802.1x

WPA(2)

CHAP

RADIUS

Wireless Alphabet Soup

AES

TKIP

802.11i

CBC-MAC

WN

Mixing Up A Secure Wireless Network

EAP

TSN

WPA(1)

AP

EAPOL

WEP

PEAP

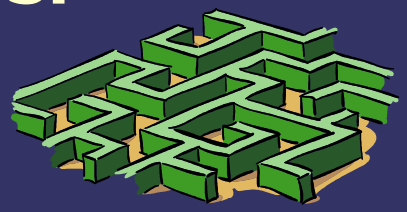
PAP

SP

CCMP

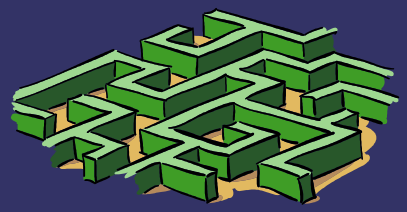
RSN

AS



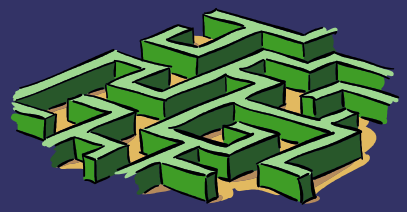
Copyright

Adam Tauno Williams (awilliam@whitemice.org)
http://www.whitemiceconsulting.com



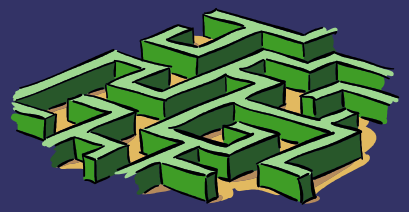
The Problem

- Wireless networks are especially prone to attack and compromise.
  - Need a way to authorize users & devices to use the network.
  - Traffic over the wireless connection must be secured.
    - Must be simple to use.
    - Must be broadly supported.
    - Must integrate with existing services.
    - Must be robust and reliable.
- The answer is WPA
  - 802.1x/EAP + RADIUS + TSN/RSN
  - aka “802.11i”



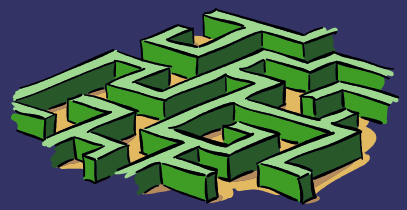
What is WPA?

- WPA was born out of frustration with the slow moving 802.11i standard.
  - WEP is broken, we need a solution NOW!
  - WPA is intended to be 802.11i compatible.
- WPA is "Wi-Fi Proctected Access"
  - WPA(1), also called TSN, is: TKIP + 802.1X
    - TSN is "Transitional Secure Network"
  - WPA(2), also called RSN is: CCMP + 802.1x
    - RSN is "Robust Secure Network"



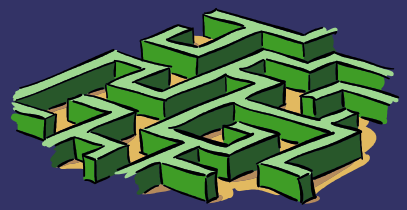
What is 802.1x

- 802.1x is “port based authentication”
  - In this context a “port” is a single attachement point to a network.
    - The port on an Ethernet hub.
    - The association between a SP and an AP
    - A VPN connection
    - etc....
  - Think '*gatekeeper*'
    - The gate is closed until it is opened.
    - The gate is either open or closed.
  - Port based authentication uses EAP
    - EAP was originally designed for authenticating dial-up users over PPP



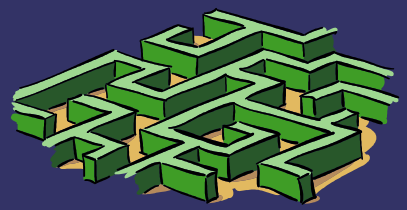
What Is EAP

- EAP is "Extensible Authentication Protocol"
  - The protocol used between client and the network access device (switch, AP, etc...)
    - The wire protocol in the case of wireless clients authenticating to an AP is EAPOL
      - EAPOL is "EAP Over LAN"
    - EAP is **NOT** an IP Protocol
  - EAP is a way of encapsulating authentication requests.
    - An enormous variety of authentication mechanisms can be encapsulated over EAP.
      - PAP, CHAP, M$-CHAP, OTP, Kerberos, Public Key, etc...
      - So just "au*thentication over EAP*" means almost nothing.



