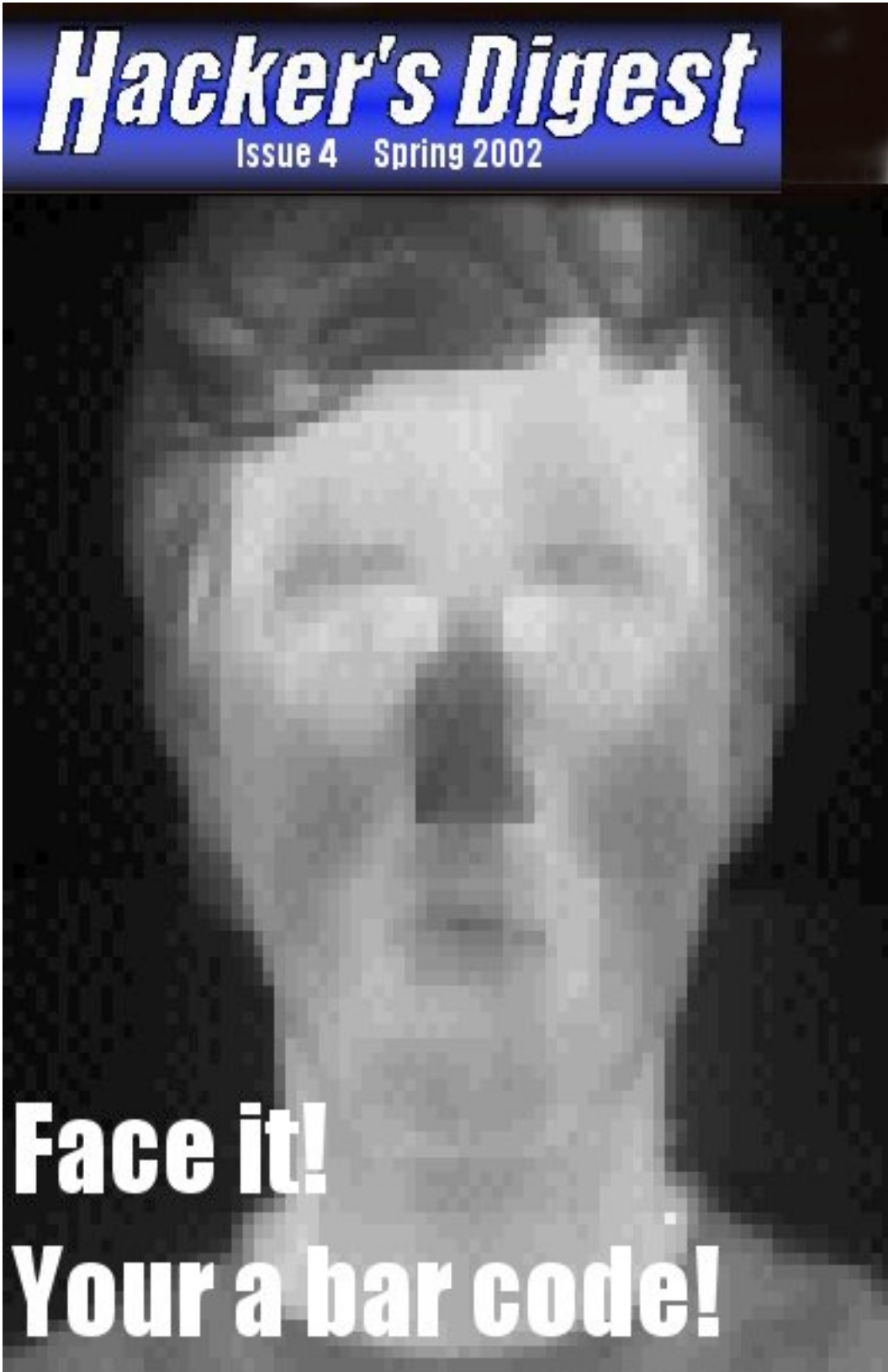
Hacker's Digest

Issue 4 Spring 2002



Face it!

Your a bar code!

"The civilized world has been attacked by terrorists. We have to defend ourselves. It's wartime, and we have to give up some civil liberties in order to secure ourselves against the danger."

-Some Jackass

~ Staff ~

John Thornton
Editor-In-Chief

Tawnya Richards
Office Coordinator

Laszlo Acs
Editor

Circuit
Steady Writer

Dark Fairytale
Steady Writer

Writers:

UnKnOwN Source

Actinide

iDefense

the Pull

k00p$ta

Phr34k

ic0n

Obscure^

HACKER'S DIGEST

Spring ISSUE 4 2002

FACE IT!

YOUR A BAR CODE!

With the U.S. still nervous after the terrorist attacks, face recognition systems are starting to show up more and more, to the point that its not making the news anymore. These systems scan your face as if it were a bar code against a database of individuals. There are some very good reasons to have face regognition systems installed. For example, a bank could keep track of who is entering and exiting a safe without having to go though hours of video tape. It could liment access to wepons, evidence, nuclear materials, or biohazards.

However, there is much more money to be made then selling these systems to banks, governments, and precinets. Why not put these systems all over the streets to the point that you can not take out your trash without getting scaned. We could get all the terrorist! In a perfect world, perhaps this system could be put into place and never be abuse. In a perfect world we would not need these systems. The problem is that the power to track an individual at all times is absolute power and we all know absolute power corrupts. Lets take a look at how these systems can be abused.

Abuse, Abuse, and more Abuse

Crime
With the internet in just about every place of business face recognition sytems could be networked with other compaines' face recognition systems. What happens if one of the workers swipes the database and sells it to anyone who is willing to throw money at him? The collected data could be used to spot patterns and when you take off for your nine to five job your house is getting riped off?

Mistakes
Well, we know that there will be mistakes with the system. False positives among the obvious but lets say that you are seen on a street that is frequented by drug dealers. These reports will create facts that will need to be explained. Shadows, occlusions, reflections, and multiple uncontrolled light sources play a factor in false positives.

Face recognition is nearly useless for identifying terrorists in a crowd
"Face recognition is nearly useless for the application that has been most widely discussed since the September 11th attacks on New York and Washington: identifying terrorists in a crowd. The reasons why are statistical. Let us assume, with extreme generosity, that a face recognition system is 99.99 percent accurate. In other words, if a high-quality photograph of your face is not in the "terrorist watch list" database, then it is 99.99 percent likely that the software will not produce a match when it scans your face in real life. Then let us say that one airline passenger in ten million has their face in the database. Now, 99.99 percent probably sounds good. It means one failure in 10,000. In scanning ten million passengers, however, one failure in 10,000 means 1000 failures — and only one correct match of a real terrorist. In other words, 999 matches out of 1000 will be false, and each of those false matches will cost time and effort that could have been spent protecting security in other ways. Perhaps one would argue that 1000 false alarms are worth the benefits of one hijacking prevented. Once the initial shock of the recent attacks wears off, however, the enormous percentage of false matches will condition security workers to assume that all positive matches are mistaken. The great cost of implementing and maintaining the face recognition systems will have gone to waste. The fact is, spotting terrorists in a crowd is a needle-in-a-haystack problem, and automatic face recognition is not a needle-in-a-haystack-quality technology. Hijackings can be prevented in many ways, and resources should be invested in the measures that are likely to work." a very good fact from Philip E. Agre paper "Your Face Is Not a Bar Code: Arguments Against

Automatic Face Recognition in Public Places".

In Tampa Florida where face reginition was installed and abandoned because of false positives. "The earliest logs provided by the department show activity for July 12, 13, 14, and 20, 2001. On those dates, the system operators logged fourteen instances in which the system indicated a possible match. Of the fourteen matches on those four days, all were false alarms," the ACLU notes.

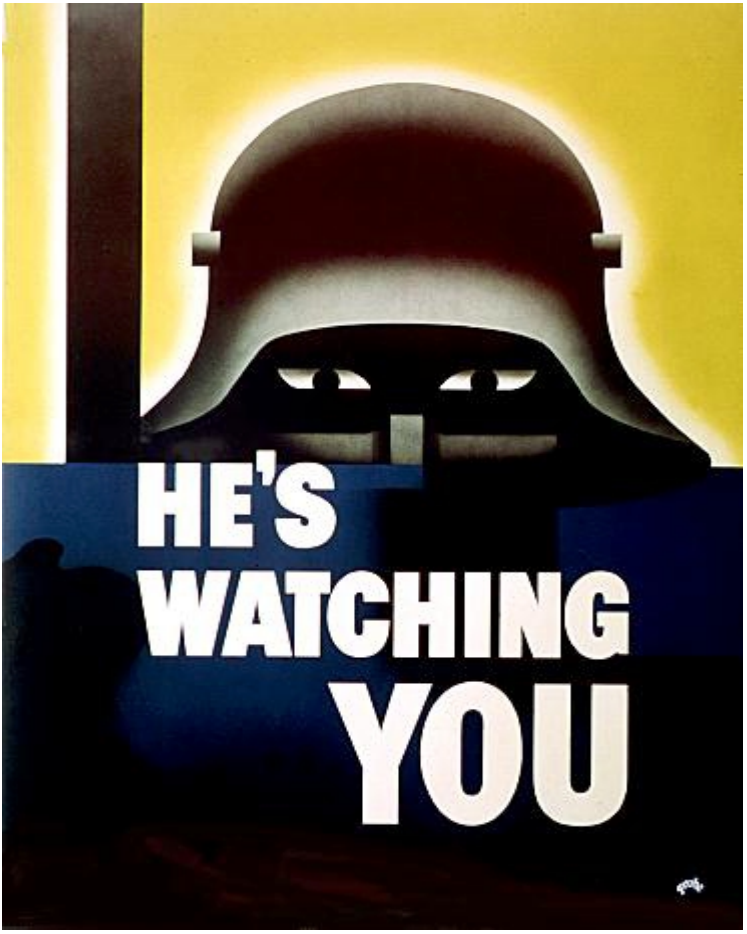
"All of the people in our database are wanted criminals. We don't store any of the images that our cameras capture, except when they match an image in the database. So the only people who have any cause for complaint are criminals."

How easy would it be to have a database of just about every individual in the U.S.? That would take years right? Do you have a licence, state id, or passport in your wallet? If not, have you been arrested?

References

Philip E. Agre "Your Face Is Not a Bar Code: Arguments Against Automatic Face Recognition in Public Places".

http://dlis.gseis.ucla.edu/pagre/



COLOR CODING

By: ^CircuiT^

Ok well im writeing this cause most the files i see out there that try to explain the 25 wire color standard do a shit job of it so not only will i tell you the truth about it ill go into some detail on the history of telephone wire colors. The telephone companys have been useing colored wire since the very first telephone systems in like 1890 but it wasnt until about 1950's that the standard of the 25 color pair telephone wire was adopted by the 1960's all Bell offices were using it and its still used to this very day. Enjoy my article please.

Note: If you see a wireing color code other then this it means the phone compamy hires works from a work release program.

Standard Telecom Color Codeing

Pair #	Tip (+) Color	Ring (-) Color
1.	White	Blue
2.	White	Orange
3.	White	Green
4.	White	Brown
5.	White	Slate
6.	Red	Blue
7.	Red	Orange
8.	Red	Green
9.	Red	Brown
10.	Red	Slate
11.	Black	Blue
12.	Black	Orange
13.	Black	Green
14.	Black	Brown
15.	Black	Slate
16.	Yellow	Blue
17.	Yellow	Orange
18.	Yellow	Green
19.	Yellow	Brown
20.	Yellow	Slate
21.	Violet	Blue

Pair #	Tip (+) Color	Ring (-) Color
22.	Violet	Orange
23.	Violet	Green
24.	Violet	Brown
25.	Violet	Slate

25-Pair Color Coding/ISDN Contact Assignments

```
Designation     Wire color     |              |  Wire color
Ring           Blue/White      | 1  >|   |< 26 |  White/Blue     Tip
    |
Ring           Orange/White    | 2  >|   |< 27 |  White/Orange   Tip
    |
Ring           Green/White     | 3  >|   |< 28 |  White/Green    Tip
    |
Ring           Brown/White     | 4  >|   |< 29 |  White/Brown    Tip
    |
Ring           Slate/White     | 5  >|   |< 30 |  White/Slate    Tip
    |
Ring            Blue/Red       | 6  >|   |< 31 |  Red/Blue       Tip
    |
Ring           Orange/Red      | 7  >|   |< 32 |  Red/Orange     Tip
    |
Ring           Green/Red       | 8  >|   |< 33 |  Red/Green      Tip
    |
Ring           Brown/Red       | 9  >|   |< 34 |  Red/Brown      Tip
    |
Ring           Slate/Red       | 10 >|   |< 35 |  Red/Slate      Tip
    |
Ring           Blue/Black      | 11 >|   |< 36 |  Black/Blue     Tip
    |
Ring           Orange/Black    | 12 >|   |< 37 |  Black/Orange   Tip
    |
Ring           Green/Black     | 13 >|   |< 38 |  Black/Green    Tip
    |
Ring           Brown/Black     | 14 >|   |< 39 |  Black/Brown    Tip
    |
Ring           Slate/Black     | 15 >|   |< 40 |  Black/Slate    Tip
    |
Ring           Blue/Yellow     | 16 >|   |< 41 |  Yellow/Blue    Tip
    |
Ring           Orange/Yellow   | 17 >|   |< 42 |  Yellow/Orange  Tip
    |
Ring           Green/Yellow    | 18 >|   |< 43 |  Yellow/Green   Tip
    |
Ring           Brown/Yellow    | 19 >|   |< 44 |  Yellow/Brown   Tip
    |
Ring           Slate/Yellow    | 20 >|   |< 45 |  Yellow/Slate   Tip
    |
Ring           Blue/Violet     | 21 >|   |< 46 |  Violet/Blue    Tip
    |
Ring           Orange/Violet   | 22 >|   |< 47 |  Violet/Orange  Tip
    |
Ring           Green/Violet    | 23 >|   |< 48 |  Violet/Green   Tip
    |
Ring           Brown/Violet    | 24 >|   |< 49 |  Violet/Brown   Tip
    |
Ring           Slate/Violet    | 25 >|   |< 50 |  Violet/Slate   Tip
```

VIRUS PREDICTIONS

BY UnKnOwN Source

This article was written initially for Anti-virus companies, who blatantly ignored it. I will NOT be held accountable for anything in this text. If anyone is at fault, it is the great anti-virus companies like Mcafee (who I contacted but was snobbed by) and Microsoft for persistantly developing software which is easy to penetrate. I sympathise with any people who may be affected by this text by a melicious developer however, someone needed to inform the world of how valnerable our very infrastructure is. I am actually very sorry that it had to be released. I am aware of the implications of the text.

The rising of an electronic era is here. Everything from bank records to your personal identity are stored within computer storage space which incorporates the latest in Magnetic technology and in places where fast computer storage is necessary, storage is done using holographic technology. While computers allow extremely fast access to information, as opposed to paper records which can't me maintained indefinitely. The increased internet connectivity, which effectively acts as the heart of commerce in society today, has also increased the possibility and Verosity of viruses, which could effectively cripple the world's economy. Even worse, the increase of broadband connections by home users and wideband by businesses acts as a superhighway for such viruses. Self spreading viruses don't depend on file-size no longer because of high speed connections, and a victim could easily receive a 10 megabyte virus, without even noticing. What makes the situation worse is that most broadband connections remain on, allowing viruses to propogate while the user is sleeping. Unfortunately, with no limit to file size due to fast transfer speeds, new viruses can use the increased bandwidth to spread to epidemic proportions before even being noticed. Melissa is an example of such a virus, which managed to take the world by storm and was awed by novice computer users, dubbing it as a "super-worm". But elite computer users know better. The virus was poorly written and in reality, wasn't very powerful. It was simply a new concept.

In the future, it takes only one elite programmer to destroy a trillion dollars of data and records. Viruses could be used as a very effective terrorist attack or simply an act of war against another country, programming it to strongly affect one country, while only minimally affecting others (by making the virus only damage those systems which are within that countries IP range). Whatever it is, we can be sure that such a Super virus will occur in the future, the only question is are we are prepared for it?

The answer is NO. All of today's virus detection programs defend computers against previously used techniques of infection, none making any attempt to predict techniques that could be used in the future by Virus writers. Heuristics, a technique used to detect possible viruses is a component of many virus scanners that many debate would reduce the chances of a super virus being created to so low, that it will never happen. However, several viruses have already been released which are designed to bypass advanced detection techniques, only being detected by recently updated anti-virus definition files, or by a would-be victim who suspects the file is a virus.

In the future, we should expect features such as these:

-Ability to attack more then 1 OS with the same virus (which would be accomplished by using multiple files. Which are sent all together, one at a time, but only activating a suitable one for that computer)

-penetration via multiple random exploits (to reduce the ability to track the virus' origin)

-Stealth port scanning and TCP/IP fingerprinting to enhance decision of targets and ability to penetrate them by built-in exploits, reducing the possibilty that the virus will attack and get detected by a well defended computer.

-delayed mail attack synchronized by online atomic clocks to ensure that every computers virus activates at exactly the same time around the world. The reason why it would be delayed would be because mail transmission is easily detectable

and obvious to the victim. The idea would be to keep the virus secret for as long as possible. More then one attack date might be randomly chosen as well, to prolong the virus attack.

-Might infect specific executable files such as MSN messenger, which automatically starts up with windows to reduce detection further, instead of modifying the boot sectors or registry, which would be detected by AV software.

-Might infect files sent over popular P2P clients, possibly acting as a P2P client (which is possible as the source code for many P2P clients are freely available), which would automatically share infected executable files on the users computer, and pretend to be a client.

-Advanced polymorphic abilities to help evade detection

-There might also be a delayed wipe of the user's computer. After the virus has transmitted itself to multiple victims, it might wipe the users data with a DOD compliant wipe (or a variant, sufficient enough to only make files unrecoverable) to remove traces of itself, and also destroy data. This is where the real damage would lie. The virus may also wipe only every dozen sectors or so sectors instead of every individual one (to speed up the process), and wipe the Fat allocation tables/file indexes on the operating system

-Ability to communicate amongst themselves using an Advanced P2P network, sending information amongst themselves on targets already attacked and targets which seem to be well protected, maybe even organising a DOS attack against such well protected targets to wreck havoc on the internet.

-Mass flooding of the internet to prevent people from recieving anti-virus updates.

-Integration of common social engineering techniques

Today's virus detection scenarios are based on the concept that the victim suspects that a file is suspicious and on anti-virus definitions. However, super-viruses of the future wouldn't need to be run by the computer, as it would execute itself without user intervention and might even disable the antivirus software.

