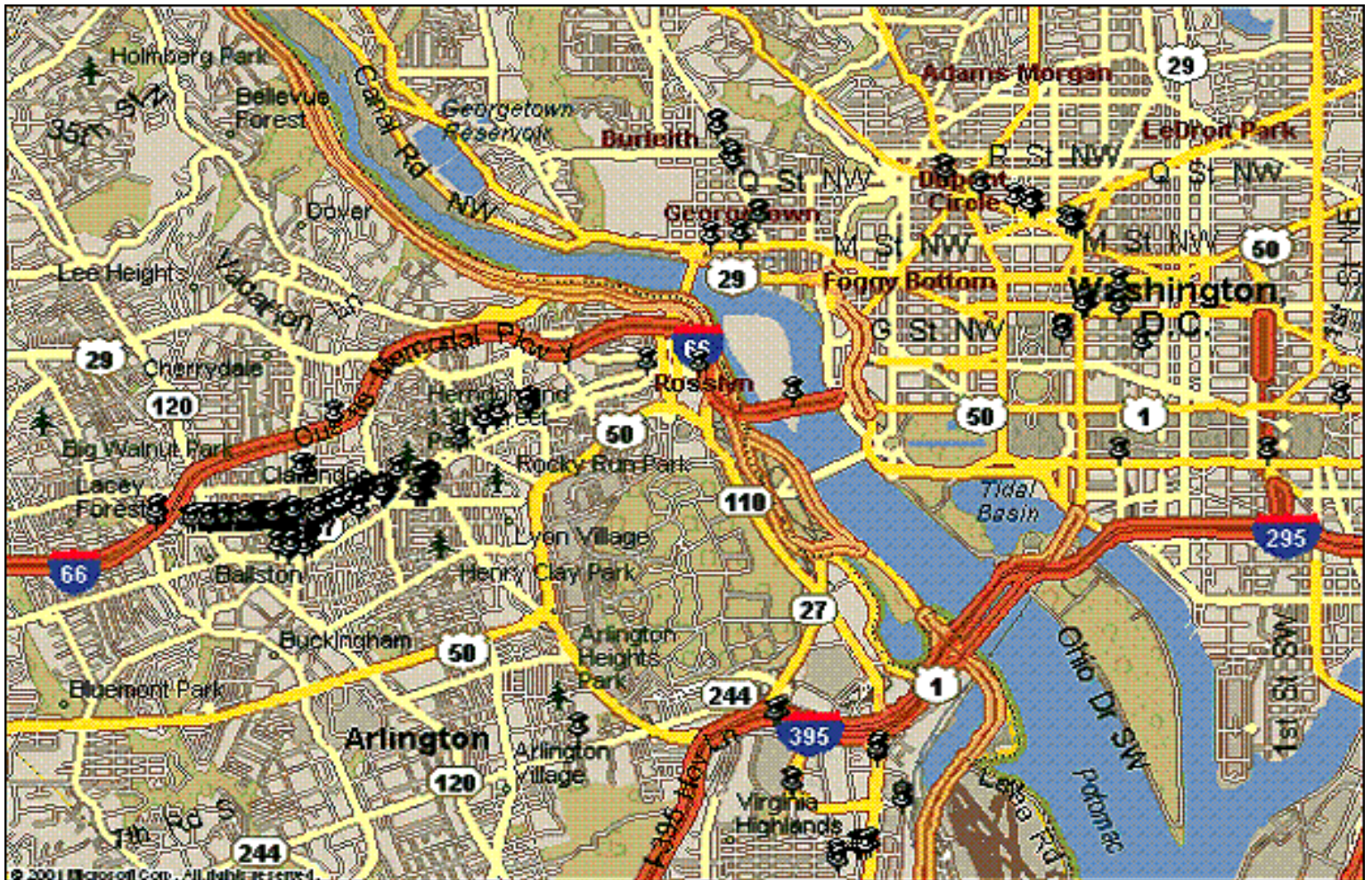


Figure 5: SAIC's huge WiFi honeypot farm in Washington DC

References

[ref 1, Lance Spitzner's web site : <http://www.trackinghackers.com>]

[ref 2, Hervieux and Meurisse, Symposium SÃ©curitÃ© des Technologies de l'Information et des Communications, SSTIC 2003, Rennes, France, UML as a Honeygot, <http://www.sstic.org/SSTIC03/resumes03.shtml#UML> and http://www.sstic.org/SSTIC03/presentations/Honeypots_UML___M._Hervieux_T._Meurisse/]

[ref 3, CanSecWest 2004, Towards Evil Honeygot, when they bite back <http://www.cansecwest.com>]

[ref 4, Honeygot project, by Niels Provos : wireless honeypots examples at <http://honeygot.org/config/wireless> and at <http://honeygot.org/configuration.php>]

[[ref 5](#), FakeAP tool, by BlackAlchemy : <http://www.blackalchemy.to/project/fakeap/>]

[[ref 6](#), Wi-Fi Honeygot a New Hacker Trap, by Kevin Poulsen, <http://www.securityfocus.com/news/552>]

Credits

Thanks to Lance Spitzner and other Honeygot folks for the small but interesting discussions we had about WiFi and honeypots in Chicago, during the annual Honeygot meeting 2003 (ugly WiFi network quickly set up with strange packets caught). Special greetings to the wireless experts from the French Honeygot Project, team Rstack and its weird sub-team Droids (Troglocan, etc).

About the Author

[Laurent OUDOT](#) is a computer security engineer employed by the Commissariat a l'Energie Atomique in France. On his spare time, he is a member of the team [Rstack](#) with other security addicts. Concerning honeypots, Laurent is an active member of the [French Honeygot Project](#) which is part of the [Honeygot Alliance](#).

View [more articles by Laurent Oudot](#) on SecurityFocus.

[Privacy Statement](#)

Copyright 2006, SecurityFocus