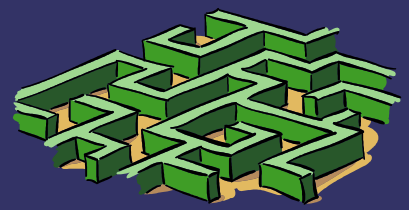
Encryption Terms

- TKIP
  - The “Temporary Key Integrity Protocol” is an encryption protocol based on RC4.
    - A 'temporary' fix to the WEP train wreck.
      - Meant to be compatible with legacy 802.11 hardware.
      - Changes keys periodically.
      - Uses a 48 bit vector vs. WEPs 24 bit vector.
- CCMP
  - The “Counter Mode with CBC-MAC” is a new encryption protocol based on AES.
    - AES is “Advanced Encryption Support”
    - Demands much more CPU horsepower than RC4.
      - To support CCMP hardware needs to be designed to support CCMP, usually involves a dedicated coprocessor.



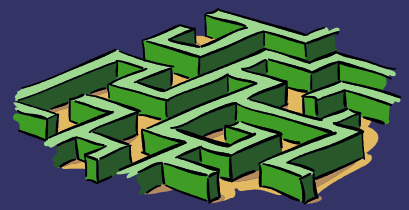
What is 802.11i

- 802.11i is a standard for constructing robustly secure networks.
  - Requires AES encryption
    - Will not work with older [aka most current] hardware.
    - Encyrpts the entire frame
      - WPA(1) only encrypts the payload
  - WPA2 is 802.11i



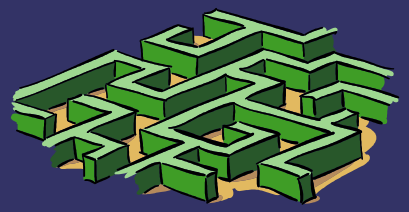
What Is RADIUS

- RADIUS is the "Remote Authentication Dial In User Service" developed for ISPs to authenticate users.
  - RADIUS was designed to provide "AAA"
    - Authentication
    - Authorization
    - Accounting
  - RADIUS is an open standard.
    - http://www.ietf.org/rfc/rfc2865.txt
  - RADIUS usually front-ends another authentication service.



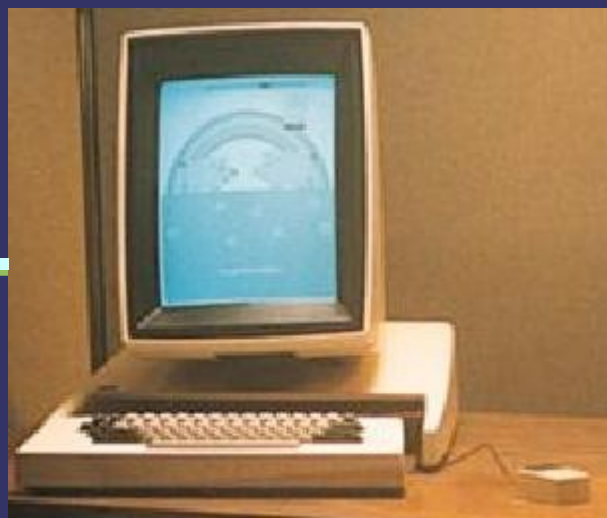
Terms & Acronyms

- Wireless Node [WN]
  - The device requesting network access.
    - My laptop
- Supplicant [SP]
  - The software on the client that manages authentication and authorization.
- Authenticator
  - The software performing the authentication,
    - Translating the EAP frames into RADIUS requests.
    - Usually this is hosted on the Access Point [AP]
- Authentication Server [AS]
  - The service or device that is performing the act accepting or rejecting user credentials.
    - Our FreeRADIUS server.