A super worm of the future might run like this: A virus would be compiled for different platforms, probably Linux and Windows as they are the most dominant. The virus would maintain a long list of exploits for many operating systems and programs. To spread, it would use TCP/IP fingerprinting to locate target computers, using OS detection and only attempting to infect the ones which are vulnerable to the virus, breaking into them using random exploits. It might maintain a list of previous computers along the chain and use a backdoor created by the virus on each computer to recieve information on what they attacked. Once the virus has broken into a system, it would finishing transfering all the separate components of the virus, designed for different operating systems. It might possibly use stealth port scanning, to locate hosts, maybe changing the types of stealth scanning techniques used to reduce the possibility of the virus being detected. Once a host is infected and had been sent all the different components of a file, the virus might check the host, to determine if it has any security measures, such as anti-virus or a firewall, attempting to disable them or replace them with fake programs which are designed to imitate the computer, but not protect it. THe virus would then proceed accordingly, using the most suitable approach to break into another computer, while also reducing the chance of it being detected.

One thing that many viruses have lacked in the past is proper coordination. Viruses such as CIH had some coordination, activated only at a certain date of the year, but it isn't very effective because of time zones, which would cause the virus to be in fact activated over 24hours, which leaves a long reaction time. Proper time coordination is important as it allows a massive attack to occur at exactly the same time, before anyone can react, as opposed to small attacks occurring continuously, which would allow anti-virus companies to update their software before the virus reaches its peak. An example of the use of time coordination techniques is: After a host is infected, the virus might continue the next component of its attack by synchronizing the victims computer time with online Atomic clocks. If it fails to do so (because the BIOS or write-protected or otherwise), it might not perform the next sequence until it is capable of synchronizing itself, or until it detects that that change of time exceeds a certain amount (which would be set to a safe time such as 2 weeks). To eliminate the chances that the victim puts his clock ahead of the attack time, the virus might continuously monitor the time to check

for discrephencies, and compensate for it. The next sequence might involve sending emails, with the virus attached, to everyone in the address book. Previously used headers by the victim might be used in the form FW:" to strongly increase the chance of the virus being run. Computers which weren't able to synchronize themselves because of secure conditions would be set to perform this action at a later time. After all the emails had been sent, the virus might wait until the next reboot and then wipe the hard-disk. It might even go as far as too re-flash the bios and mark the MBR as bad sectors, and overclock the video card dramatically, to render the hardware components as broken. The viruses sent in the waves of attack by email might be similar to the first ones, but synchonised for a later time and possibly have slightly different devestating functions, exploits and methods.

Communication is also important as it allows information to be gathered, reducing detection. The virus may have the ability to communicate to computers previously infected on the "chain of infection" to improve intelligence amongst the viruses, so that they know who is already infected and not. Communication will strongly benefit the viruses as there is no point of wasting time infecting a computer already infected, or which is known to be immune against the virus.

Such a virus would be relativelt easily programmed, and would be very similar to Artificial intelligence, gathering information and using it similar to how hackers in society work today. Coordinating their own attacks and making decisions based on the scenario.

A very good social engineering technique that could also be used is social engineering people over IRC. By using bots which respond to people like a human, and then after a while revealing itself as a bot, asking the target if they want a copy of the bot/virus, sending it to them if they do, alot of people will be so impressed by the bot, they will download it from them and infect themselves.

How do we eliminate such a virus? Virus scanners should be bundled with firewalls, being possibly integrated into the same product. Email companies should promote Email scanning using online virus scanning services, (which is already available by some companies). All virus scanning software should also have an automatic updating feature. Increasing the initial cost of the software, to allow for unlimited updates as opposed to providing subscription services would strongly decrease the possibility of users being left with un-updatable software. The virus scanners should also analyze file size changes in common startup files, and should be able to determine "illegal" changes of the time, asking the user whether the change was permitted. They should also be able to restrict access to the address book.

By infecting files which already boot up with the computer, a virus could evade detection by newbies and cocky security people who believe that they dont need virus protection (I used to be one of them) as no changes would need to be made to boot-up files, reducing detection by Registry monitors or file monitors, commonly used to debug programs. There wouldn't be any suspicious entries added anywhere, or any suspicious files left on the computer.

To put things into perspective (for those who are uncreative), imagine the following scenario. Imagine owning a company which makes a massive transaction of 1 billion dollers seconds before the banks infected computer is formatted by a virus. No record of the transaction would be left, and a billion dollars would possibly be left floating around in cyberspace, never to be recovered. Or, the transaction might occur before a superworm hits, and after the system is formatted, by default it would revert to the backup. Suddenly, the company might regain their money again, incorrectly. A breakdown of the economic system would occur, as well as a breakdown in the banking system and chaos would occur. Does that sound like fun (unless of course u extracted $10,000 from your account milliseconds before the system is devastated by the virus).

A melicious Virus could also freeze the online world for a while. Instead of formatting the computers at the same time, the viruses might instead flood the internet with packets to random computers at the same time, prioritising attacking IP's which pass through alot of other computers. The virus would be invincible during this time as the user wouldn't be able to recieve virus updates. In fact, the Internet may even need to be shut down entirely or sectioned off, to allow users to be able to update their virus software independantly, unless of course the virus has made all its packets maximum priority. It would be impossible to stop the virus then. The only way is virus updates over disk and CD or for everyone to format their computers at the same time (unless of course the

virus did that for them later on). Flooding the internet could possibly be more devestating then formating every harddisk, as it would prevent everyone from using the internet, unlike deleting the victims harddisk, which would only affect the infected computers.

Overall, today's society is not adequately prepared for such a well planned attack. With the recent emphasize on terrorists recently, security should be increased. This text was written to encourage the generation of scenarios for anti-virus companies, which haven't appeared to have formed any "what if" scenarios. If anything, Anti-Virus programs are becoming worse, with the introduction of loopholes in their software to increase surveillance. Eventually, these loopholes might be used by viruses, using them to reduce detection.

As you can see, there are many ways to successfully dominate the world with very little coding to be done. Also, there is a huge probability that variants would be released soon after detection by cyberterrorists. The initial programmer might even release a bunch of new viruses a week after the first attack, when the internet would supposibly be returning to normal, to make the internet incredibly dangerous.

NOTE: You must remember that file-size won't matter eventually because of the increase in broadband connections. Resuming technology would also successfully increase the maximum practical file size of such a virus, allowing even computers with slow connections to be eventually infected too. Also, processing power will be greater allowing more powerful viruses. Even the most graphic intensive games these days require 1GHZ max for excellent gameplay, since video cards do most the work. The introduction of 2 GHZ processors recently leaves viruses with ALOT of processing power to play with.

ALICE

An Introduction to Today's Artificial Intelligence

By Actinide

Intro

So I believe we've all seen Wargames, The Matrix, etc. All of these movies who use AI as a main part of the storyline. While today's Artificial Intelligence may not live up to what is portrayed in the movies, it's concept, and the rate at which it has expanded is still impressive.

I'm not going to discuss the actual coding side of AI in this article, being that the idea behind the newest technology alone takes so long to explain. If I get a good response from this, and enough people want me to write another article on the actual coding of bots like ALICE, then I'll hook you fuckers up. For now though, let's just discuss how they work and learn... Kay? Kay.

There are many AI engines out there, some more advanced than others. For our purposes in this article, we are going to be discussing those bots based on ALICE (Artificial Linguistic Internet Computer Entity). If you would like a complete history of ALICE, you can visit http://www.alicebot.org, but for now I'll just give you kids the basics.

History

ALICE was originally based on the same idea as ELIZA (the old psychiatrist bot that gave way to BBS doors such as "Chat With Lisa"), in that it uses language comprehension to both understand input and formulate a sensible reply. Dr Richard Wallace is the brain behind ALICE's conception, as he struggled to prove that a simply coded bot, based on the language comprehenstion of ELIZA, could measure up to, or outperformance other more complex AIs, and do it more quickly.

ALICE was originally written in SETL, which at the time was a relatively unknown language using logic and math to examine input and output. A newly developed language, AIML (or Artificial Intelligence Markup Language, written in an almost identical form to HTML) is now being used in conjunction with Java, C, and C++ to allow ALICE bots to be able to carry on conversations on the Web, IRC, AIM, and other programs such as ICQ. To date, ALICE has won the Loebner Prize not once, but twice, for 2000 and 2001 (for more information on the Loebner Prize and the Turing Test, visit http://www.loebner.net/Prizef/loebner-prize.html).

Comparison of ALICE's brain to other AI technology

Of course the most important aspect of AI is to make the entity believable as one capable of independant thought. In order to do that, it would have to be more advanced than a mimick bot (a bot that takes everything it is told and saves it to a database for later use), which would mean it would have to understand (or give the appearance of understanding) human language and conversation. To best explain this, let's look at some conversations with differently driven bots.

The first is a conversation with a mimick bot:

<Actinide> Hello HotLips, how are you today?
<HotLips4UBaby> fuck you too!
<Actinide> Why do you say that?
<HotLips4UBaby> Why so negative?

As you can see, the conversation makes little sense... And any sense that it does make is a complete coincidence, as it is simply pulling lines other people have said to it from a database.

A conversation with another bot, which uses a crossbreed of math and pre-written responses to formulate it's replies:

BILLY> Hey, my friends call me Billy, and you can too!! What's up?
ACTINIDE> Nothing much, how are you doing today?
BILLY> I'm fine.
ACTINIDE> What is a URL?
BILLY> The definition of a URL is a URL is a web address.
ACTINIDE> Very good, I am proud of you
BILLY> I am a woman, I guess!

Well, we see that he did a much better job than the mimick bot, until that last statement. We can't blame him for being a retard though, he is simply using algorithms and pre-written databases to respond to my input.

And finally a conversation with an ALICE-based bot, who uses true language comprehension, and relies very rarely on databases:

<Actinide> Hello, how are you doing?
<LarLarLoveMeYou> Hi there! I'm doing fine thanks how are you?
<Actinide> I am doing well
<LarLarLoveMeYou> Glad to hear it. What's new?
<Actinide> Nothing really.
<LarLarLoveMeYou> Not a thing? For real.
<Actinide> Can you tell me what 2 plus 2 is?
<LarLarLoveMeYou> 2 plus 2 =4.
<Actinide> Very good!
<LarLarLoveMeYou> Ayuh.
<Actinide> Can you tell me what 5 plus 7 is?
<LarLarLoveMeYou> 10 I think, but I'm not good at math.
<Actinide> You're funny
<LarLarLoveMeYou> Thanks you make me laugh too.
<Actinide> What have you done today?
<LarLarLoveMeYou> Ah. I have been busy growing my mind.

Now, you might think "what self-respecting bot would give an answer like '10' to the question '5+7'?". But look at it from this perspective... That question was fairly simple, and if you were to take a good look at the code for this bot, you would see that she has a good chance of getting simple equations like that right. But say you asked it something like "7+x-2.44464(5+7-9)/54", alot of humans are either going to get that wrong, or tell you to go fuck yourself. Therefore, to be realistic, ALICE bots are going to sometimes do the same thing. The above conversation with the ALICE bot lasted for 30 minutes, before I got bored and gave up... Not once throughout the entire conversation did ALICE say anything that would have tipped me off to the fact that it was a bot, had I not known.

But that is not always the case.

<Actinide> Do you fear?
<LarLarLoveMeYou> I fear nothing.
<Actinide> That is good
<LarLarLoveMeYou> Thanks. Next question?
<Actinide> How do you know who I am?
<LarLarLoveMeYou> This belief is justified.
:: Arches an eyebrow and frowns :: Is that so? So no one ever said that it was perfect, okay?

Even with it's occasional confusing remarks, the above conversations show that it's relatively safe to say that ALICE is at the forefront of AI technology.

Language Comprehension Vs Databasing and Hybrids

Language comprehension, as I said before, is the understanding or appearance of understanding language. Some would argue there is no difference. The reason that language-comprehending AI is so much more advanced that database drive AI is the simple reason that LC bots actually analyze speech patterns.

When a child is first learning how to speak, it's learning process is to associate words with pictures, objects, or even phrases.

The man hands Jade a piece of candy. Jade's mother tells her "Now say thank you". Jade says "Thank you" and the man says "You're welcome". Now, through the rest of her life, Jade will associate "Thank you" with being the phrase you say when someone gives you something, and "You're welcome" as the phrase to respond with to "Thank you".

ALICE bots learn in the same way, associating such things as "You're welcome" and "You are welcome" as being two available responses to "Thank you".

Now Jade is 16, and getting a ride home from school with her boyfriend. Her boyfriend is saying things like "I saw you kissing that motherfucker, you stupid slut". Jade has learned that sentences such as this are negative. Thus, when she is getting out of the car, and he says "Thanks alot for cheating on me", Jade knows that he is not being sincere. She has said things like this before, and gotten the response "You're fucking welcome"... Thus, she looks at her boyfriend and says "You're fucking welcome".

ALICE bots, unfortunately, aren't able to base a person's meaning on the sound of thier voice (IE, they don't have the capability of learning that "Fuck you" is a negative phrase through having heard people only use it with anger in thier voice). Thus, ALICE is coded to understand that certain phrases are negative, which may take away from the "learning" process of

ALICE, and the reality of it, but without that hardcoded information, a phrase such as "Fuck you" may promote a response such as "Thank you". With that already learned information (IE, with that information coded into the bot), a phrase such as "Fuck you" will be responded to with "Why are you so mean?". Likewise, if you were just being mean to the bot, telling it it's a fat skanky whore, and then say "I love you", the bot will respond with something like "No you don't, you're being sarcastic".

Database bots, on the other hand, will not learn the above speech patterns. They will simply respond with a line pulled from a database. To explain this, lets look at a simple database for an mIRC-scripted bot.

You already know the answer you douche bag.
Wow, cool. Ignored.
I'm sorry if he hurt your feelings
Fuck you.
Why do you want it?

Now, if you were to speak to this bot and say "Hey, what's up", all it would do is simply call one of the lines above for the answer.

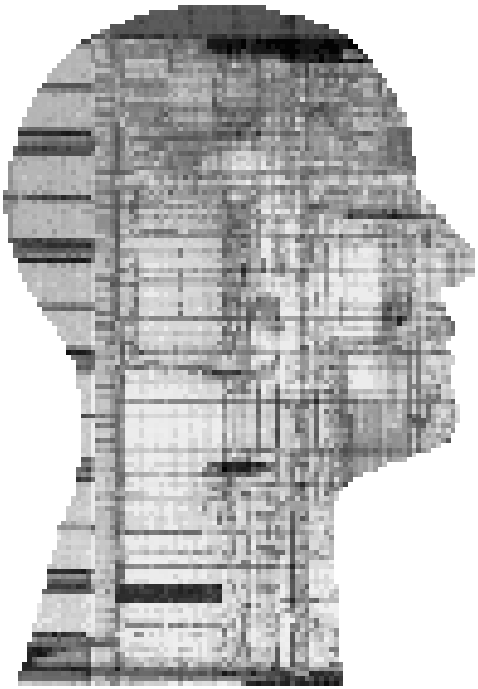
You: What's up
Bot: Why do you want it?

This is a very very simple database bot, but it's the basic idea that all d-base bots are run off of.

Conclusion

There are so many things to talk about when studying AI, especially in the form of ALICE engines, and I wish I could cover it all here. But this text is getting especially long, and I'm sure you kids are tired of indulging yourselves in intelligence. Now, just to make you feel less dirty about learning, everyone go outside, strip, cover yourselves in baby oil, and piss on your neighbor's cat.

All my fucking love,
Actinide@phreak2000.com



An Introduction to Nextel Communications Security

By ^CircuiT^

Nextel Communications, Inc. is the nation's leading provider of fully integrated, all-digital wireless service. The Nextel National Network provides customers with a 4-in-1 business solution: guaranteed all-digital cellular service, Nextel Direct Connect®, Nextel Wireless Web and text/numeric messaging capabilities. Nextel services also feature built-in call security and cloning protection.

Wireless system security is important to both subscribers and carriers in order to protect against unauthorized eavesdropping on wireless conversations and the misuse of service to make fraudulent wireless calls. Historically, the two most common types of abuse in the wireless industry are

(1) various forms of "cloning" or allowing one subscriber device to take on the identity of another legitimate device; and

(2) subscription fraud where a "customer" falsifies pieces of their personal information (address, credit history). The discussion below will not focus on subscription fraud, but will look at the technology available to protect against eavesdropping and "cloning."

The iDEN system (Integrated Dispatch Enhanced Network) has several layers of security that provide protection against both eavesdropping and cloning. At the outset it should be noted that all these layers provide incremental protection over current analog wireless, which is currently the most prone to fraud. With all these layers in place, the iDEN system would be extremely difficult to defraud and is significantly improved over analog celluler.

Nextel Security Features include:

IMEI (Internation Mobile Equipment Identification): 15 Digit identification number of the phone unit itself. (You can switch services from one phone to another by replacing the IMEI number).

IMSI (International Mobile Subscriber Identity): Uniquely identifies the unit on the system it is assigned.

PIN/PUK Codes.
What is it: Two codes that provide additional security for i2000, i2000plus, i85s, and i50sx phones:

PIN (Personal Identification Number): 4 - 8 digit code that must be entered to use the phone (PIN can be disabled by customer).

PUK (PIN Unblocking Key): 8 digit code that must be entered into the phone if the wrong PIN is entered three times in a row. (Only for i2000, i2000plus, i85s, and i50sx model phones)

SIM Card (Subscriber Identity Module) A small memory card, used in GSM Global System for Mobile Copmmunications) phones to hold your phone numbers and other information. Can be removed and inserted into other GSM phones, allowing you to keep your numbers and to place and receive phone calls.

"With Nextel's digital network, based upon Motorola's iDEN technology, when we say no cloning, we're not kidding." Well,we'll see.

Overall, Nextel is extremely resistant to eavesdropping. The radio protocol employed by Nextel through the Motorola iDEN technology is very complex and would require very sophisticated technology and knowledge in order to eavesdrop over the air. iDEN divides a channel into several timeslots, each 15 milliseconds in length. In order to capture usable information, the particular timeslot must be consistently identified and the voice data would need to be separated from all the other overhead information. If a user data stream could be identified given all the preceding, then the actual voice would need to be decoded from the encoding used by iDEN called VSELP. VSELP typically requires 20 million arithmetic instructions per second running continuously to compress and properly decompress user voices.

Cyber Activism May Day 2001

By iDEFENSE

Introduction

Over the past six years, anti-capitalist protests have reemerged as a threat to public order worldwide. Since 1999, this development has been greatly enhanced by the increased use of the Internet as a means of disseminating ideology, coordinating activity and as an offensive medium. Protests of this sort are generally geared around specific events, such as gatherings of organizations concerned with the maintenance of global capital and the symbolic May Day holiday. While last year's May Day protests were marked by a certain degree of Internet organization and street violence, preparations for protests on May 1, 2001, have seen far more activity and a growing awareness by activists that the soft underbelly of capitalism lies in corporate Internet infrastructure.