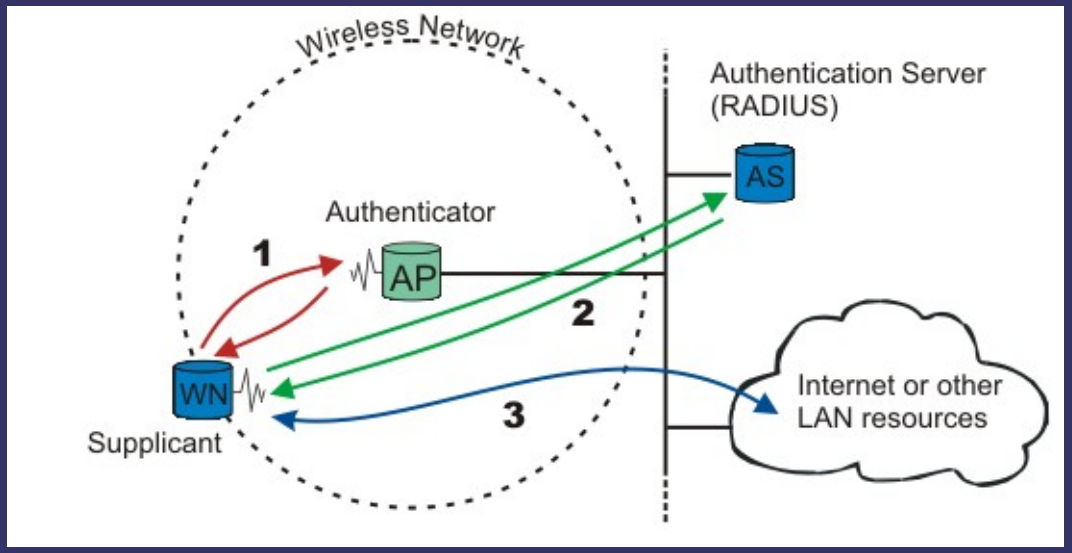
A Visual


Supplicant
WN
EAP
TKIP or CCMP
Authenticator
Access Point
AP
RADIUS
AS
Ethernet
Authentication Service
(RADIUS Server)

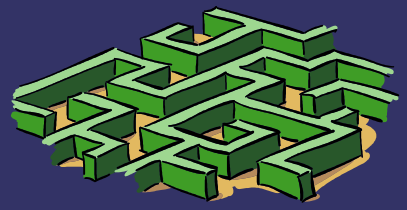


Authenticating

Image from http://distributions.linux.com/howtos/8021X-HOWTO/intro.shtml

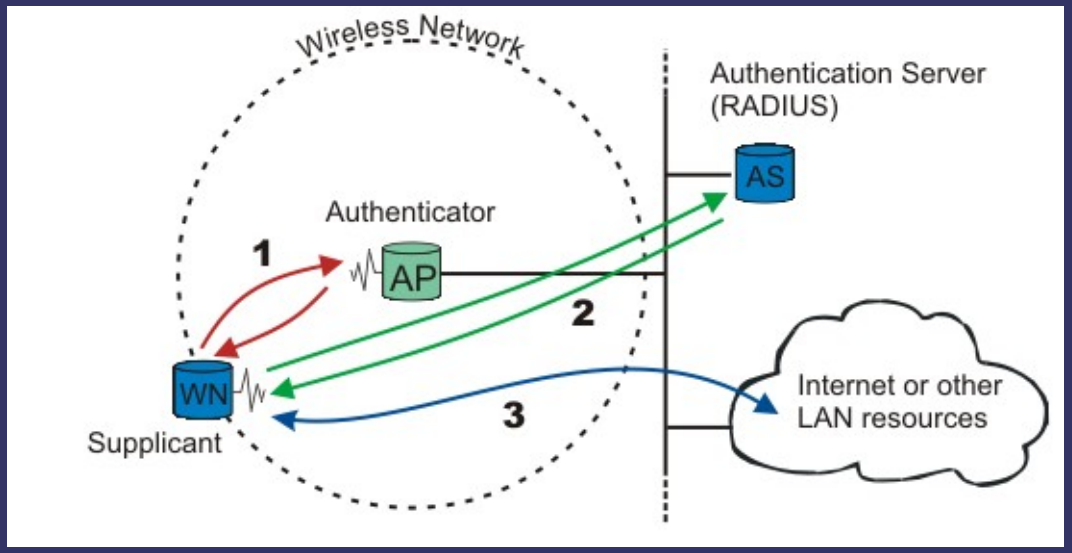


- Step #1
  - WN requests access
    - Only EAP traffic is permitted (No IP!)
    - This is an exchange of identity.

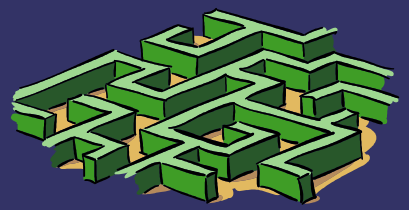


Authenticating

Image from http://distributions.linux.com/howtos/8021X-HOWTO/intro.shtml

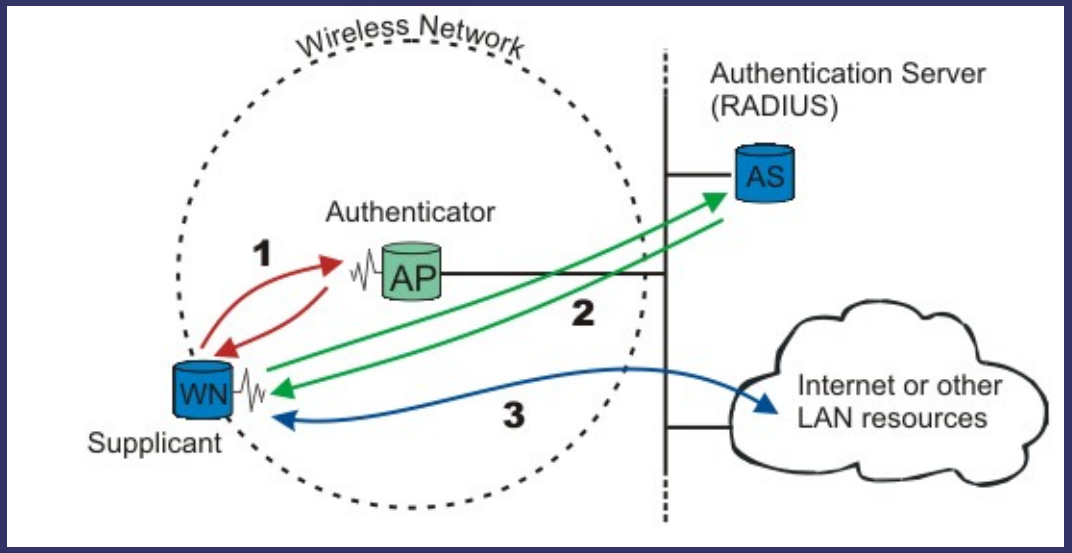


- Step #2
  - WN authenticates
    - The AP acts as a translator and relay
      - EAP (EAPOL) <-> RADIUS
        - The AP knows NOTHING about the authentication process or mechanism.

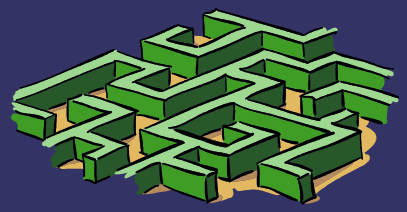


Authentication

Image from http://distributions.linux.com/howtos/8021X-HOWTO/intro.shtml

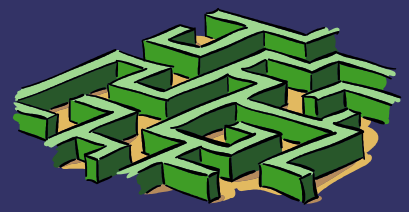


- Step #3
  - The port is openned
    - The AS responds with success and the port is opened.
      - WN can now proceed with acquiring an IP address.



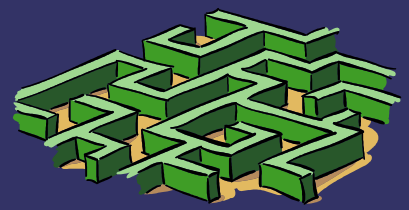
FreeRADIUS

- FreeRADIUS is a full-featured enterprise ready RADIUS service provider (AS).
  - http://www.freeradius.org
  - GPL
  - Full support for RFC 2865 and 2866
  - Specific support for hardware from more than 50 vendors.
  - Supports a myriad of EAP encapsulated authentication methods.
  - Provided by main stream distributions.