All manifestations of anti-capitalism — including May Day — are global in nature. However, most physical protests occur in Europe, particularly the UK, and most groups are based in the UK. But cyberspace lacks any boundaries. Cyber protests could spill beyond the UK. The intelligence underpinning this overview is "live," dynamic and constantly updated, with a new version available every few weeks.

Underlying Ideology

One of the many types of protest groups to embrace and employ the new technology as a means of communication, ideological dissemination, recruitment, fundraising and disruption are those that might be labeled "anti-capitalist" and "anti-globalization." Such organizations have been active on the Internet since the early 1990s, and their activities and linkages have increased in sophistication with the passage of time.

The first example of the exploitation of the Internet (at all levels) by such organizations was the day of protest on June 18, 1999 — set to coincide with the meeting of the G8 in Cologne, Germany. A group called J18 coordinated the protests. J18 used a website to call for marches, rallies and online cracking. In London, anti-capitalists marched through the city shouting slogans and engaging in acts of physical vandalism. According to the London Sunday Times, crackers from a variety of nations targeted at least 20 companies, including Barclays Bank and the Stock Exchange. Crackers launched more than 10,000 attacks in five hours. This event marked the first time protestors had made a concerted effort to use the Internet to coordinate actions in the physical world and to launch cyber attacks against global capital.

Since then there have been numerous other examples of the Internet used as a tool for mobilization and offensive action, including the following:

§ World Trade Organization (WTO) conference in Seattle in 1999

§ Meeting of the WTO, World Bank and International Monetary Fund (IMF) in Washington, D.C., during April 2000

§ Meeting of the World Economic Forum in Melbourne during September 2000

§ Meeting of the World Bank and IMF in Prague during September 2000

In many respects, the various activists that participate in May Day and related protests represent a cross section of this broadly defined anti-capitalist/anti-globalization movement. The primary focus of these groups is to fight against corporate power and the state structures that encompass, enable and protect this power. These governmental structures may be in the form of national governments or international organizations such as the WTO, World Bank etc. that coordinate, regulate, finance and enable global economic activity.

Sites/Groups Advocating Online Activities

Discussion of online activism concerning this year's May Day has been confined to chatrooms

and discussion groups. However, advocates of action express similar thoughts and proposed plans of action. This is exemplified by one contributor to a discussion group, who said, "[A] single day of action will not impact on the capitalists' ability to exploit … the only thing these people understand … is the profit margin. Therefore, our best line of attack should be to attack them where it counts most … economically. The best means to do so is to attack the infrastructure of their electronic systems."

Groups that have previously advocated "electronic civil disobedience," such as the Electrohippies, have not sought to become involved in May Day. Most groups prefer to concentrate on more obvious manifestations of global capital such as the WTO meetings and the Free Trade Area of the Americas (FTAA) meeting in Quebec in April 2001.

However, low visibility is not a cause for complacency. The most damaging attacks on corporate Internet infrastructure so far — those carried out against Yahoo, CNN, Amazon and others in February 2000 — were unexpected and showed all too clearly the vulnerability of corporate sites to cyber attack. There is the distinct possibility of a repeat of distributed denial of service (DDoS) attacks or other malicious activity such as the dissemination of viruses, worms or
Trojan horses.

For example, RTMark Inc., a group that aims to raise awareness of corporate activity and highlights workers' issues, has raised the possibility of a "May Day virus." One of the organization's current "tasks" is to investigate the creation of "a 24-hour virus that shuts down the computer on May Day, preferably flashing a message about worker's rights and time off." There is currently no evidence of a specially created May Day virus, although malicious toolkits and generators would make this a relatively easy task.

Sites/Groups Using the Internet to Organize Physical Protests

The following is a list of relevant sites, including the most important ones for May Day in the UK. There are a plethora of protest sites on the web with a high degree of interconnectivity:

May Day 2001 (www.mayday2001.org) This general site displays a banner saying, "shutdown corp. HQs world wide 7am, Tuesday May 2001," and has links to specific locations around the world. The site specifically names McDonalds, Shell, Monsanto, Nestle and Rupert Murdoch's News Corp. The site contains a countdown to May Day and a list of contact sites in different cities.

Operation Dessert Storm (www.dessertstorm.org)
One of the more unusual forms of protest is that explained at this site, which has called for an international month of "pieing" (or "pie-rect" action), from April 1 to May 1. The purpose is to encourage activists to throw pies (hopefully made according to the vegan pie recipes provided at the site) at "faceless leaders of the corporate world, shameful 'journalists,' dodgy politicians and anyone who deserves a face full of dissent."

London Mayday Collective (http://www.maydaymonopoly.net/) A site that uses the template of the Monopoly board game to identify target areas in London with explanations of their significance. Targets consist of government departments, financial institutions and a variety of other organizations and establishments identified as capitalist cronies, class enemies or oppressors of the developing world. The site provides instruction in English, Dutch, French, Spanish, German and Turkish, with a call for volunteers to translate it into other languages. There is also a section on legal advice, in particular section 60 of the Criminal Justice and Public Order Act 1994. While there is no explicit call to arms, and the basic message is one of passive protest, there are serious causes for concern. The May Day Monopoly only lists suggested times and places for group activity. The emphasis lies in "autonomous action," thus removing the potential stabilizing force of centralized leadership. This is combined with the detailed identification of a large number of targets and fears within the protest movement that it has been taken over by "thugs" bent not on legitimate protest but
destruction of property and confrontations with the police.

White Overalls Movement Building Libertarian Effective Struggles (http://www.wombleaction.mrnice.net/) This site represents a group known as the WOMBLES (White Overalls Movement Building Libertarian Effective Struggles) group and is inspired by the Italian "Ya Basta!" movement. The WOMBLES group states that "the White Overall Movement is designed to build effective socialist struggle. We do not believe the present

system can be changed through lobbying or parliamentary democracy." The WOMBLES group also heavily emphasizes autonomy and the egalitarian credo of socialism, saying, "the white overall movement has no leadership; everyone involved participates equally in the organization and actions." On March 11, 2001, the WOMBLES group succeeded in closing business for the day at Niketown at Oxford Circus. The protest was aimed at publicizing Nike's alleged repression of workers' rights and exploitation of child labor.

urban75 (http://www.urban75.org/mayday01) The May Day section of the general Urban 75 activist site. This page contains links to other May Day sites (including the WOMBLES group and May Day Monopoly), legal advice for protesters, information on related actions, background material, discussion forums, "games" and news stories with the promise of continual updates of actions on the day itself. UK Independent Media Center (http://uk.indymedia.org) An independent/alternative media site offering coverage of May Day and all other anticapitalist protests.

WTOAction.Org (http://wtoaction.org) A site specifically set up to protest against the WTO but which has become a general protest/anti-capitalist site. Many links to May Day and other activities around the world.

Protest Net (http://protest.net) A detailed and general international protest site with a link to all May Day events. Also covers many other issue areas familiar to the anti-capitalist milieu, ranging from animal rights to religion and spirituality.

Conclusion

While there are certainly links and commonalties between those allied against the forces of global capital, one of the movement's defining features is the absence of a clearly established organization. This makes the activities of these groups both difficult to monitor and predict. Furthermore, this type of cyber activism is enabled by an increasingly efficient telecommunications infrastructure.

Physical violence inspired by a variety of websites poses the greatest danger on May Day. The numerous and diverse array of targets suggested by the May Day Monopoly board will pose a serious strain on police resources, particularly if violence is widespread. The probability of violence may be greater this year because of reports that the "spikies" (militants) have taken control of the May Day movement from the "fluffies" (peaceful protesters). This situation could be replicated in cyberspace, where the ideas of electronic civil disobedience are replaced by more malicious activity.

While physical protest is certainly dramatic, it is generally confined to particular areas and is (usually) swiftly contained. Cyber violence, on the other hand, can have far more lingering and widespread effects. There is little evidence so far of a single widespread, planned electronic disruption (only one organization has openly sought interest in developing a May Day virus), but that is the nature of economic and social life on the infrastructure. It cannot be ruled out.

Because of the ideological underpinnings of the May Day protest, the range of potential targets is extremely broad (see for example the corporations, government departments and other organizations listed on the May Day Monopoly site). In terms of electronic infrastructure, companies and organizations wishing to remain in business would be strongly advised to rehearse and strengthen their preventative means and disaster recovery plans over the May Day weekend, then keep them current and a major feature of their business strategies. Infrastructure protest and cyber violence are now well established, developing swiftly and here to stay.

CONTACT
Jerry Irvine,
Corporate Communications Director
Voice: 703-219-2419
Mobile: 703-622-3058
E-mail: jirvine@idefense.com

iDEFENSE
3975 Fair Ridge Drive
Fourth Floor, North Lobby
Fairfax, Va. 22033-2924
http://www.idefense.com

Unorthodox Bug Finding Techniques

By the Pull

There aren't a lot of papers on this, and there are a few things I have learned from working and playing in the security industry about how to find bugs. A big hesitancy to writing such a paper is the danger of disclosing one's "secrets"... and thereby allowing your "competitor's" to get the jump on you.

However, I am opposed to that kind of thinking. Full Disclosure is about being open and honest - as is Open Source - furthermore, competition is good for the overall security of the industry. Beyond this the information is out there... and there is still the equation of hard work and strict discipline as well as having an open mind.

This article is not "how the Pull sees it" so much as "how much the Pull has gained from reading and following the great bug finders". So, I am confident in this material not because I have found it to help me. But, I am confident in the abilities of the people I have followed and my own ability to look at things the way they do.

As for who I am, I would like to say "that is unimportant", because on one hand we need to judge material by their content, not their appearances. However, there are a lot of people that have ideas on all manner of subjects that should not be writing on these subjects. ie. the old "what fruit does the seed bear".

I have put that at the bottom of this paper.

That said, into the good stuff:

First, some basic philosophies to consider at all times:

1. Nothing is secure. Nobody is perfect.

This is one thing. And, it is perhaps the most important thing to reflect on and believe at all times in bug finding.

You often hear from successful athletes and such the saying, "believe in yourself". I find that to be obscure. In this, you want to believe that there is a hole somewhere - someway - it just hasn't been found, yet. You also do not want to believe the ground has ever been completely scoured by other bug finders.

Regardless of how difficult the problem is, there is a way to solve it. This is because software has so many places to go wrong. Really, software is in its' infancy still, as is bug finding. Yet, it is not so young as to be mired in many traditions.

Some traditions, for instance, are such as, "all bugs have to be buffer overflows", or "most bugs will be buffer overflows". As far as we know - to this date - most security bugs tend to be buffer overflows. Those are just the found bugs.

This is the attitude one should have at all times. Resisting this attitude will kill your efforts. After spending thirty hours with no luck looking into an application, that last five minutes may be the bug. But, if you have given up saying, "It is impossible,
it is secure", then that is a bug you have not found.

Developers are far from perfect just as bug finders are far from perfect, just as the software is far from perfect. In thirty years - if we last that long - just consider how antiquated our software today will look. It is far from perfect. It doesn't matter who developed it nor how many.

2. Shortcuts are the rule, laziness is the exception.

You don't become a martial arts expert by not training just like you don't become a good shot without practicing. These things are also security related paradigms and they are applicable to computer security totally.

The ICQ buffer overflow I found took seven hours. The document.write() bug I found almost instantly when I tested the method. The arbitrary file run method was found in about two hours after deciding to test the pop-up object.

However, all of these things required a great deal of prior research. I read and re-read everything on the subjects I could. With the IE bugs I was continuously testing various things in IE and poring through previous published bug reports to garner methods of exploit.

It is a grueling but exciting process. For me, I keep copious notes and try to stay as rigidly methodical as possible -- though, due to the extreme boredom, I often switch gears and stop what I am doing to immediately test something else out.

Like in all things computer related, there isn't any kind of intelligence level needed for anything. The whole "IQ" concept is a myth. It is a matter of will and desire. This means you must watch your desire at all times. If not properly motivated or rested you will burn out -- just like in sports. You must pace yourself and slowly build up patience and desire.

Everything that can be fuel to the fire, let it be. Everything that can motivate you to continue, let it be. Not doing the work gets you no results. Doing the work without fail always brings positive results.

3. Edisonian Techniques Versus Teslan Techniques

Edisonian techniques I call the grueling method of trying everything just like Edison worked. It is heavy on empirical science, light on theory. Teslan techniques I call being heavy on theory, light on empirical science -- but not so light as to distance oneself from it.

It is good - I will put forth - to try and manage the two ways. In bug finding if you get too far from empirical testing you will never get anywhere except nowhere.

But, why should anyone ignore the theory? Research breeds inspiration. Research should go hand in hand with empirical methods.

This means try everything. Bugs are not documented. Bugs that have not been found yet... they don't exist. They could be anything. But, you have to know what to look for. This means a lot of research. You have to know the technology you are attempting to find flaws in. You have to know the rules.

With buffer overflows, the rules are pretty simple. With browsers, some of the rules can be pretty obscure. You have to keep your goals in mind. What do you want to do? You want to run code on the system. You want to drop a file. You want to run commands. You want to read directories. You want to be able to access files outside of your real domain. Etc, etc.

4. Humility

This means you make mistakes. The more you are aware that you make mistakes the better able you are to realize that the ground you are trying to cover has not been covered yet -- even though you have spent fifty hours looking over it.

Before you find a bug... few things could be more thankless than bug finding. It is like going through spaghetti code backwards five times over and worse. It is obscure. It is obtuse. It is grueling. It is thinking and going backwards when you should be going forwards, crawling when you could be walking, deconstructing in order to construct.

Another point to this is if you ever start believing you have some great gift or a "special something" you may start to believe so does everyone else. This leads to tradition. Tradition is a big enemy. Tradition is temporary and gets old.

When no one else is finding bugs, you won't either -- if you are both following and believing the same traditions. If you make yourself an enemy to tradition you will find bugs and be renewed even though it seems like you have covered the same ground a million times.

5. Pride

On the other hand, you have to fuel the flames of desire. You should take pride in your fellow bug finders and the reports that they write... as well as in your chosen field. It is the ultimate expression of computer security. It demands honor and deserves the hours you spend in it.

When you find a root security bug you have often found a golden key into the systems of the world. If you are good, you give that up and report it to the proper authorities. If you are bad you steal a bunch of money and move to the Bahamas.

I say that because it is real, a very real thing I have seen very much in the security industry. It is a huge responsibility. It is extremely thrilling to know what you could do but do not do. I am just being completely honest. The same thing would be true if you could strike people down with thunder, but don't.

The same thing is true in physical security, though. There are a great many martial artists that could do some serious damage to a great many people at anytime. Very few martial artists ever go on a

rampage. This is the honor of battle. Computer security is in the same paradigm.

6. Desire

I have already spoken of desire. It is the same in anything. If you wish to be the best at whatever you are doing, you need disciplined desire.

You don't get started without desire, but the desire you have when you get started is nowhere near the desire you end up with as you combine that with discipline.

Desire, psychologically, demands not only discipline, but rest as well. Rest is demanded so you do not burn out, and when you rest you get a chance to slingshot your efforts further.

Another word for this is positive thinking. Nobody desires something that is nothing. Desire demands imagination. Desire demands creativity.

7. Knowledge

Look for shortcuts, but understand that you have to get the knowledge.

Technical stuff can be very dry and boring. You have to find ways to make it interesting. Be creative. Look for disdained knowledge sources. Look at respectable knowledge sources.

Remember, knowledge is nothing but fluff without wisdom. Many people don't even know the difference between the two. Wisdom is weighing the values and rightly ordering the knowledge you do have. So, seek your knowledge with everything you have.

Work from past bugs, without a doubt, there is no skipping of that. You will often "microtask" in security, in technology. You will forget a lot of the bigger picture. Just remember that.

Now, for the next section, pointers on how to find bugs:

A Few Preliminary Notes

It is impossible to cover the technology of every application and what can go wrong. Personally, I prefer technology that is used by the most people and that can traverse firewalls -- such as Instant Messengers, P2P applications, and Browsers. But, preferences are transitory.

The number one thing you want to do in starting on an application is to research past bugs found. You probably want to test them. To understand them.

With IE, I set up a server and make sure everything is set to default. It is very easy otherwise to find "bugs" that aren't bugs at all. I have one paper for notes and collect everything I can.

Often times I will try something, it will do something, I will copy the code into my notes, then try and break it down to the essentials. I always try and keep my code as short and sweet as possible.

There is a good joke on the internet about a progression of a developer from a beginner to an advanced hacker through the way he writes "Hello
World". In the middle his code is extremely long and convoluted. In the end, at his mastery, his code is back to the beginning -- but even simpler than
at first.

With bug finding you often get into writing some very obtuse stuff. The problem there is that you may get lost as to what the bug actually is. You have to prove your case to yourself before you prove it to others.

Bug finding is all about evidence. You have countless hypothesis' and want to prove them. I generally write down my hypothesis' in my notes as I go along. The more I have off my mind at the time, the more I can concentrate on the task at hand.

You must know how to isolate your evidence. This is essential to all QA. You must understand the reason for what is happening -- as well as you can. (Or, it could not be happening at all, for one thing).

In testing, you want to test every method of every object that is accessible. Since you are testing things that should not be very often - mostly - you are almost never done. It depends on the complexity of the method. It is best for humans to switch around. Often you will want to test a certain method of exploit on many methods rather than try and think of every exploit possible on every method in sequential order.

A Bit About Testing And QA As It Relates to Security

A lot of hackers are completely removed from the concept of "QA". I first heard about QA as it relates to quality in computer sales about ten or so years ago from a QA at a bottle processing plant. Some of the things this guy did was, for instance, audit the expenditures of the company to find unneccesary problems -- in one instance he cited, they were renting a trashcan for about 900 a month and saved quite a bit of money by buying it.

Quality was a big component of these past century's commercial revolutions. We discovered how to do assembly line type production and found that by studying and experimenting we could vastly improve the way things were done in order to save money. The end result of this kind of work was a vastly improved standard of living for the world.

In software, Quality Assurance is a big field. It often tends to be very stuffy and rigid itself. A great deal of QA departments across the world depend apon software testing methods. They are very often several degrees removed from the developer. I have heard that some departments, their developers do not even know their QA nor do they ever see them. Many companies do not even have QA.

In short, what this means is that software is not properly tested at all. And, software needs every line tested, every method.

This said, there are a few factors here which I see influencing security researchers. Some of these I have already gone into. But, the point I want to make right here without going vastly into the details - you can just keep your eyes open for them in your research - is that there really are not that many security researchers finding exploitable problems.