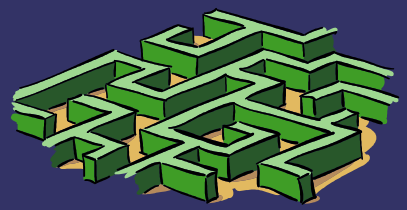
FreeRADIUS Authorization

- FreeRADIUS supports the following authroriza-tion data sources:
  - Files
    - Text
    - DB / DBM
  - LDAP
    - OpenLDAP
    - Novell NDS
    - Sun One
    - Any LDAPv3 compliant DSA
  - Local Executable
  - Perl script
  - Python script
- SQL Database
  - Oracle
  - PostgreSQL
  - Sybase
  - IBM DB2
  - MySQL
  - ODBC
    - iODBC
    - uniXODBC



FreeRADIUS Mechanisms

- FreeRADIUS can authorize users using a variety of methods
  - PAP (PAM, LDAP, Files)
  - CHAP, M$-CHAP, M$-CHAPv2
  - NTLM (M$-DC, Samba-DC, LDAP)
  - Proxy to another RADIUS server
  - CRAM
  - SIP Digest
  - Nestscape-MTA-MD5
  - Kerberos
  - X9.9 (CRYTO Card)
- Custom mechanisms can also be developed.
  - Perl
  - Python



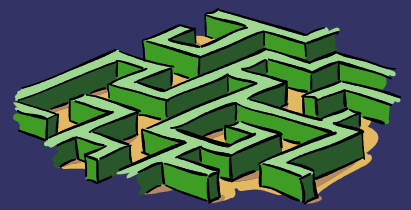
FreeRADIUS EAP

- FreeRADIUS supports a variety of EAP mechanisms.
  - EAP-MD5
  - Cisco LEAP
  - EAP-MSCHAP-V2
  - EAP-GTC
  - EAP-SIM
  - EAP-TLS
  - EAP-TTLS
  - EAP-PEAP

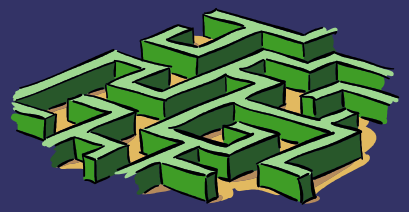
These mechanisms are considered weak.

Requires PKI (Ugh!)

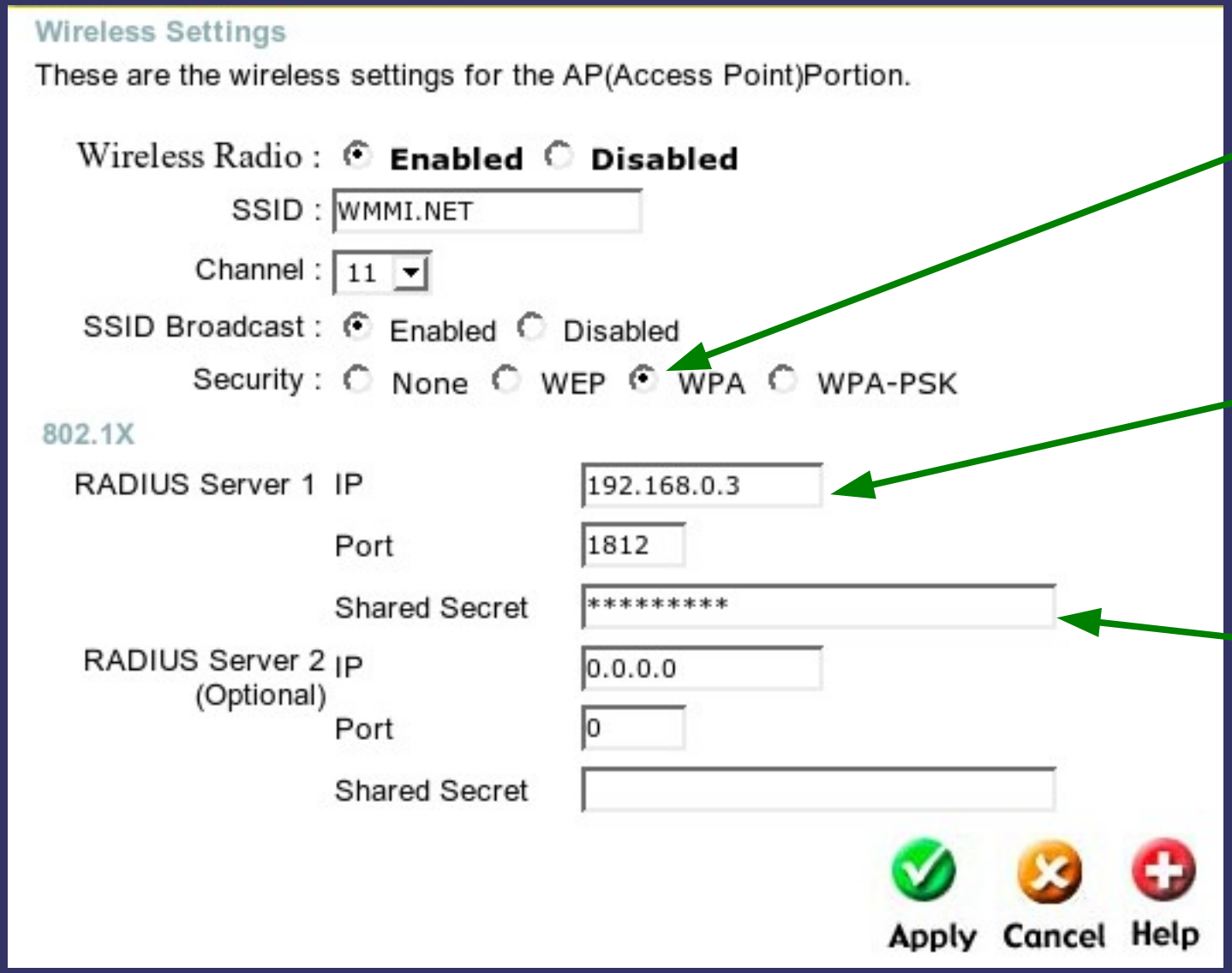
These mechanisms require OpenSSL.



Setting Up A WPA Network



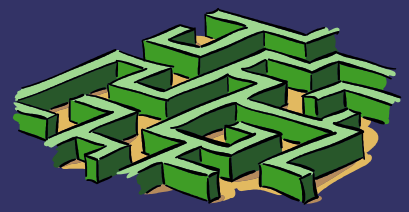
Setting Up Authenticator



Configure your AP to use WPA

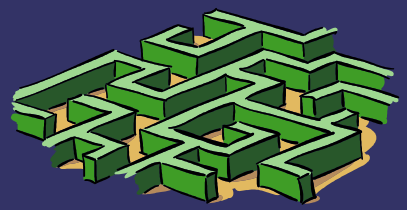
Tell the AP where the RADIUS server is.

Enter a secret passphrase.



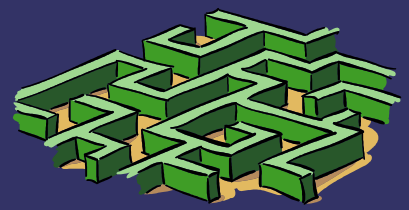
Select An EAP Mechanism

- We are using EAP-PEAP
  - PEAP is “Protected EAP”
  - Does not require PKI
  - User enters a username and a password
  - Uses M$-CHAPv2
    - Password never crosses the wire.
    - Can authenticate against an NT hash of the user's password.
      - Samba DCs have this credential.
      - Also will work with an NTLM mechanism.
  - Very widely supported
    - Natively supported by that other operating system.
      - Adding ZERO software to Win32 clients was a design requirement.
    - Works well with Open Source supplicants.



Configuring FreeRADIUS

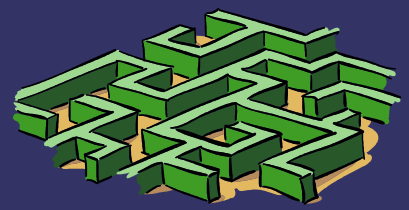
- Install FreeRADIUS
  - Configuration files are in /etc/raddb
    - clients.conf
      - Enumerates the authenticators
    - eap
      - Configures encryption and EAP method.
    - radiusd.conf
      - Overall server configuration
    - users
      - Enumerates users or user defaults
    - ldap.attrmap
      - Maps RADIUS attributes to LDAP attributes
- Open UDP Ports 1812/1813



Configuration structure

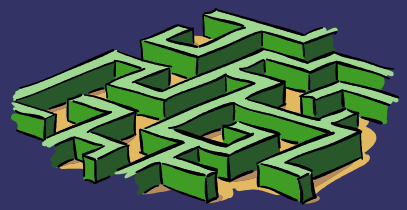
- FreeRADIUS configuration files are nested.
  - radiusd.conf includes the other configuration files.
    - $INCLUDE ${confdir}/eap.conf
  - Nested levels are in the form of:

```
name {
    directive = value
    name {
        directive = value
        name {
            ...
            }
        }
    }
}
```