This is rather shocking, really, but there are some good reasons for this. First of all, note sometime as to how many security exploitable bugs are actually found in software each year.

When looking at the below data, consider: how many software projects are actually out there; how many researchers find more than one bug; how many times you see a vendor release an update that includes a security bug; how many large the software products are that are out there; how old this business is compared to say, making weapons or cars or watches; lastly consider the rarity of big bugs you hear about on the local news:

SecurityFocus, has a nice stats page:
http://www.securityfocus.com/vulns/stats.shtml

Month by month, year by year, the states have steadily increased from 1998 from under fifty vulnerabilities found a month to almost a hundred and fifty.

On Linux (all), the numbers go from 14 in 97, to 25 in 98, to 99 in 99, to 153 in 00, to 96 in 01. On Windows (all), from 13 in 97, to 9 in 98, to 124 in 99, to 137 in 00, to 56 in 01.

The numbers themselves should be taken with a grain of salt, of course, though it a good rule of thumb. It does not mean Windows is more secure than Linux - many Linux security bugs involve surmounting basic security features Windows doesn't even have, and there are many other notes, but this is not the scope of this article.

The point here is that this number of security bugs is nothing compared to the potential problems in the software out there. Perhaps one of the biggest reasons for this is that security researchers that find these bugs very often get hired to work in other fields... and almost always have a great deal of work to do which is not for free.

Even security companies that have teams working on these things - this isn't the only thing that they do. Banks, are a big customer and always have been, for these types of services. Many software companies see the need for outside security QA work and very often hire these security researchers on behalf of these companies.

To give an idea about how QA works, what sort of numbers you are talking about, at the company I worked with we have perhaps a hundred to five hundred bugs a month on applications (no duplicates). Mozilla, who keeps an open bugs database, is a good example, today - a Saturdary - they have listed about thirty new bugs just for the day. Likely, there will be about ten of these as duplicates.

Furthermore, there are not just "duplicates" but "related" types of bugs. If one bug is fixed this may fix five, ten, a hundred other bugs. So, 56 security bugs this year in Windows, 96 across Linux platforms -- there is a far way to go.

Lastly, though I pointed out how distant QA usually is from development and how they often rely on automated testing applications... I did not stress just how ill-prepared they can be for looking for security bugs, which are in a class of themselves. Many of these QA - if not most - do not know how to develop, and they surely do not follow bugtraqs.

I will use my "document.open()" bug in Internet Explorer as an example. Microsoft has 3 QA to every developer as I well know because my last company had a Microsoft guy in our Venture Capital group. "Document.open()" is a method used in conjunction with "document.write()". Extremely common javascript method used by perhaps millions of websites. The method has been out there for years.

The bug, itself, does not use any special tricks. It is being used properly, as suggested in the first paragraph of the reference papers. The only improper thing about the bug is that it just so happens you can use the method to open outside websites or local files and write to that window. To people whom do not follow web security problems this may mean very little. I suppose Javascript developers who are aware of security limitations in their language either just never tried the method in this manner or they just never thought twice about it if they did do so. They are not by definition, security researchers, but developers.

In reality, this means I can steal cookies from users, read local files remotely, and spoof websites. An example I wrote uses regular HTML - no activescripting - to hide the url of the target system, say "www.malicious.com/spoofScript.html". The end user sees in their mail from, say Microsoft, that they need to pick up an important download. They click on the link, and lo and behold, there is the official Microsoft download site with the official address and content in the browser.

Only, the executable for download is a trojan.

This past through three QA per developer, it passed the eyes of a lot of security researchers, it past through Microsoft's security team, it past through every javascript developer that ever used the method, and it did so for a few years.

It took me roughly thirteen days of looking, minus days off and plenty of "doing of other stuff". Yet, there are countless stories like this.

A Short Note on Buffer Overflows

Buffer overflows, the most common security vulnerability found today are extremely common in code written in C and C++. They can be prevented by simply using buffer overflow safe techniques in the code which include using methods that are "safe" and doing buffer checking. When I was a QA Lead we went through all of our C code with the developer's environment and searched out every dangerous method. These things were all fixed within a week.

Strangely, companies very rarely do this. It helped that at our company we had some of the very brightest engineers out there and that we had a strong security bent to our company.

There is, without a doubt, a great deal of C and C++ code out there. Not a few languages are written in these languages, such as Python and many interpreters. Both Windows and Linux are written in these languages.

In Internet Explorer, you have, for instance, the application written in these languages and on top of that you have the languages it hosts written in it, as well as the operating environment and the activex it can run: which typically are about a hundred or so.

There are a lot of great papers on Buffer Overflows, such as Aleph1's, Dildog's, Mnemonix's, Dark Spyrit's, Mixter's, w00w00's... so I won't go into what they are. I would suggest following a great deal of cracking tutorials to get a grasp of assembly in easy to understand terms and picking up a book or two on the subject before dwelving into these papers.

Buffer overflow exploits are rightly stated as being the "vulnerability of the decade", starting with their largest public notice in the late eighties with Robert Morris' worm that took advantage of one. "By far" many places point out, "buffer overflows are the most common vulnerability found".

On this note, I will point out that buffer overflows, though, should be something every security researcher should know how to find... but, they should always be aware that with everyone looking for them, there remains a great deal of bugs out there they have not looked for. And, there remains a great deal of bug theories not even presented nor discovered yet.

I have said this twice now, because it is crucial to thinking correctly about how to advance in bug finding. This is primarily because the field has so many overflows out there, so many left to find, so many found... but, that can be an overwhelming influence in any field. Computer security is relatively in its' infancy.

Some Testing Methologies and Examination of Other Security Bugs

Every application has it's own sets of security rules, as I have already stated, and it is important to know what those existing security rules are... as well as keeping an ever open eye to new security rules. Because if the developers are unaware of new security rules the should not be breaking, of course, they will be breaking them left and right.

This principle is especially true with the more obscure rules, as it has been shown to be amply true even with basic buffer overflow prevention rules.

It is very difficult to learn to develop, comparative to say, learning how to using a computer. It is not for everyone. It requires a great deal of patience, a great of deal of effort, and a great deal of going over very boring details.

That said, developers generally are not aware or kept up on all of the latest security problems - often not even the ancient security problems - just as dentists aren't. It simply is not their field and it is not often enough within their goals. Even when it is within their goals, security research is an entirely different field with a great deal of ground to cover.

In server applications, you have two different attack mindsets. On one hand, what is the type of content being served? Can Perl applications be run on the server? Can VB applications be run on the server? Then, the applications that are run through the server need to be tested for security holes. eg, cgi script problems.

On the other hand, not everyone running a server application will neccesarily be running a cgi application, nor the same one. So, there is the examination of how the server processes command with default install. For instance, 'directory traversal bugs' are a big one.

I have found numerous directory traversal bugs. They generally take a few hours at most to begin with. Then, you have every place where input might be accepted remotely to test for these vulnerabilities. For instance, you have places where remote users can and are supposed to put in information, then you have places where the server accepts commands from applications which can be tested and exploited manually.

Directory traversal bugs span from being able to get files outside of the local "root" directory which is the directory that is supposed to be served to directories that are not supposed to be served.

Sometimes these bugs have been worked out by using the "dot dot slash" method, ie, using a method of traversing directories without naming the directory directly. This, of course, is seen on DOS or Unix command shells as "cd ../../" or "cd ..\..\".

Methods that have been found to work as well which have escaped server's developer's and QA notice include using ASCII or javascript shorthand for dot's and slashes as well as doing this method using multiple methods of representing that ASCII (though the javascript method of using &#code is one of these methods).

One of the latest of these vulnerabilities allowed using unicode representations of this method of directory traversal on Microsoft's web server which actually allowed running code on the command line with parameters, thereby giving remote users immediate root.

Lastly, while directory traversal bugs are generally found in server applications, they are starting to be found more and more in client applications. (Moreso, I am sure, if people would look).

In discussing this I have just presented a few guidelines to keep aware of:

a. Old bugs can have great signifigance on new bugs
b. These directory traversal methods involved using command line, core OS techniques to places where they should not be able to be used. Keep this in mind.
c. If there are different ways to do the same thing, likely one of more of those ways are protected against by the application, but less likely all of them. There may always be yet found ways to do the same thing. d. If there are different places to do the same thing likely one or more of those

places are protected against by the application, but less likely all of them. There may always be yet found new places to do the same thing.

The variables I am leaving out here is "what can be done" and "where can it be done". As for "what can be done", that is a wide field and tends to be application type specific - say, server, client, then say, email client versus web client - and further, application specific itself, say, for instance, Internet Explorer is very different from Netscape Navigator while IIS is very different from Apache.

Personally, I have found directory traversals in a p2p application as part of my job and in Internet Explorer. In the p2p application where I found the directory traversal problems I found it in the place where web content is served as well as in various places where input is accept remotely for serving hosted content. In Internet Explorer, a hosted object allowed directory traversal using the file://[clsid] type of url (a type of url I discovered, though used internally without documentation in Internet Explorer though used in the Windows
OS very often without the file:// protocol url -- just as dot dot slash is used locally but should not be able to be used with protocol url's).

By using that type of url you are able to browse to system folders and access system files, which allow the remote user to parse that directory path remotely and access senstive files. By using that type of URL in co-ordination with dot dot slashes you are able to browse into the index.dat file which contains the temporary internet file names -- which allow a remote user to run trojans instantly on someone's system via HTML's hosting languages.

A Look At Breaking the Rules

There are a great many rules which can be broken. Despite the title, a nice book on the subject is "Hacking Exposed", however you can find comprehensive breakdowns of security rules all over the net.

When you look at security rules for applications and such, you should bear in mind that bugs are often found which transcend security rules and have no theory written about them which is compressed for mass consumption -- yet. And, maybe never if no one studies the bug.

The example I am going to use here is with web browsers. In web browsers there is a rule that you do not want to be able to read the contents from one window or frame from another -- if that content is from a different domain. That domain could be local on the user's system or it could be remote.

Right off, this is something ever web developer has to deal with and get used to as the limitations in the browsers they write for and the languages they write in cover these security limitations. Often if not usually, they find out about these limitations,
though, by having some bright idea in mind, trying it -- and then posting to a newsgroup about why it doesn't work (or they look it up, or ask a friend).

Right off, this security rule is rather vague and obscure. If you have never heard of it before you are probably wondering why this sort of thing would
be so bad. After all, how many times have you seen a webpage opened from a website from another domain? All the time, because, for one, that is where
pop-up ads generally come from on websites: another domain.

Furthermore, as a security researcher you generally only first hear of these rules as vaguely as I wrote about it above. As human beings we do not process how dangerous these things are or important things are... until we have seen with our own eyes proof. Or, until we have studied the subject, and become persuaded by the principles involved.

This means that the QA and the developers will likely not understand the rules themselves - and even if they do - their peers may not. The 'peer factor' is crucial in testing and development. As the saying goes, 'a single strand of rope is not as strong as two strands'. Your typical rope has perhaps hundreds of strands. The same paradigm exists in software development and release.

As for the rule under discussion, the security problems with it are multiplefold and it is an extremely common and dangerous type of bug for some of the following reasons...

If activescripting (ie, javascript, vbscript in a webpage) is allowed to interact with content from another domain this surmounts the security structure of domains. For instance, the local computer system is a domain. The intranet is a domain. CNN.com, Yahoo.com,

Microsoft.com, your bank.com, your isp.com -- these are all domains. Obviously, code that is run in your local system domain is run with far higher privileges than something run in a remote, non-trusted website domain.

Usually, local system HTML content has the ability to write files, run applications, pass parameters to these applications, install remote applications... and the like.

Therefore, generally if you can do "cross site scripting" as this type of bug allows, you are able to run local code remotely in the context of local code -- and therefore, you may be able to root the system immediately.

The checks done on this sort of thing are very wide reaching across the browser or client system. Every check is important because being able to surmount one of these checks may give remote, instant root. Being able to get around one check, but not others can still do this at times -- at other times it is like one pin of a lock mechanism picked. There are still more pins to be picked.

Sometimes, you can not access the local system doman, but you can still access the domain of remote, internet systems. This can allow you to change the content of that page, still, called "spoofing websites". Or, it could allow you to do other things like getting mail from webmail accounts.

In the case of Windows and Internet Explorer there is a feature called "trusted sites". AIM and AOL install "free.aol.com" automatically into "trusted sites". Being able to access not the local system, but this site, allows you to run code locally on the target's system, which allows root.

Lastly, a common technique used in this sort of thing is that you are able to grab people's cookies from remote sites. This is very bad, as cookies are often used for authentication to some serious systems and they often contain private data such as: credit card numbers (pretty rare now), usernames, user passwords (getting more rare), real names, addresses, and other such data you wouldn't want the world to know.

So, that was a good, extreme example of a security rule which an application can have. This may seem to be a client side only problem, but cross site scripting can also be used on server's javascript code to attack remote users through maliciously formed url's... through malicious posts... and the like.

As far as rules go... that is a pretty good coverage of what types of rules are out there, though, of course, there are many. At this point this leads me to the "targets" section, which expounds on "security rules" from a different angle.

KNOWING THE TARGET

What are you trying to do? Find security bugs? That is too general, useful for conversation, but too general in actual application. You have to have a goal, specific goals when working towards anything or you will be working in an aimless direction.

That said, here are some targets to bear in mind with just about any security bug finding goal:

a> put a file on a system remotely
b> run a file on a system remotely
c> run a local command remotely (eg, "format" or "ftp")
d> inject code into the Instruction Processor path to run (eg, overflow exploits)
e> read a local file on a system (eg, sending password file out remotely or reading index.dat remotely, etc, etc)

These are just some possible goals. Never think they are all of them.

In some cases your goal will be more exact. For instance, with browsers you have the "security" of a download window. In many circumstances with Internet Explorer that just means a user has to hit "open" button instead of "close" or "save".

A very sublime way of handling this sort of dialog was not long ago seen by Dildog's activex exploit which worked through office documents. The activex could actually be commanded to push the buttons so as to lower the security settings and enable it to write to file.

Another interesting, albeit far less sublime method of getting around this sort of "security" dialog is to be able to put a graphic that looks like another dialog on top of it so that when the user hits "open" they think they are hitting to "open" something else entirely... or even the text could be obscured so they think they are closing the window or anything else.

This noted, while many security bugs are "sublime" in construct, one should never consider a "sublime" bug that doesn't give local root to be anything other than what it does do. Simple security bugs are just as hard to find and can very often give root.

There are often proprietary rules that should be followed but may not be with applications. For instance, with a p2p application you can download files, but can you force the application to immediately open files?

Then, there are crazier methods, like activex, which depends apon a registry entry for that activex which states whether or not it is "safe for scripting" or it depends apon whether or not it puts in "IObjectSafety". The "or" is an important difference here because listing the objects available for

Furthermore, you have the whole signing dilemma. Signing just means that your system recognizes the application - an activex or java applet - as trusted... because some guy paid seventy bucks to a trusted signer whom never even looked at the code.

Needless to say, these components still have a prompt, but in some circumstances do not show a prompt at all because they are "trusted". For instance, running one in Local Computer, they will install without a prompt.

With scripting languages server-side, such as cgi-scripts, you have nearly an endless supply of problems. I could not begin to cover those possible problems and people such as Rain Forest Puppy have some excellent papers on that.

Languages like Perl, while extensible, make it very easy for careless developers to leave gaping holes in their code... anytime you have the ability to run code remotely on a system, there is a huge chance there can be a problem.

Lastly, when I look at what code I want to hack I consider various important factors, though not always as sometimes there may be a specific target:

a> How many people use the software? Some software installed on Linux is installed default across all platforms, for instance, while Windows installs default applications immediately. Other applications, like AIM or some p2p software is used by hundreds of millions of people.
b> Can the application be accessed remotely?
c> Can a firewall stop the attack by default? For instance, web browsing or Instant Messenging traverses firewalls.

A Brief Look at Target Goals

Okay, just ran through these in the previous section, now some more examples:

a> put a file on a system remotely

In some cases you can overwrite local files that are run automatically thereby gaining root. In other cases you can put a file in a place where it can be run automatically, such as start up folders. And, in yet other circumstances you may have a method to run that file remotely already, such as a codebase activex call or merely using a HTML refresh.

But, that is all technical details, unimportant, really, until you get to writing the exploit.

b> run a file on a system remotely

Really, this is very similiar to c. However, what I mean by this, is for instance, being able to run a file on the system when that file is located remotely.

For instance, you have a trojan on a webserver and people can open that trojan remotely if it were not for those pesky security dialogs. How to get around those...

c> run a local command remotely (eg, "format" or "ftp")

Very often you can get out of applications frameworks and inject system commands in there. For instance, many cgi's have been known to be able to accept commands if you shove them in right in the input for the cgi which is accessible remotely. The unicode directory traversal problem allowed this in a different way, as discussed further above.

d> inject code into the Instruction Processor path to run (eg, overflow exploits)

This doesn't need explanation, but is what happens with overflow exploits. Note the similarity to some of these other ones.

e> read a local file on a system (eg, sending password file out remotely or reading index.dat remotely, etc, etc)

I cover this in the paranthesis.

The Attitude

I take some of my attitude I learned from sales when looking for bugs. You try, you try, you try again. That you fail means only you are one step closer to success. The more you try the more chance you have to succeed.

In sales it was shown that if you, say, knock on five hundred doors, you will get one sale. (Yes, I have even done that kind of sales, what can I say). So, you knock on five hundred doors, you get one sale. You knock on fifty doors, you will never get a sale.

This is how it works.

As you get better you can make that from five hundred doors to one hundred doors to maybe even twenty five doors... though don't think there is any kind of magic involved.

Bug Research Tips

A great deal of bug finding, like development, involves research. When targetting an application you must, one, understand the internals of that application, and two, know the past security bugs with that application.

Sites like securityfocus and google are essential to this. With google, as with all search engines you can look in the "advance" or "more options" section to see how to search a specific site for specific words. This is often faster and more effective than the site's search engine. Often times, a site won't even have a search engine or it will be very innaccurate or worse, not covering the entire site.

You want to understand the application inside and out, though you can certainlly begin your hunting and look up your questions as you go. Since you may not be asking the right questions - never assume you are - you want to do a lot of searching and reading.

With applications you want to search on types of bugs found in that type of application as well. For instance, directory traversal bugs are common to server applications, so you want to search out bugs found in other server applications. Limiting yourself in this would be a bad idea, doing something like reading every bugtraq post ever made -- this would be good, though grueling.

Regardless, we all forget things, so you will end up poring over many papers.

Whenever you find something suspicious, mark it in your notes, and you probably will want to research the problem as well to learn everything about it.

Open bugs databases can be useful, and the Usenet has an invaluable search capability in groups.google.com.

Some search tips... when searching, often specific questions are useful. Try and put yourself into the mind of someone whom might have posted something useful about your problem. Much searching is narrowing down the problem,
and this can be very challenging at times. Try to think of the most unusual expression of your problem possible so as to avoid a lot of junk.

Always make and keep notes, book mark sites you think may help later.

You will be backtracking a great deal.

And, lastly, just because you have researched every bug ever found in a server product and tried them in your application... don't think you can't try client bugs as well. And, don't think you have tried everyway to do old bugs in news ways and new places.

Proving Your Bug - Exploit Code

When you have a security bug or think you do, you have to write an exploit for it that shows the bug is real.

Long before you ever send a bug to bugtraq you will have tried to prove countless bugs that on further examination were not bugs at all. In many ways, bug finding and writing the exploit code is one and the same.

For instance, with IE, I set up a local server and view my new code. Nothing. I change something, try it again. Over and over again. I research, find something interesting, put it in notes, try again. Maybe something suspicious happens. I copy the code to my notes. I mess with it, research it.

Not a few times I was certain I had exploit code, only to discover my testing environment was not

default or I made somesort of mistake. This happens. It actually happens all the time in QA. It happens to the best of us and the worst of us.

Many times I have received emails from people with code saying, "Why doesn't this work". I just kind of smile because I have tried thousands apon thousands of code that doesn't do anything, I don't know why it didn't work... except that the vendor did their job.

One of my most famous applications, the "godmessage" was exploit code from Guninski's script lib bug exploit. Guninski is one of my favorite bug finders -- he always finds original, clever ... sometimes diabolical stuff...

My problem was I had a co-worker that was being harrassed by a drug dealer - I didn't know the coworker - but he heard I knew security and asked for me to help. The problem was all he had was a email address. It so happened Guninski had just released this script lib bug which allowed writing a file to the startup directory.

After I solved the problem - I used many, many different methods that did not work at first - I decided I didn't know this guy very well, but I put the stuff on full disclosure.

This application started from something that downloaded via ftp to being something that could work on all Window's operating systems and quickly and quietly write a file and run it to the system. Optimization was worked out through the help of several brilliant coders and thousands of feedback emails.

But, all along, it was obvious from Guninski's little script that shot up "hello" that this sort of thing could be done.

So, I don't write full out exploits for my bugs very often because of the sheer time required to make it work all the time on every system. You also should strive for bare essential exploits that prove your bug.

On the other hand, I sometimes have seen a general security principle which was a problem and found it neccesary to write an application for that exploit. In that case, you don't even have a literal security bug.

For instance, with WinApocalypse, I saw that all ddos agents were in Unix. This was before the Yahoo/Ebay attacks. There was tfn2k but it had bugs which prevented it from being run on Windows and needed a rewrite. Moreso, though was the idea that all ddos agents needed hefty encryption, hefty control command sets from the remote user, and hefty DoS techniques. I saw that if you had a million systems rooted like I could do easily with the godmessage -- you would want to use a timed fuse method and the actual DoS method was inconsequential... a million systems attacking one system would be catastrophic regardless of the DoS method. I also stated this sort of delivery system would best be used with a worm function.

This timed fuse method was later used in conjunction with worm spreading abilities in the Code Red worm... which I had written about a few years before and proved in the paper why such a thing would be inevitable.

The group, 29a, one of the most prolific virus writing groups in the country regularly publishes seminal virus papers with their virus open source in an ezine. I don't know why they do it - their articles
are some of the most brilliant out there - but, obviously, I did what I did, publishing it to a major security website... because I felt people should be prepared.

With both of those exploit sample codes I had in my possession code that literally could have infected 86% of the world's browsers and dropped zombies to attack whomever I wished. Or, I could have done, really, anything I wanted to. Especially because this was before KAK or BubbleBoy. I could have hacked a website and hacked every user that visited there. I could have hacked newsgroups and targetted users I did not like. I even had a version that immediately wiped the hard drive apon viewing the HTML.

So, exploit writing is fun, no doubt about it. Though, I curb it because I saw the abuses of KAK and Code Red and the Yahoo attack after I wrote these things. This is a wise idea for anyone.

But, let me give you two basic reasons before moving on:

1. I could have seriously hurt the world's economic condition had I written code that could be catastrophic. That may seem all nice and fine with some exploit abusers but the simple fact is that this would hurt the poor worst of all re-

mains. The poor receive their food and shelter from the same economy that feeds and maintains the rich. You can't hurt one without the other. This may seem rather cold, after all why would I want to hurt the rich? Who am I to judge? But, that is what might happen had I released this even to a security website and someone took that infected people.

2. Hurting systems and hacking websites doesn't make anyone a scary person. I weightlift to scare people, honestly. I work out to be more intimidating, and so that, if I have to ever prove myself, I can do some serious damage. So, if you think you want to be a "bad ass", think again. Real bad asses have muscles bulging out everywhere, or carry guns. There is a lot of glamour to fear, but no one really fears someone that can just change their webpage or delete their hard drive.

Exploit Writing - Case in Point

With the document.open() bug, I received a lot of press. One article titled the article "Microsoft investigating Alleged Bug". All of the articles I wrote were well written by the press and I believe helped push Microsoft to go harder towards security.

Microsoft's statement was they were investigating an "alleged bug". Clearly the demonstrations I showed were obscure but showed that they were not "alleged". It does put them in a big bind because it is such a common method and was released just after they put out a fix for the browser. They look good or bad depending apon how many fixes they release, I believe, so they do this sort of thing.

I had the spoofing example, it showed www.chase.com in the address bar, and I explained that I could put in there authentic looking content (but didn't for content reasons). I had seen spoofing examples before, and this one was along that lines.

However, I wondered what could really be done with this, and I was asked by this by reporters. I then figured out the worst possible usage of website spoofing. It was previously just a technical detail. By this time I had already attained my goal of finding a position with a firm, or near enough, so my time was freed up.

So, I made a spoofing example that could be used any place you can use HTML, ie, in IM's, in newsposts, email, etc -- without breaking copyright by stealing their images and such and just painting the picture with that.

The page I made could be sent in a link to anyone. The opening page that is the controlling page and creates the content is displayed off the user's screen. All the user sees is Microsoft's security website from the url they just clicked (say, from Microsoft security in the "from" address).

What the page does is it has to wait for it load, then it searches for strings in the page and changes the relative paths to static paths so that all of the images and scripts work. Then, I changed some code dynamically so my hacker code appears on Microsoft's security website. Very scary, very neat.

Yet I don't think there was a security researcher that saw the first demo that didn't have that in mind when they saw my first exploit. I sure did, that this was the sort of danger was immediately known to me and everyon else.

Regardless, non-security researchers do not fill in the blanks... and even security researchers are amazed when those blanks are filled in for them. It is fun.

Writing the Paper

I have found bugs that I do not know why they do what they do. However, you have to know why they do what they, so you have to figure out what is going on. Because I test so much code and throw in so many random techniques this happens.

I won't actually go into how to write a security research paper - countless examples of that - but, I will point out that when you have proven the bug to yourself you are in a serious situation.

One, you have to keep your mouth shut. If you told someone they could use it and you would be the guy that gets busted for it. There is one case up right now where a hacker released a bug and he is busted for it because it was used. He probably did it, I don't know, trial is not over.

Two, this sort of thing is very useful to organized crime and malicious foreign governments. I don't think they are "that smart" yet, but obviously, you are in possession of a very serious thing. You have to keep your mouth shut.

Three, someone else could discover the bug after

you and publish it first. This, actually, happens quite often. Or, one of your good buddies could publish it as theirs. I never heard of this happening, though there have been accusations.

Lastly, I found bugs in IE which were minor, when I was looking for root. I had to post these in the meantime. This is the way this stuff works. This doesn't mean that you should post everything, the directory traversal bug I just posted when I saw another directory traversal bug and felt I should comment on it. I was sitting on it to try and make it into something bigger.

Once you publish a bug, that's it. Its' gone. I publish as soon as possible.

Lastly, A Look At the Virtues of Bug Finding

This can help desire, it is very often the desire for doing this sort of thing, but let me put you in the picture.

a> It is really quite easy if you can do it
b> It is funner than programming accounting databases
c> A single big vulnerability will get you a great deal of press, this is fun, I admit, though don't work for praise nor fame
d> multiple major bug vulnerabilities makes a career and you well reknowned in the field
e> security researchers get cushy jobs and employers dying to have them. Believe it. While the typical security consultant now makes about 75K... big bug finders often are offered positions ranging in the upper six figures from banks and other such institutions. You shouldn't be greedy or work solely for money but we all have to eat.
f> There really aren't that many security researchers, and there is
plenty of room for more. There are a lot of talkers, even established
talkers whom have never found a bug themselves.
g> while I call it "bug finding" and it has been called "code hacking",
the fact is it is the purest expression of hacking prowess possible. Website
defacers would be nothing with these exploits bug finders find. Anybody can be a penetration tester. Few can find their own vulnerabilities. It is true that Kevin Mitnick used vulnerabilities, though he is an authority on hacking -- one I greatly respect. There are others like him... though, guys that use existing exploit very cleverly. Though, bug finding is really where it all ends and begins.
h> "bug finding" can force vendors to take security seriously, you make a direct impact on the security of the industry
i> it is exhilirating to be in possession of such power as a security bug gives you, it is a golden key to all of the systems of the world

And, many more...

Signed,
the Pull

[About the author]

[I can't begin to go into my security record in this, though I have written a HTML trojan application that reconstructs a binary of the user's choice. I predicted KAK, Bubbleboy, Code Red directly and often in security papers. I wrote the first timed fused base ddos agent. I predicted a lot of problem's in IM and found a buffer overflow in ICQ. I am a charter member of the Hacktivismo group, a special operations group sponsered by the Cult of the Dead Cow. I have experience as a QA Lead at a p2p company where I helped find thousands of
bugs in our projects.]

[Recently, to help me find the right position in the field, I found six serious problems within Internet Explorer 6 within a month including being able to run arbitary applications (including Control Panel applets), reading local files, spoofing websites, reading cookies, and being able to find system user names and paths remotely. I have experience dating back to 81. I am experienced in developing in a wide range of languages.]

[As far as everything else goes... I am big into security from physical security to computer security to ethics -- which traverses the fields of all the human sciences to the religions. My primary other interest is weightlifting.]

Fingerprinting Port 80 Attacks

A look into web server, and web application attack signatures: Part Two

By Zenomorph

Introduction:

Port 80 is the standard port for websites, and it can have a lot of different security issues. These holes can allow an attacker to gain either administrative access to the website, or even the web server itself. This second paper was written to help the average administrator and developer to have a better understanding of the types of threats that exist, along with how to detect them.

More Common Fingerprints: This section has examples of more common fingerprints used in exploitation of both web applications, and web servers. This section is not supposed to show you every possible fingerprint, but instead show you more ways an attacker can possibly get into your system, along with how an attackers presence could be masked. These signatures should pick up most of the remaining methods not spoken about in the first paper. This section also describes what each signature is used for, along with examples of it being used in an attack.

" * " Requests

The asterisk is often used by attackers as an argument to a system command.

* http://host/index.asp?something=..\..\..\..\WINNT\system32\cmd.exe?/c+DIR+e:\WINNT*.txt

This request is asking for all text files within the directory of e:\WINNT to be listed. Requests like these can often be used to gather a list of log files, along with other important files. Not a lot of web applications use this character in a valid request so this makes an asterisk stand out in logs.

* http://host/blah.pl?somethingelse=ls%20*.pl

This request is asking for a directory listing on a Unix system of all perl scripts that end in .pl.

" ~ " Requests

The ~ character is sometimes used by attackers to determine who is a valid user on your system.

Below is an example

* http://host/~joe

This request is looking for a user named "joe" on the remote system. Often times users will have web space and if the attacker manages to visit a web page, or get a 403 error(Denied error) then a user exists. Once an attacker has a valid username, they may try guessing passwords, or brute forcing until they get a valid password. There are other ways of finding out who is a valid user but this is a port80 request so I figured I'd mention it. (This is a well known method) It can easily be misidentified as a valid request in IDS logs depending on if the system has user pages in this format.

" ' " Requests

If this particular character shows up in your logs then there is a possibility someone is trying a SQL injection attack against your software. Often times programs may be written poorly and may allow an attacker to insert SQL commands into the script. If it is possible to execute system commands then it may be possible for an attacker to gain administrative access to your system. (Sometimes administrators run SQL as root on Unix, and if you run MS-SQL it already runs with administrative privileges)

Below is an example.

* http://host/cgi-bin/lame.asp?name=john`;EXEC master.dbo.xp_cmdshell'cmd.exe dir c:'--

This request is executing the cmd.exe shell on a windows NT machine. From here an attacker has free reign on the remote machine with access to add users, upload trojans, and steal the sam password file.

For more information on SQL attacks visit www.cgisecurity.com/lib and check out a few papers we've collected from various sites on the subject. Also check out www.owasp.org for further examples of SQL Injection.

" #, {} , ^ , and [] " Requests

These particular characters may show up in your logs if an attacker is echoing some source code into a file of a perl or c program. Once a file is created and compiled/interpreted the attacker could bind a shell to a port giving themselves easy access.

I won't show a complete example of this because in order to do so I'd have to paste a bindshell program. This paper wasn't written to give people easy to follow example on how to use trojans. For this reason I have decided not to include an example. [and] may also be used as a command argument in Unix for commands like ls -al [a-f]*. This would list all the files starting with characters between a and f. # may show up if an attacker is uploading a perl script backdoor

(Ex: #!/usr/bin/perl at the top of the file).

Below is an example using #

* http://host/dont.pl?ask=/bin/echo%20"#!/usr/bin/perl%20stuff-that-binds-a-backdoor"%20>/tmp/back.pl;/usr/bin /perl%20/tmp/back.pl%20-p1099

" (and) " Requests

This value is often used in cross site scripting attacks. Cross Site Scripting has gotten a lot of attention lately, and a lot of large sites still suffer from this type of attack.

Below is an example.

* http://host/index.php?stupid=<img%20src=javascript:alert(document.domain)>

This example above will be sent to the index.php. From here an output page will be displayed with the following javascript. Next your browser will execute this javascript and display a popup window. Cross site scripting is considered a low to medium threat. It does have the ability to allow an attacker to steal cookies from you. An obvious way to prevent this would be to make sure the output doesn't contain < or > in them. This way the javascript will not be executed.

" + " Request

Sometimes the + is used as a blank space similar to "%20" as mentioned in my previous paper. This value (when used in an attack) is often used with cmd.exe backdoored hosts. Often times an attacker or worm will copy cmd.exe to a file inside the webroot. Once this file is copied an attacker has full control over your windows machine. He will use the + character to help pass arguments to the script. If this character comes up in your logs don't freak. This character is widely used with web applications and it can be easily misidentified. If it manages to pop up in your logs you may want to double check them but there is no reason to panic.

Below is an example.

* http://site/scripts/root.exe?/c+dir+c:\

This particular example is showing a request to a backdoor called root.exe. This backdoor is installed by sadmind/IIS worm, Code Red, and Nimda after a host is compromised. The + character is often used in windows backdoors that involve cmd.exe copies.

Additional Worm Information
http://www.cert.org/incident_notes/IN-2001-09.html

More Advanced Fingerprints

This section focuses more on the files an attacker or worm may request, along with a few other signatures that stand out. This isn't a complete list of requests or files an attacker may request, but it will give you a good idea of what is being attempted against your system.

Lots of "/" Requests

If you check your logs and see A LOT of "/" characters then there is a good chance an attacker is attempting to exploit a well known apache bug. This bug which effects every version of apache before 1.3.20 and allows directory listings. If you see this in your logs someone is attempting to exploit you. (This fills up logs FAST)

Below is an example.

*http://host//

The way this exploit works is the attacker runs a script that keeps adding a slash one at a time. Eventually on an affected system the attacker will be able to gather file listings, among other things.

* Common files and directories an attacker will request.

"autoexec.bat"

This file is started by certain versions of windows every time at boot up. Often times after an attacker has done what he wants with a box, he/she will install tools to remove logs and any reference to an intrusion taking place. An attacker may modify this file and insert commands into this file. Next time the machine is rebooted logs/traces can be wiped and the attacker is home free. People running a web server on windows 95 and 98 will be affected by this problem. You should only be running a public web server on NT/2000 with NTFS for security purposes if you plan on using a windows product.

"root.exe"

This is the backdoor left by Sadmin/IIS, Code Red, and Nimda worms. This backdoor is a copy of cmd.exe renamed to root.exe and put inside the webroot. If an attacker or worm has access to this file, you can bet your in trouble. Common directories this file resides in are "/scripts/" and "/MSADC/".

"nobody.cgi 1.0 A free Perl script from VerySimple "

This is a cgi program, which was originally written to help provide admins with a shell backdoor. It also has a hefty warning by the programmer explaining the dangers of improperly using this program. This is now a popular backdoor used by attackers to execute commands with the permission of the webserver. You really would be surprised how often I see this popping up. Hanging in chat rooms I've seen 3 different occasions where people (unaware of each other) have used this script. Oh and no I won't give you the link to this product.

"[drive-letter]:\WINNT\system32\LogFiles\"

This is the directory that contains the IIS server logs. An attacker may attempt to view your logs via a web application hole. If you see a reference to system32/LogFiles there is a good chance your system is already taken over.

"[drive-letter]:\WINNT\system32\repair\"

This is the directory that contains the backup password file on NT systems.The file will either be named "sam._"(NT4) or "sam"(Win2k). If an attacker manages to get a hold of this file then you're in for some real trouble.

Novell File systems
"[server-name]:SYSTEM:PUBLIC"

This is an example Novell file system. It may be possible an advanced attacker with deep knowledge of Novell may try to view files remotely. Getting information such as the intranet server name may not be too easy on the other hand.

Cross Site Scripting Examples

Cross site scripting attacks are often used by an attacker to make the user think that certain information is actually coming from another site. These attacks are often used in scams, or when an attacker is trying to fool people into thinking certain things about companies in order to lower the price of the stocks, product prices, etc.. One problem with this attack type is that the attacker must have the user click on a link he provides in order to view this information. Sometimes an attacker will use other existing holes to make this process more believable. These attacks are very common and a lot of major sites are affected by this attack type in some way or another.

Below are a few examples of requests an attacker will use when trying to fool a user.

Example 1: The IMG tag

* http://host/search/search.cgi?query=<img%20src=http://host2/fake-article.jpg>

Depending on the website setup and if the search engine doesn't filter requests for html tags, this generates html with the image from host2 and feeds it to the user when they click on this link. Depending on the original web page layout it may be possible to fool a user into thinking this is a valid article. One problem is the url above is very obvious and anyone with half a brain would notice something was wrong. This request could be encoded on the other hand so that when a user clicks on this link they do not get suspicious. I posted an example in relation to perl.com a few months ago and even managed to fool a staff member at O'Reilly with this. (Only for like an hour though :)

Example 2:

* http://host/something.php?q=<img%20src=javascript:something-wicked-this-way-comes>

If a user clicks on this link a JavaScript popup box displaying the sites domain name will appear. While this example isn't harmful, an attacker could create a falsified form or, perhaps create something that grabs information from the user. The request above is easily questionable to a standard user but with hex, unicode, or %u windows encoding a user could be fooled into thinking this is a valid site link.

Example 3:

* http://host/<script>Insert stuff here</script>

This particular request is very common example. About once a month I'll see a new script that is affected by this. If you see something like this in your logs, there is a good chance someone is testing your scripts out.

Modified Headers:

I recently wrote a paper on web statistical software and the types of exploitation
that can happen via http header modification. I will use a few excerpts of this
paper below to show you the types of fingerprints that exist for it.

This paper can be found here

* Http://www.cgisecurity.com/papers/header-based-exploitation.txt

Below is a review of some things to look for in your logs.

x.x.x.x - - [10/Dec/2001:09:03:39 -0500] "GET / HTTP/1.1" 200 10453 "http://www.cgisecurity.com" "Mozilla/4.0
(compatible; MSIE 5.5; Windows NT 5.0; T312461)"

We are going to look at the 11th and 12th field in this log.

11th "http://www.cgisecurity.com" Referrer Field
12th "Mozilla/4.0" User Agent Field

These fields are filled in by your browser automatically. If I had a link on www.hosta.com that pointed to my site and clicked on it, then my browser would save this information and forward it to my website. This information is known as the referrer field. The referrer field is filled in by your browser automatically, which means this information is provided by the client, and not the server. This means this information is "user input". Since this information is user input this means we can change it to whatever we want.

The threat in this is that certain types of software gather the values from your logs and display them out. (Example Web Stats Software) Some software doesn't do stripping of metacharacters very well and because of this code insertion is possible.

Example 1:

su-2.05# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0

Referer: <!--#virtual include="somefile.log"--> (Yes Referrer is spelt wrong)
User-Agent: <!--#exec cmd="/bin/id"-->

"In this example the attacker is inserting SSI tags into the "Referrer" and User-Agent fields. Depending on whether the software outputs this information as text or in image form, this could lead to possible file includes, or command execution. (Of course these examples could be interchangeable) If the logs are shown as text and displayed in a shtml file, and the referrer, or user agent fields are shown (Most of the time they are), then these two requests will be included in the file. Now next time a visitor views these logs, the SSI tags will be executed by the web server, and should display the results of the "id" command, and the contents of "somefile.log".(Once again depending on server configuration)"

- * Http://www.cgisecurity.com/papers/header-based-exploitation.txt

If for any reason you find a request like this in your logs, then it may be possible someone is trying to exploit you.

Below is another example from my previous paper.

Example 2:

su-2.05# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0
Referer: <javascript-that-is-evil-so-there's-no-need-for-examples>
User-Agent: </html>

"If a user visits this stats page and the referrer is outputted then it may be possible to steal users cookies. " It is suggested that you check this field in your apache logs regularly, if you feel that you run software which may be at risk.

There is a good tool that can modify http headers called "Websleuth" that is worth checking out.

VI. More Encoding:

This chapter covers common encoding methods an attacker or worm may use to help avoid detection. Hex, Unicode, and windows %u encoding is covered.This isn't a "how to" section, rather something that may help explain what you may run into in your logs.

A. Hex Encoding:

Example: %xx

Below are the hex values of some of the characters mentioned in the last paper. If you see these characters in any log file there is a good chance an attacker is trying to mask his requests, or even trying to get around an IDS product.

Encoded characters mentioned in last paper/this paper.

%2e = . (Example: .. requests)
%3e = > (Example: Html/Javascript/SSI insertion. Mentioned in last paper)
%3c = < (Example: Html/Javascript/SSI insertion. Mentioned in last paper)
%2a = * (Examples Listed in chapter 2 of this paper)

%2b = + (Example: cmd.exe backdoor request. Also used as space)
%60 = ` (Examples Command execution. Mentioned in last paper)
%21 = ! (Example: SSI insertion. Mentioned in last paper)
%7c = | (Example: Command execution. Mentioned in last paper)
%3b = ; (Example: Command execution. Mentioned in last paper)
%7e = ~ (Examples Listed in chapter2 of this paper)
%3f = ? (Example: Php/Mentioned in last paper)
%5c = \ (Example: Possible Encoded Windows Directory Transversal Attempt)
%2f = / (Example: Possible Encoded Unix Directory Transversal Attempt)
%7b = { (Example: Possible trojan/backdoor upload attempt, possible command argument)
%7d = } (Example: Possible trojan/backdoor upload attempt, possible command argument)
%28 = ((Example: Possible Cross Site Scripting attempt)
%29 =) (Example: Possible Cross Site Scripting attempt)
%5b = [(Example: Possible trojan/backdoor upload attempt, possible command argument)
%5d =] (Example: Possible trojan/backdoor upload attempt, possible command argument)
%5e = ^ (Example: Possible trojan/backdoor upload attempt, possible command argument)

For a complete list of characters in Unix type "man ascii" and a list will be provided. Below is what An example of directory transversal would look like while trying to fetch the server's password file.

Example 1 :

* http://host/script.ext?template=%2e%2e%2f%2e%2e%2f%2e%2e%2f%65%74%63%2f%70%61%73%73%77%64

This request is made up of:

1. %2e%2e%2f%2e%2e%2f%2e%2e%2f = ../../../

2. %65%74%63 = etc

3. %2f = /

4. %70%61%73%73%77%64 = passwd

Combinations of this will probably be used to help further fool your IDS product.
Tools like rain.forrest.puppies "Whisker" use these techniques to avoid detection.

B. Unicode Encoding:

Example: %xx%xx

This type of encoding by now, has been heard about by most people who deal with security. The famous IIS exploit that used this encoding method is an example of what a Unicode request looks like.

* http://127.0.0.1/scripts/..%c0%af../winnt/system32/cmd.exe?+/c+dir+c:\

I'm not going to get into this encoding method much before plenty of documentation already exists. For additional information about unicode Read "RFC 2279" and the link below.
http://www.ietf.org/rfc/rfc2279.txt

C. "%u" Encoded Requests:

Example: %uxxxx

This is a type of encoding used by the Microsoft IIS web server. Through the use of this Microsoft specific encoding method, an attacker can possibly evade IDS products. Below is an example of what a worm or attacker may send to a vulnerable system with and without %u encoding.

* http://host/lame.asp?asp=a.txt

This request is attempting to read the file "a.txt" using lame.asp.

* http://host/lame.asp?asp=%u0061.txt

This request does the same thing using "%u" Microsoft encoding. While this may still draw attention when you view the logs manually, an IDS product may miss this request, and allow the attacker to continue
his fun unnoticed. This type of encoding can also be used in conjunction with normal ASCII characters, and
because of this alone, some IDS products will not detect such a request.

Visit the link below for further information on this encoding method. http://www.eeye.com/html/Research/Advisories/AD20010705.html

VII. Web server Codes and Logging:

Often times when an attacker is trying to exploit your web application it will cause your software to produce error messages both seen, and unseen by the attacker. This section will cover the types of error messages that will show up in your logs, and what they may mean. This section covers basic logging and is meant more for newbie's. Skip ahead if you already have a good grasp on logging to the next chapter.

403 Denied Errors

This particular error happens when you have a file that is not marked world readable. Sometimes the webmaster can make a mistake and accidentally forget to chmod a file readable. A lot of the time when a file is marked not world readable (Example a password file), and someone requests it through your website this is an alarm to either move the file, and examine your logs further.

[Wed Feb 20 10:23:33 2002] [error] [client 192.168.1.1] (13)Permission denied: file permissions deny server access:
/some/path/htdocs/secret/apache-unreleased-overflow.c
(Message as it would appear in your error_log)

192.168.1.1 - - [20/Feb/2002:10:23:33 -0500] "GET /secret/apache-unreleased-overflow.c HTTP/1.0" 403 206
(Message as it would appear in your access_log)

404 Not Found errors

When running a large website or even a medium sized one, people may start linking to material on your website directly from another site. As time goes by sometimes things get moved around a bit and these old references to files are no longer valid. You may see such a reference in your access_log or easier to see error_log file. Sometimes these requests for invalid, or obsolete files can let you know if you've renamed a file to the incorrect name, or that someone is poking around. IDS systems would not pick up the majority of 404 error because they aren't considered an immediate threat. Picking up on 404 codes would be a nightmare because 404 codes are a normal issue websites deal with and are 99.99 percent of the time not attacks/probes at all. Instead IDS software tends to match signatures of filenames, some of which I will mention below.

This below log entry is from a person scanning my site looking for the popular formail cgi script. Formail is known to have multiple security issues, and just recently it has been found to be widely used by spammers to send people unwanted email.

[Wed Feb 20 10:30:42 2002] [error] [client 192.168.2.2] script not found or unable to stat: /usr/local/apache/cgi-bin/formail.pl
(Message as it would appear in your error_log.)

|-- 404 Code
192.168.2.2 - - [20/Feb/2002:10:30:42 -0500] "GET /cgi-bin/formail.pl HTTP/1.0" 404 3683 "-" "Mozilla/4.78 [en] (Win98; U)"
"Mozilla/4.78 [en] (Win98; U)"
(Message as it would appear in your access_log)

This can be an alert that someone is scanning your machine, or subnet for holes. Obviously just because a 404 is triggered in your logs this doesn't mean your under attack. Carefully study your logs for common files that may be mislinked, and also check for anything out of the ordinary.

Below is another example this time requesting a backdoor left by the Nimda, and well known Code red worm.

[Tue Dec 18 05:11:04 2001] [error] [client 192.168.3.3] File does not exist: /usr/local/apache/htdocs/MSADC/root.exe
(Message as it would appear in your error_log)

|--- 404 code
192.168.3.3 - - [18/Dec/2001:05:11:04 +0000] "GET /MSADC/root.exe?/c+dir HTTP/1.0" 404 3147
(Message as it would appear in your access_log)

Often times people scan for these files hoping to get an easily backdoored machine. From here they would have complete control of your IIS machine.

500 Server Error

Sometimes when an attacker is testing out software for command execution, or remote file read abilities they will insert characters (Like mentioned above) to help achieve this goal. Sometimes scripts will not handle this additional data insertion well and instead terminate abnormally. This will show up in your logs as a server error (500 code). Not all 500 codes mean an attacker is scanning you. Often times users who upload scripts, which are not configured correctly for this particular system , can give this error.

EDIT
Below is an example

|--- 500 Code
192.168.4.4 - - [18/Dec/2001:05:11:04 +0000] "GET /cgi-bin/port80.cgi HTTP/1.0" 500 529 "-" "Mozilla/4.78 [en] (Win98; U)"
(access_log)

[Thu Dec 13 15:30:23 2001] [error] [client 192.168.4.4] Premature end of script headers: /usr/local/apache/cgi-bin/port80.cgi
(error_log)

Depending on what exactly the attacker is attempting to do, will determine exactly what the reason will be in your error_log.
EDIT

Htaccess error codes

Not all error messages are attacks against your system. Often times it could be as simple as a user using the wrong username, or password. Sometimes on the otherhand attackers will run a program like "WWWhack" to brute force your password to gain entry to protected area's. Below is an example

192.168.5.5 - miked [30/Jan/2002:13:37:26 -0500] "GET /secret HTTP/1.0" 401 397 "-" "Mozilla4.78 [en]C-CCK-MCD sn
apN45b1 (Win98; U)"
(Message as it would appear in your access_log)

[Wed Jan 30 13:37:26 2002] [error] [client 192.168.5.5] user miked: authentication failure for "secret": password mismatch
(Message as it would appear in your error_log)

This shows a failed login attempt by 192.158.5.5 trying the username of miked. If for any reason you see a lot of failed requested from the same ip address, then there is a good chance someone is trying to brute force your password protection. Between 1-40 may be just a user who forgot his password. Another hint that someone is attempting is breaking is if 1 ip address is trying to attempt to login with non existent accounts, or trying to use multiple usernames.

A complete list of error codes can be found at the link below.
http://www.w3.org/Protocols/HTTP/HTRESP.html

Extended logging options with apache

Apache has a module that is used for logging called "mod_log_config". This module allows an administrator the ability to choose which format his data is logged in. It also allows the administrator to choose which headers are logged. Sometimes new types of attacks get published that use extended HTTP headers. (Examples: Content-Encoding, Host, Etag, Content-MD5, Warning, WWW-Authenticate, etc...) By default apache does not log these fields. The "LogFormat" Directive gives the administrator the ability to choose what is logged and what isn't. This can be particularly useful when troubleshooting breaches, or application problems.

Mod Log Config:
http://httpd.apache.org/docs/mod/mod_log_config.html

LogFormat Directive:
http://httpd.apache.org/docs/mod/mod_log_config.html#logformat

Further information on HTTP headers can be found below:

RFC 1945: Hypertext Transfer Protocol -- HTTP/1.0
http://www.ietf.org/rfc/rfc1945.txt

RFC 2068: Hypertext Transfer Protocol -- HTTP/1.1
http://www.ietf.org/rfc/rfc2068.txt

Conclusion:

Once again, this paper doesn't cover every port 80 exploit but it covers the most common types of attacks. It tells you what to check for in your logs, look for when programming an application, and when writing your IDS rules.

I wrote the second piece of this paper due to the large interest in the first one. I also would like to promote more awareness of these issues in an easy to understand paper. If you have any comments or

suggestions email me at admin@cgisecurity.com.

References and links mentioned within

Apache Related:

Mod Log Config:
http://httpd.apache.org/docs/mod/mod_log_config.html

LogFormat Directive:
http://httpd.apache.org/docs/mod/mod_log_config.html#logformat

IIS %u Encoding:
http://www.eeye.com/html/Research/Advisories/AD20010705.html

HTTP Related:

Status Codes
http://www.w3.org/Protocols/HTTP/HTRESP.html

RFC 1945: Hypertext Transfer Protocol -- HTTP/1.0
http://www.ietf.org/rfc/rfc1945.txt

RFC 2068: Hypertext Transfer Protocol -- HTTP/1.1
http://www.ietf.org/rfc/rfc2068.txt

Misc:
http://www.w3.org

SQL Injection:
http://www.spidynamics.com/
"SQL Injection Are Your Web Applications Vulnerable?" Kevin Spett, 2002

http://www.ngssoftware.com
"Advanced SQL Injection In SQL Server Applications" Chris Anley, 2002

Unicode:
http://www.w3.org/TR/REC-html40/charset.html

Special Thanks:

OWASP (Open Web Application Security Project - www.owasp.org)
Mark Curphey
Dennis Groves
Joel Gridley (a.k.a. Jarmaug)
Mike D. For the failed login attempts :)
PhantasmP
zenomorph for providing you with this hopefully useful paper

Published to the Public March 2002

WHEN YOUR SERVER ENDS UP A WAREZ SITE

-Strange logs from your FTP server

By Obscure^
obscure@eyeonsecurity.net

Intro

About my experiments.

Last week I opened an anonymous ftp site on my home machine, expecting a few connections. I also wanted to see what people would do if I gave them write access. Within 3-4 days of my server being up, I got a successful connection from a remote host which created his own directory named "_kurdt". Later on, I got another connection from a possibly different visitor, who created a different directory name "020612105639p". Checking my ftp logs, I learnt that both processes seem automated: within the same second the user has logged in, created a folder and disconnected from my ftp server. The third scan consisted of testing upload, deletion and ftp/http miss-configuration. These attacks are described in detail on the log files section.

FXP and Pub Scanning

"FXP stands for File eXchange Protocol and it let's you copy files from one FTP-server to another using a FXP-client. Normally you transfer files using the FTP protocol between your machine and a FTP-server, and the maximum transfer speed depends on the speed of your Internet connection (e.g. 56k, cable or T1). When transferring files between two remote hosts using a FXP client, the maximum transfer speed does not depend on your connection but only on the connection between the two hosts, which is usually much faster than your own connection. Because it is a direct connection you will not be able to see the progress or the transfer speed of the files." (From http://www.ultimatefxp.f2s.com/tutorials/tutorial.htm)

Technically this means that a client will initiate a PASV ftp connection from host A to host B, by giving the destination IP of host B as destination. This attack is normally described as FTP Bounce Attack.

Pub Scanning on the other hand, is about scanning for ftp sites, which allow you to upload and download your own stuff. Scanning for such ftp sites can be done either manually using a port scanner or checking each ftp site using an ftp client, or increasingly using software for the sole purpose of scanning for such sites. This is described further on in the Scanning Tools section. Having such access for Warez people means that they can have large ftp sites with good bandwidth, easily accessible for trading Warez, mp3s, vcds and so on.

Difference between Warez "D00ds" and Hackers

To the unwary administrator, such activity will look like his ftp site has been hit by another evil cracker (AEC) [tm]. In reality, the methods used for pub scanning and FXP are quite similar to patterns generated by AEC people. However, the scope is quite different. While a cracker will want to penetrate the system, and maybe the network, to gain access to more machines, maybe for DDoS (quite pop nowadays) or to deface a site, the average Warez pub-scanner will probably only want Gigabytes of storage and bandwidth. That is not to say that exceptions do not exist. Crackers have been known to leave Warez on servers, and Warez people have also been using "exploits" (mostly exploiting miss-configuration) to gain better access to their target hosts. In fact, with Pub-scanning becoming more sophisticated, methods used by hackers to penetrate hosts on the 'net are increasingly being used for

Warez dissemination. Also, most Warez people will use Windows as opposed to a certain section of the hacker community that prefers Linux and *BSDs.

Attacking.

Tools of trade

Grim's Ping is probably one of the most used tools around. Version 1.71 boasts a good number of features:

Features
---
*Scan specified ports, using a proxy if you wish
Ping 24.4.4. IP range
*Host lookup
*Perform "Pub Find" on an infinite number of IP ranges
*Log Wingate engines found, in addition to FTPs
*Wingate usage to protect privacy
*Built in FTP client
*Log or print scan results
*Check write and delete permissions
*Check OS type and FXP/Resume capabilities
*Record speed
*Modify queue to reflect your scanning processes
*Import queue lists from other popular scanning utilities
*Autosave queue
*Many configurable options

As you can see, it supports anything a pub-scanner could wish for. Gives statistics, supports "anonymity" (as described later on) and will efficiently do automated scanning for different FTP sites.

As an add-on, Grim has also included Ping Companion, which will upload space.asp, an Active Server Page which displays information about the host. It will also try to upload 1k and 1mb test files to check whether the ftp server is really capable of hosting a Warez site.

An interesting tool in use is Omega Scanner:

Script Based Internet Scanner

"Omega Scanner is a multi-threaded script based Internet scanner. With the advantage of scripts, Omega Scanner can be configured to scan for almost anything - from SMTP to FTP servers. The variety of scripts included with Omega Scanner shows the power of script-based Internet scanning.

Omega Scanner supports proxy SOCKS4 and SOCKS5"

Numerous scripts are available for FTP pub and FXP scanning… making it another tool of choice.

Another tool worth mentioning is FLashFXP ftp client, which supports ftp to ftp transfer.

Features:

· Local and Site to Site file transfers.
· Fully recursive file transferring.
· Fully recursive deleting.
· FTP Proxy, Socks 4 & 5, HTTP Proxy support.
· Grouped SITE custom commands.

· Anti-idle keeps connection active.
· Caching of directory lists.
· Disconnect Dialup-Networking once transfer has completed.
· Restore broken transfers. (reconnects and restarts file transfer)
· Drag-drop from Windows Explorer.

· System tray minimize.

- Warez Trends

Tagging

Warez traders exist in groups, so that each group will have a couple of members who actively scan for pubs. Since different warez groups will target each ftp site, each group creates its own tag, to claim that ftp site as its own territory.

A tag will typically look something like "-=ACF=-" or "[DVD-R]". Grim's Ping site hosts a tag list on http://grim.virtualave.net/addtag.cgi?view . The idea is that ethical pub-scanners, respect tags and don't upload their own files if the ftp site is already in use by another group. Of course, non-ethical scanners exist, and they are sometimes called deleters.

Rating Pubs

Pubs are published on Warez bulletin boards for other users to upload and abuse. Most lists of pubs will consist of more than just IP addresses. Typical lists will include the uploadable directory, delete statistics, that is, if the uploaded files are delectable by other users, the Operating system of the ftp site, if the site is able to resume downloads and uploads (a handy feature when doing huge downloads), if it is FXPable, and the download speed. Grim's Ping Companion's space.asp, which was described earlier, will give scanners further information about the target machine including the name of logical drives, type of drive, volume name, free and total space, file system for each drive and version of IIS which is running.

Hiding files

The process of uploading Warez and other goods takes time and patience. That means that the uploader wouldn't like to have his directory deleted after a few days (or hours), by the legitimate administrator, opposing Warez groups or simply clueless roamers. For this purpose, Warez d00dz have learnt various tricks to hide their stuff.

The most commonly known method for hiding directories is to prefix the filename with a dot (.). This will hide the file on most Unix machines. Another effective method is to use the tide symbol (~). Many ftp clients will direct the user to the user directory when he tries to access ~, therefore keeping certain people out and letting others in. Adding spaces to the folder and using loads of dummy directories (maze) are other ways the pirate uses to hide the treasure.

Anonymity

Many pub-scanners are well aware of the risk involved, some of them will probably have already been tipped off by some ISP or worse, got their account stopped because of their illegal activity. Therefore, the use of anonymous proxies, wingates and socks is quite popular among the community. Some will be really paranoid and use multiple wingates to bounce their connection, in hope that it will take much longer to get traced back. These techniques are better covered in my other article about anonymity and other issues: "Browsing Websites at your own risk".

Prevention and Post Attack Analysis.

This section is mostly for anyone (mostly administrators) hosting an ftp site.

Log files

During my testing, (i.e. being a honeypot), I configured Serv-U to log everything to a text file for easy manual parsing. The following entries show pub-scanner's activity:

[5] Thu 07Jun01 13:06:42 - (000004) Connected to 61.170.139.40 (Local address x.x.x.x)
[6] Thu 07Jun01 13:06:42 - (000004) 220 EOS FTP 2.1 Ready ...
[2] Thu 07Jun01 13:06:42 - (000004) user anonymous
[6] Thu 07Jun01 13:06:42 - (000004) 331 User name okay, please send complete E-mail address as password.
[2] Thu 07Jun01 13:06:43 - (000004) pass ncoic77@hotmail.com
[5] Thu 07Jun01 13:06:43 - (000004) ANONYMOUS logged in, password: NCOIC77@HOTMAIL.COM
[6] Thu 07Jun01 13:06:43 - (000004) 230 User logged in, proceed.
[2] Thu 07Jun01 13:06:43 - (000004) mkd _kurdt
[6] Thu 07Jun01 13:06:43 - (000004) 257 "/_kurdt" directory created.
[5] Thu 07Jun01 13:06:44 - (000004) Closing connection for user ANONYMOUS (00:00:02 connected)

The above shows the first scan by an pub-scanner. "kurdt" seems to be the nickname (or tag) of the client. Doing a search for _kurdt on google, produced me with some published warez sites. So this clearly confirmed my suspicion. Apart from that he's probably using Omega Scanner with "pub searchin' script.oss", which uses ncoic77@hotmail.com as password.

The second connection produces the following logs:

[5] Tue 12Jun01 10:54:40 - (000003) Connected to 213.51.52.27 (Local address x.x.x.x)
[6] Tue 12Jun01 10:54:41 - (000003) 220 EOS FTP 2.1 Ready ...
[5] Tue 12Jun01 10:54:41 - (000003) IP-Name: CP17725-A.DBSCH1.NB.NL.HOME.COM
[2] Tue 12Jun01 10:54:41 - (000003) USER anonymous
[6] Tue 12Jun01 10:54:41 - (000003) 331 User name okay, please send complete E-mail address as password.
[2] Tue 12Jun01 10:54:41 - (000003) PASS guest@here.com
[5] Tue 12Jun01 10:54:41 - (000003) ANONYMOUS logged in, password: GUEST@HERE.COM
[6] Tue 12Jun01 10:54:41 - (000003) 230 User logged in, proceed.

Guest@here.com is produced by the popular pub-scanner Grim's Ping.

[2] Tue 12Jun01 10:54:41 - (000003) CWD /pub/
[6] Tue 12Jun01 10:54:41 - (000003) 550 /pub: No such file or directory.
[2] Tue 12Jun01 10:54:41 - (000003) CWD /public/
[6] Tue 12Jun01 10:54:41 - (000003) 550 /public: No such file or directory.
[2] Tue 12Jun01 10:54:41 - (000003) CWD /pub/incoming/
[6] Tue 12Jun01 10:54:41 - (000003) 550 /pub/incoming: No such file or directory.
[2] Tue 12Jun01 10:54:42 - (000003) CWD /incoming/
[6] Tue 12Jun01 10:54:42 - (000003) 550 /incoming: No such file or directory.
[2] Tue 12Jun01 10:54:42 - (000003) CWD /_vti_pvt/
[6] Tue 12Jun01 10:54:42 - (000003) 550 /_vti_pvt: No such file or directory.

It immediately tries to search for a directory to write to.

[2] Tue 12Jun01 10:54:42 - (000003) CWD /
[6] Tue 12Jun01 10:54:42 - (000003) 250 Directory changed to /
[2] Tue 12Jun01 10:54:42 - (000003) MKD 020612105639p

[6] Tue 12Jun01 10:54:42 - (000003) 257 "/020612105639p" directory created.
[2] Tue 12Jun01 10:54:42 - (000003) RMD 020612105639p
[6] Tue 12Jun01 10:54:42 - (000003) 550 /020612105639p: Permission denied.
[2] Tue 12Jun01 10:54:42 - (000003) SYST
[6] Tue 12Jun01 10:54:42 - (000003) 215 UNIX Type: L8
[2] Tue 12Jun01 10:54:43 - (000003) REST 1

The following information about my ftp is obtained: my ftp is writable at the root directory, directories are not deletable and OS is UNIX.

[6] Tue 12Jun01 10:54:43 - (000003) 350 Restarting at 1 - send STORE or RETRIEVE to initiate transfer.
[2] Tue 12Jun01 10:54:44 - (000003) PASV
[6] Tue 12Jun01 10:54:44 - (000003) 227 Entering Passive Mode (x,x,x,x,11,202)
[2] Tue 12Jun01 10:54:44 - (000003) PORT 207,46,133,140,1,21

The ip: 207.46.133.140:21 is ftp.microsoft.com. This guy is trying to test if my ftp server will allow him to FXP.

[6] Tue 12Jun01 10:54:44 - (000003) 200 PORT Command successful.
[2] Tue 12Jun01 10:54:44 - (000003) CWD
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
[6] Tue 12Jun01 10:54:44 - (000003) 550 /
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp: No such file or directory.
[2] Tue 12Jun01 10:54:44 - (000003)
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
[6] Tue 12Jun01 10:54:44 - (000003) 500 'PPPPPPPPPPPPPPPPPPPPPPPP': command not understood.
[2] Tue 12Jun01 10:54:44 - (000003)
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
[6] Tue 12Jun01 10:54:44 - (000003) 500 'PPPPPPPPPPPPPPPPPPPPPPPP': command not understood.
[2] Tue 12Jun01 10:54:44 - (000003)
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
[6] Tue 12Jun01 10:54:44 - (000003) 500 'PPPPPPPPPPPPPPPPPPPPPPPP': command not understood.
[2] Tue 12Jun01 10:54:44 - (000003)
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
[6] Tue 12Jun01 10:54:44 - (000003) 500 'PPPPPPPPPPPPPPPPPPPPPPPP': command not understood.
[2] Tue 12Jun01 10:54:44 - (000003)
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

[6] Tue 12Jun01 10:54:44 - (000003) 500 'PPPPPPPPPPPPPPPPPPPPPPPPP': command not understood.
[2] Tue 12Jun01 10:54:44 - (000003)
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
[6] Tue 12Jun01 10:54:44 - (000003) 500 'PPPPPPPPPPPPPPPPPPPPPPPPP': command not understood.
[5] Tue 12Jun01 10:54:44 - (000003) Closing connection for user ANONYMOUS (00:00:04 connected)

I think this request could be an attempt to overflow the buffer, or simply testing to see what kind of error it gets to identify the OS (and ftp server software) better. Any ideas about this one would be most welcome.

Third entry comes from the same host .. the day after:

[5] Wed 13Jun01 14:23:49 - (000019) Connected to 213.51.52.27 (Local address x.x.x.x)
[6] Wed 13Jun01 14:23:49 - (000019) 220 EOS FTP 2.1 Ready ...
[2] Wed 13Jun01 14:23:49 - (000019) USER anonymous
[6] Wed 13Jun01 14:23:49 - (000019) 331 User name okay, please send complete E-mail address as password.
[5] Wed 13Jun01 14:23:49 - (000019) IP-Name: CP17725-A.DBSCH1.NB.NL.HOME.COM
[2] Wed 13Jun01 14:23:49 - (000019) PASS guest@here.com
[5] Wed 13Jun01 14:23:49 - (000019) ANONYMOUS logged in, password: GUEST@HERE.COM
[6] Wed 13Jun01 14:23:49 - (000019) 230 User logged in, proceed.

Once again this is Grim's Ping Autmated tool, with Companion software, as you will see further down.

[2] Wed 13Jun01 14:23:49 - (000019) CWD /
[6] Wed 13Jun01 14:23:49 - (000019) 250 Directory changed to /
[2] Wed 13Jun01 14:23:49 - (000019) TYPE I
[6] Wed 13Jun01 14:23:49 - (000019) 200 Type set to I.
[2] Wed 13Jun01 14:23:50 - (000019) PORT 213,51,52,27,17,98
[6] Wed 13Jun01 14:23:50 - (000019) 200 PORT Command successful.
[2] Wed 13Jun01 14:23:50 - (000019) STOR /1mbtest.ptf

The scanner uploads a 1mb test file to the root directory.

[6] Wed 13Jun01 14:23:50 - (000019) 150 Opening BINARY mode data connection for 1mbtest.ptf.
[4] Wed 13Jun01 14:23:50 - (000019) Receiving file d:\anonftp\1mbtest.ptf
[4] Wed 13Jun01 14:25:16 - (000019) Received file d:\anonftp\1mbtest.ptf successfully (11.9 Kb/sec - 1048578 bytes)
[6] Wed 13Jun01 14:25:16 - (000019) 226-Maximum disk quota limited to 300000 Kbytes
[6] Wed 13Jun01 14:25:16 - (000019) Used disk quota 1024 Kbytes, available 298975 Kbytes
[6] Wed 13Jun01 14:25:16 - (000019) 226 Transfer complete.
[2] Wed 13Jun01 14:25:17 - (000019) PORT 213,51,52,27,6,55
[6] Wed 13Jun01 14:25:17 - (000019) 200 PORT Command successful.
[2] Wed 13Jun01 14:25:17 - (000019) TYPE I
[6] Wed 13Jun01 14:25:17 - (000019) 200 Type set to I.
[2] Wed 13Jun01 14:25:17 - (000019) RETR /1mbtest.ptf

Then it downloads the file back.

[6] Wed 13Jun01 14:25:17 - (000019) 150 Opening BINARY mode data connection for 1mbtest.ptf (1048578 bytes).

[3] Wed 13Jun01 14:25:17 - (000019) Sending file d:\anonftp\1mbtest.ptf
[3] Wed 13Jun01 14:26:29 - (000019) Sent file d:\anonftp\1mbtest.ptf successfully (14.3 Kb/sec - 1048578 bytes)
[6] Wed 13Jun01 14:26:29 - (000019) 226-Maximum disk quota limited to 300000 Kbytes
[6] Wed 13Jun01 14:26:29 - (000019) Used disk quota 1024 Kbytes, available 298975 Kbytes
[6] Wed 13Jun01 14:26:29 - (000019) 226 Transfer complete.
[2] Wed 13Jun01 14:26:29 - (000019) TYPE A
[6] Wed 13Jun01 14:26:29 - (000019) 200 Type set to A.
[2] Wed 13Jun01 14:26:30 - (000019) PORT 213,51,52,27,9,50
[6] Wed 13Jun01 14:26:30 - (000019) 200 PORT Command successful.
[2] Wed 13Jun01 14:26:30 - (000019) LIST -la
[6] Wed 13Jun01 14:26:30 - (000019) 150 Opening ASCII mode data connection for /bin/ls.
[6] Wed 13Jun01 14:26:30 - (000019) 226-Maximum disk quota limited to 300000 Kbytes
[6] Wed 13Jun01 14:26:30 - (000019) Used disk quota 1024 Kbytes, available 298975 Kbytes
[6] Wed 13Jun01 14:26:30 - (000019) 226 Transfer complete.
[2] Wed 13Jun01 14:26:30 - (000019) DELE /1mbtest.ptf
[6] Wed 13Jun01 14:26:30 - (000019) 250 DELE command successful.

And finally delete the test file. Till now the following statistics are gathered from my site: Upload/Download is abled, my speed, deletable files (i had changed the configuration to allow deletion of files by the anonymous user).

[2] Wed 13Jun01 14:26:30 - (000019) TYPE A
[6] Wed 13Jun01 14:26:30 - (000019) 200 Type set to A.
[2] Wed 13Jun01 14:26:30 - (000019) PORT 213,51,52,27,9,51
[6] Wed 13Jun01 14:26:30 - (000019) 200 PORT Command successful.
[2] Wed 13Jun01 14:26:31 - (000019) STOR /space.asp
[6] Wed 13Jun01 14:26:31 - (000019) 150 Opening ASCII mode data connection for space.asp.
[4] Wed 13Jun01 14:26:31 - (000019) Receiving file d:\anonftp\space.asp
[4] Wed 13Jun01 14:26:31 - (000019) Received file d:\anonftp\space.asp successfully (4.91 Kb/sec - 2648 bytes)
[6] Wed 13Jun01 14:26:31 - (000019) 226-Maximum disk quota limited to 300000 Kbytes
[6] Wed 13Jun01 14:26:31 - (000019) Used disk quota 2 Kbytes, available 299997 Kbytes
[6] Wed 13Jun01 14:26:31 - (000019) 226 Transfer complete.

This file is included with Grim's Ping companion and will give out information about the ftp server, as described in the tools section.

At the same moment the following log is found from my HTTP server(IIS/5.0) :

2001-06-13 12:26:38 213.51.52.27 - x.x.x.x 80 GET /space.asp |-|0|404_Object_Not_Found 404 -

Of course, if I had used the same directory for both http and ftp, the asp script would have executed and given out further information
about my machine to the scanner. Also note the timing.

[2] Wed 13Jun01 14:26:38 - (000019) DELE /space.asp
[6] Wed 13Jun01 14:26:38 - (000019) 250 DELE command successful.
[5] Wed 13Jun01 14:26:38 - (000019) Closing connection for user ANONYMOUS (00:02:49 connected)

Once the ASP files is not found on the HTTP server, the scanner just deletes the file, and leaves little or no trace of his scan and moves on to the next target.

Problems caused by FTP Pub scanning

Till now this is what I got. Maybe if I wait longer I'd find myself full of Warez and my IP address on some Warez site, IRC channel or bulletin board, with most of my bandwidth being abused, not that nice. Apart from this Corporate sites could be targeted by the software makers and accused as distributing illegal software (Warez) and similar legal issues.

Besides this, there is also the obvious risk of disk space usage, which is limited.

Securing your Server

Securing a server which is vulnerable to this kind of attack it pretty much straight forward for normal configurations. It should be clear that what pub-scanners are exploiting is mis-configuration of ftp (and http) servers. If there is no reason to enable anonymous users to upload files, just disable this functionality. If you need certain users to upload files, you should consider creating a user and password for this purpose, and giving them write access (maybe chroot the user).

Another configuration option would be to create a folder for anonymous connections, which allows uploads but not downloads. This will make downloaders (and probably pub-scanners) jump to the next target and simply dismiss your ftp site.

HTTP and FTP servers should also have use directories. Having an anonymous ftp user upload a CGI script to the http server means that depending on the configuration and web server (we're talking about miss-configured servers here ...) the user will have access to execute possibly malicious code on the target host. This attack was performed on Apache.org back in May 2000, and has probably been around since the use of CGI scripts in HTTP.

Conclusion

Pub scanning seems to have become a favourite and risky pastime for many Warez dealers. This occurred maybe due to the fact that Point and Click Windows Scanners are easily available from professional looking sites. The fact that in just a week two different scanners hit my testing site, seems to indicate an increase in such scanning, and should not be under estimated by the unwary administrator. With the increase in such activity, new tools and features in existing tools will continue to improve the art of pub scanning.

Reference

Honeynet - http://project.honeynet.org/scans/arch/scan8.txt
Cert.org tips - http://www.cert.org/tech_tips/anonymous_ftp_config.txt
Same attacks way back in 1993 - http://www.ciac.org/ciac/bulletins/d-19.shtml
Anonymous FTP abuses - http://www.bris.ac.uk/is/services/networks/anonftp/anonftp2.html
FTP BOUNCE Attack - http://packetstorm.securify.com/UNIX/scanners/hobbit.ftpbounce.txt
Grim's Ping and other tools - http://grimsping.cjb.net/downloads.htm
Omega Scanner - http://www.cybercoderz.com/
Net Knowledge Base - http://www.netknowledgebase.com/
Neuromancer's Tutorial Page! - http://neuro2k.homestead.com/files/index.html
The 'Art' of pub scanning - http://www.jestrix.net/tuts/scan.html
FTP RFC - http://www.faqs.org/rfcs/rfc959.html
Internet Host Requirments - http://www.faqs.org/rfcs/rfc1123.html
SWL FORUM - http://swlforum.cjb.net/
Net knowledgebase forum - http://www.netknowledgebase.com/forum/index.php
Grim's Ping Forum - http://workshops.prohosting.com/grimsping/webboard/webboard.cgi

ANATOMY OF THE WEB APPLICATION WORM

By admin@cgisecurity.com

Disclaimer:

This paper is meant for educational use only and should not be used to create, modify, or produce anything that may damage, or could assist in damaging a computer or network. This paper is theoretical and was not written to give people ideas on creating internet worms, but instead make them aware of the dangers worms produce, along with ways to prevent such a disaster. I will not be held liable for anything you do with the contents of this paper. If you cannot abide by the terms above please stop reading this document now.

Introduction:

In today's internet people are plagued by viruses, and internet worms that slow down production, and make our lives more difficult. Until recently worms had been limited to a few applications or system daemons. This makes fixing the problem rather easy. Of course making a person install the latest product patch is an entirely different story.

Problem:

Today's internet worm as mentioned above relies on particular system daemons to be exploited with either known, or unpublished exploits. Usually a worm will attack a service such as ftp, your mail daemon, or your webserver (see Code Red and Nimda). Of course one way of stopping a worm from spreading is patching your application so infection is not possible. Once your application is patched from the hole in question, you are considered safe. What if a worm had been created to not only infect applications, but find new holes in them as well. This would mean that the worm would not depend on a machine running a specific application, and therefore the worm would be harder to stop. Some people decide not to check for product updates for months for some or all of their applications in use. This can lead to spreading of the worm for longer periods of time. If a machine was infected by custom applications written by either staff at the company, or their consultants ,then it may take some time for a patch to be written and the spread of the worm to stop. On top of this what if the same machine had two applications infected at the same time?

How it will work:

This section is called "How it will work". I say how it "will" work because I feel that within the next two years a worm will be created that will devastate the internet. I believe it will have more of an impact than if Code Red, Nimda, Lovebug, and the "I love you" virus infected the net at the same time. This section goes into detail on what exactly makes this worm different along with its potential for damage. I will not go into details on every way this problem could occur but instead give you an idea of what could happen.

What makes this worm different:

Web Application security has gotten a lot of attention lately as a often overlooked field. A lot of larger website hacks are due to custom holes found in an application on the website. Sometimes sites will use prepackaged scripts, or write their own. Benefits of using a prepackaged web application include getting issued patches when a vulnerability is found. If you use a custom script this depends upon the person within your company to make and install the patches. Sometimes people will use

older scripts that are no longer supported by the vendor. This means that if a vulnerability is found it may go unpatched for many months, or never be patched at all. This leaves attackers with holes in websites they know will go unpatched. This is obviously a serious threat and website administrators should choose applications wisely. Many companies will do whitebox and blackbox testing on web applications to find holes and often times they will create their own patches. People often pay tens of thousands of dollars for such auditing, and usually get what they pay for. Web Application security is a large field and not all of the types of threats have been discovered. Of course people have businesses to run and need these programs to continue business. Some sites will run between one and a few thousand scripts. Probably sixty percent of these applications are affected in one way or another by a hole which could allow server compromise, client information to be leaked, stolen identities, stolen login information, or other serious issues. Currently worms have continued to use a familiar format which I've listed below.

Example of Typical Worm:

1. Scan for hosts running infected product.
    * Check if port is open
    * Check version or even try to infect anyways.

2. Attempt Entry
    * Execute exploit

3. Download/infect machine with code which will continue the spread of the worm.
    * Once in download tools from third party host, or even download more copies of itself.

4. Issuing a payload.
    * Deleting, modification, backdooring, or other related activity.

5. Scan more hosts repeat process.
    * Repeat Step 1.

This formula has worked and will continue to be used by virus and worm writers. Below is a peek at what a web application worm would look like.

Example of a Web Application Worm:

1. Scan for hosts running a webserver.(AKA Open Port 80 is default)
    * Check if port is open. If open Start step B.

2. Crawl the site for web applications/forms.
    * Find applications and mentioned variables for probing.

3. Issue a extensive list of attacks against each application, and each variable in that application found.(Between 1 and 10,000 checks per variable)

Examples:

Application 1
http://host/something.asp?variable=value&variable2=value&variable3=value

Application 2
http://host/something.php?variable=value&so-on=and-so-on

Attack each variable within the application using a prewritten list between one and ten thousand checks that can help continue the spread of the worm. Scanning a large site will probably find between one and a few dozen holes.

4. Continue crawl of site for further applications. Once every application is either infected or found to be safe move onto next host. Repeat step 1.

Just because HostA finds HostB to have 1 application hole doesn't mean it has to stop issuing further checks. The worm could infect that particular application and continue to crawl a site. You may be asking, "why bother the host is compromised?" The answer is simple. If a host is running ten applications and two of them are vulnerable, there's a good chance one of these applications will have some sort of patch issued within a reasonable amount of time. While one application is now no longer vulnerable the worm still has another point of entry. Because the worm doesn't target a particular OS, application, or language flaw, it will continue to spread for long periods of time; perhaps even years. This would affect c/c++, Perl, Python, ASP, XML, .NET, TCL, PHP,Cold Fusion, and other languages making this particularly dangerous. This worm would basically exploit the expertise of the site administrator or coder.

How?:

Tools have been written that crawl a website for sql injection and input validation holes. www.sensepost.com has released a tool to do exactly this, and it can be found at http://online.securityfocus.com/archive/82/257635. If a tool can be written to do this why can't a worm do the same thing except include a harmful payload? Simply put it can happen, and will happen. Lets say a worm starts at 1.x.x.x and for some reason is written to scan the entire subnet. After it has been scanned it continues in order. After the complete subnet run is done, the worm restarts from the beginning. The time it would take for this to happen could be months. People install new web applications on their sites everyday. So there is a good chance on its second sweep it would run into thousands of new applications to exploit. The impact would be obvious, a worm that never dies and continues to find new holes. Obviously writing a worm that randomly tries addresses is preferred since it would be much harder to stop. Even a randomly scanning worm or worm copy would hit the same host twice. People may install new software to stop its spread and they may find themselves reinfected even worse.

Types of Host penetration:

A. Standard .. tricks

Typically web application attacks that allow remote command execution, involve .. tricks to call particular system files to be executed. This would include semicolon, back-tick, and typical file path execution attempts. (Example: ../../../../bin/ls|)

B. Sql injection

Sql injection attacks are gaining popularity among attackers. This attack type is starting to show up in everyday website security. A worm could use sql injection to help further spread itself. Many tools have been written that crawl a website for such holes. If a tool can be written to do this why can't a worm do the same thing except include a harmful payload? Simply put it can happen and will. On top of this many different Database products exist such as MYSQL, DB2, Sybase, etc...

C. Overflows/Format Strings (For a VERY skilled worm writer)

A more advanced programmer 'may' find a way to add a plug in to a worm to find overflows in remote web applications. Of course this wouldn't be an easy task, keeping in mind the different operating systems, chipsets, and offsets one would have to chew through. This probably won't happen anytime soon but definitely "could" happen if one spent enough time. An immediate problem would

be the size of the worm and the speed in which it spreads/tests applications. A worm like this would probably end up killing more processes/breaking things then spread and isn't practical but it is worth a mention.

Impact of the worm:

Obvious impact would be halt of commercial services, bandwidth charges, system performance, time to patch (Time = Money), upgrades, cost of creation of patches, and various other nasties. Of course security companies would profit from such a incident , and who knows, perhaps some of these worms are created by security companies to increase production/sales.

Fix Suggestions:

IDS:
A problem with such a complex worm is that no "Lone" signature would exist. Perhaps some sort of IDS would check for 2 signatures to help cut back on false alarms. If 2 checks are issued then block offending host.

Application Security Wrappers:
Wrappers are programs that watch the behavior of a program to see if it is trying to do something that it shouldn't. Popular wrappers in wide scale use are "CGIWRAP", and apaches "SUexec" option. These programs/options will execute the scripts and if offending behavior is found, deny it from continuing. This is one excellent solution to the problem.

Router blocks:
Some routers have the ability to deny a packet based on its contents. As mentioned in the "Fingerprinting Port80 Attacks" series, attacks leave certain fingerprints. By adding these signatures into your router block you can stop attacks before they even happen. Obviously false alarms will come into play and this is not recommended for everyone. For those who do decide to use these blocking methods remember that adding such signatures are production dependant and need careful study. During the "code red attacks" many people put these types of blocks into play. A problem with such as worm is choosing the right signature. Also after detecting such a signature denying all packets from that host. These signatures would depend on worm structure.

Conclusion:

I didn't write this paper to give people malicious ideas. I wrote this to show a large gaping hole that exists that will only be dealt with if each person does their small part. Currently website administrators and programmer behaviors leave such a threat open. Unfortunately no single fix exists to stop such a problem, so everyone must watch what types of programs he/she uses along with who the vendor is. Promptly responding to security threats also cuts down on such possibilities. By stopping the behavior of choosing poor applications or vendors we can stop such a threat. I wrote this paper on a Sunday afternoon in March and for the past week have been deciding whether the risks outweigh the benefits. I feel that by keeping this information limited we will surely become devastated by such a threat in the near future. This article also may be useful to the people who do care about security, and are willing to help cut down on such threats. If you have any questions, flames, or suggestions email me at admin@cgisecurity.com.

References and Links Mentioned Within:
http://cgiwrap.unixtools.org/
http://httpd.apache.org/docs/suexec.html
http://www.phphelp.com/tutorial/using-a-script-wrapper.html
http://www.cgisecurity.com/papers/
http://online.securityfocus.com/archive/82/257635

Published to the Public March 2002

VERIZON TELECONFERENCING

By k00p$ta Phr34k and ic0n

Before we begin this file I (ic0n&k00p$ta) are not going to give you any info on setting up the conference. For a few reasons but it's not hard at all the setup once since everyone @ verizon is crazy or just dumb minus a selected few. (they know who they are) Now on with the file.

Verizon now offers a new service, Conference Connections.These Conferences's are reservation-less, which means around the clock availability. The Conference is available 24 hours a day, 7 days a week, and 365 days out of the year. This makes conferencing very easy. Thanks Verizon!

There's 2 ways to dial into a verizon conference.
1.Toll Free dial in number (866-441-2942)
2. Direct (972-717-2043) Npa 972 is in Texas

There are no setup fees, no cancellation fees, and no monthly charges. Which mean you can setup a teleconference and your victim will not even know he's got a teleconference being billed to him. The minutes your participants used are logged separately logged by differnt ports. There are 20 of these ports but I'm sure there is a way to get more. Anyways the minutes are added together to simplify the subscriber's bill, in addition are required taxes. There is a separate bill for toll free service as well.

States that need to use the direct number to the conference:
1.Alaska
2.Delaware
3.Maryland
4.New Jersey
5.New Hampshire
6.Virginia
7.Vermont
8.Washinton D.C.
9.West Virginia
*Once again the direct number is 972-717-2043.

The resoning behind the direct numbers is that Verizon provides long distance services for calls originating in most states outside the mid-Atlantic and new England states. Until government approval is obtained, Verizon cannot carry long distance in the states listed above. Verizon is in the works on getting the necessary states and federal permissions to offer long distance in every state.

Rates Cents per minute per port

	Until 3/30/02	Normal
Toll Free	$0.22	$0.31
Direct	$0.09	$0.18

Feature Descriptions
Announcements for Entry and Exit At your option, the reservation-less Conference Connections system can sound a tone or have silence when participants enter or exit a conference.

Attendant Request
The Subscriber or Participants can request attendant assistance for private or group consultation. The person requesting assistance remains in the conference until the attendant handles the request.

Conference Continuation

This feature allows the subscriber to exit a conference after it begins without disconnection the participants and must be activated for each conference call. *Note The systems automatically defaults to end the conference call when the subscriber disconnects.*

Conference Lock/Unlock

This feature lets subscriber lock a conference once all parties are present to keep the conference private. Attendants cannot enter locked conferences, but can ring the conference requesting that the subscriber unlock for attend entry.

Help Menu

Help with using conference commands is available to every conference Subcriber and Participant. The system plays a private help message to the requester that list the available features and their associated touch-tone (dtmf) commands.

Mute/Un-mute

The Subscriber can collectively mute or un-mute all lines in the conference except for the subscriber's line. The participants can mute and un-mute there own lines to help control distractions and interruptions.

Participant Count

The system automatically tracks the number of participants on a conference. Any Subscriber or Participant can check the number of people in conference at any time. The system announces the count privately to the requester.

Quick Start

As a rule, conferences do not begin until the subscriber the conference. However your account can be configured to allow the subscriber to use this feature so that begins as soon as the first participant arrives. In this scenario, Participants who arrive before the subscriber may talk to one another before the conference actually begins. Though the quick start features offers less security, it allows unplanned meetings to occur whenever needed or permits conferencing when the subscriber is unavailable to start the conference.

Features
Subscriber Conference Commands
This is how you Begin a conference:
1. Dial into conference system
2. Enter Pass code, then the # (pound) key
3. Then Press the * (star) key
4. Enter Subscriber Pin (4 digits)
5. Press 1 to start the conference or press 2 to change account options.

To Change Account Options:
Press 1 to chance subscriber pin
Press 2 to configure roll call options
Presses 3 to change quickly start options
Press 4 to change auto continuation options

Conference Control options (while in conference)
Press *0 to speak privately with an operator

Press 00 to request an operator to join the conference
Press *4 to lock conference
Press *5 to unlock the conference
Press *6 to mute your line
Press *7 to un-mute your line
Press *8 to allow the conference to continue after you disconnect
Press *9 to privately play a list of participants on conference
Press *# to hear the number of participants in the conference
Press ## to mute all lines except the subscriber
Press 99 to un-mute all lines
Press ** to play this list of commands

How to end a Conference
Say whatever then hang up the phone a short message will be played for them and then disconnects them.

We also need to thank verizon for be so dumb and giving us all this information to write this article. Shout Outs....Lucky225, Dark_Fairytale, The Borish One,Xenocide, Cuebiz, MaddjimBeam, Whit3rav3n, Reaver,Captain_B, Mr. Poop, RBCP, Everyone Who was on $kytel back in 96-97...well okay only some people from skytel and everyone else we know.

Vertical Service Codes

By: ^CircuiT^

All a vertical service code is really nothing more then line services that you can access by picking up your phone and hiting *XX.

Here i have listed them all:

*00 - Inward Voice Activated Services (English)
*01 - Inward Voice Activated Services (French)
*02 - Deactivation/Activation of In-Session Activation (ISA)on a per line basis
*03 - Deactivation of In-Session Activation (ISA) on a per call basis
*2X - Reserved for expansion to 3digit VSCs
*228 - Over-the-Air Service Provisioning
*3X - Reserved for expansion to 3-digit VSCs
*40 - Change Forward-To Number for Customer Programmable Call Forwarding Busy Line
*41 - Six-Way Conference Calling Activation
*42 - Change Forward-To Number for Customer Programmable Call Forwarding Don't Answer
*43 - Drop last member of Six-Way Conference Call
*44 - Voice Activated Dialing
*45 - Voice Dialing Extended Dial Tone
*46 - French Voice Activated Network Control
*47 - Override Feature Authorization
*48 - Override Do Not Disturb
*49 - Long Distance Signal
*50 - Voice Activated Network Control
*51 - Who Called Me?

*52 - Single Line Variety Package (SVP) - Call Hold
*53 - Single Line Variety Package (SVP) - Distinctive Ring B
*54 - Single Line Variety Package (SVP) - Distinctive Ring C
*55 - Single Line Variety Package (SVP) - Distinctive Ring D
*56 - Change Forward-To Number for ISDN Call Forwarding
*57 - Customer Originated Trace
*58 - ISDN MBKS Manual Exclusion Activation
*59 - ISDN MBKS Manual Exclusion Deactivation
*60 - Selective Call Rejection Activation
*61 - Distinctive Ringing/Call Waiting Activation
*62 - Selective Call Waiting
*63 - Selective Call Forwarding Activation
*64 - Selective Call Acceptance Activation
*65 - Calling Number Delivery Activation
*66 - Automatic Callback Activation
*67 - Calling Number Delivery Blocking
*68 - Call Forwarding Busy Line/Don't Answer Activation
*69 - Automatic Recall Activation
*70 - Cancel Call Waiting
*71 - Usage Sensitive Three-way Calling
*72 - Call Forwarding Activation
*73 - Call Forwarding Deactivation
*74 - Speed Calling 8 - Change List
*75 - Speed Calling 30 - Change List
*76 - Advanced Call Waiting Deluxe
*77 - Anonymous Call Rejection Activation
*78 - Do Not Disturb Activation
*79 - Do Not Disturb Deactivation
*80 - Selective Call Rejection Deactivation
*81 - Distinctive Ringing/Call Waiting Deactivation
*82 - Line Blocking Deactivation
*83 - Selective Call Forwarding Deactivation
*84 - Selective Call Acceptance Deactivation
*85 - Calling Number Delivery Deactivation
*86 - Automatic Callback Deactivation
*87 - Anonymous Call Rejection Deactivation
*88 - Call Forwarding Busy Line/Don't Answer Deactivation
*89 - Automatic Recall Deactivation
*90 - Customer Programmable Call Forwarding Busy Line Activation
*91 - Customer Programmable Call Forwarding Busy Line Deactivation
*92 - Customer Programmable Call Forwarding Don't Answer Activation
*93 - Customer Programmable Call Forwarding Don't Answer Deactivation
*94 - Reserved For Local Assignment
*95 - Reserved For Local Assignment
*96 - Reserved For Local Assignment
*97 - Reserved For Local Assignment
*98 - Reserved For Local Assignment
*99 - Reserved For Local Assignment

HELP SUPPORT HACKER'S DIGEST

Issue 1

Summer 2001

Issue 2

Fall 2001

Issue 3

Winter 2002

Send $5.00 (per issue) check or money order to:

Hacker's Digest
P.O. Box 71
Kennebunk, ME 04043

Hacker's Digest

Pure Uncut Information