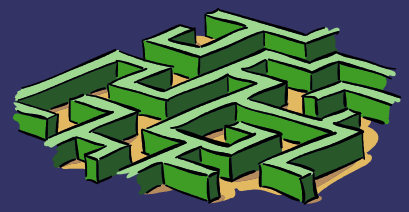
radiusd.conf

- radiusd.conf contains a variety of global configuration directives:
  - bind_address = 192.168.3.1
  - $INCLUDE ${confdir}/clients.conf
  - snmp = no
  - $INCLUDE ${confdir}/snmp.conf
  - max_requests = 1024
  - port = 1812
  - *etc...*



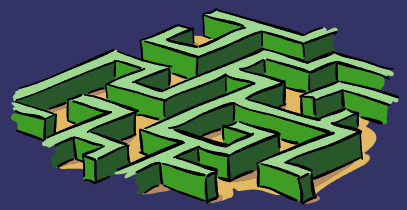
Modules & Stacks

- The “modules” section defines the 'meat' of the configuration.
  - Within “modules” is:
    - mschap
    - eap
    - ldap
    - files
    - *etc...*
  - Stacks define the modules that will be run at each event.
    - Each entry is a module defined in “modules”
    - Modules defined in each stack is run in order.
    - authorization
    - authenticate
    - *etc...*



Modules

- You only need to configure the modules you are going to use.
  - eap
    - Configures the EAP functionality
  - mschap
    - Configures M$-CHAP options.
  - ldap
    - Configures LDAP options

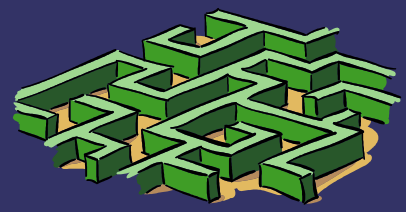


Stacks

- authorization
  - preprocess
    mschap
    suffix
    ldap
    eap
    files
- authenticate
  - Auth-Type MS-CHAP {
        mschap
    }
    eap
- preacct
- accounting
- ....

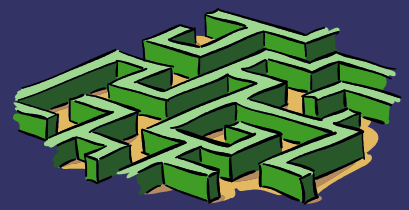
Need to modify these.

Leave this one out at first



mschap

- mschap {
  authtype = MS-CHAP
  use_mppe = yes
  require_encryption = yes
  require_strong = yes
  with_ntdomain_hack = no
  }
- “use_mppe” has to be “on”.
- You can use the “with_ntdomain_hack” if you need to strip a domain off the provided user name.



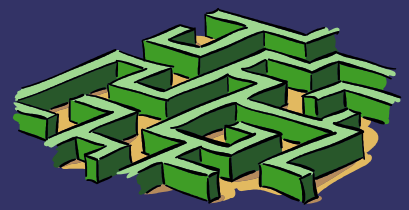
eap

```
eap {
        default_eap_type = peap
        timer_expire     = 60
        ignore_unknown_eap_types = no
        cisco_accounting_username_bug = yes
        tls {
                private_key_password = **********
                private_key_file = /etc/ssl/private/littleboy-key-v2.pem
                certificate_file = /etc/ssl/certs/littleboy-cert-v2.pem
                CA_file = /etc/ssl/certs/MorrisonIndustries-cacert-v2.pem
                dh_file = ${raddbdir}/dh
                random_file = /dev/urandom
                check_crl = no
                }
        peap {
                default_eap_type = mschapv2
                }
        mschapv2 {
                }
}
```

Set EAP mechanism

SSL Setup

EAP Mechansim Options



Add Authenticators

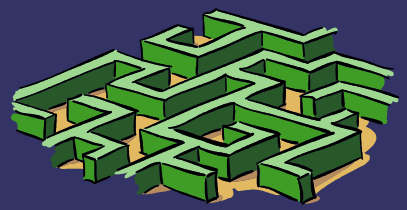
- The AP is a client of the AS.
  - The AP and the AS shared a secret called the "shared secret".
    - The clients and their secrets are defined in clients.conf
    - This secret is used to encrypt and sign packets between the AP and AS.
  - You can also set a nastype which is used to help interoperability with proprietary clients.

Making a client entry for localhost (used for testing)

```
client 127.0.0.1 {
        secret          = testing123
        shortname       = localhost
        nastype         = other
}
```

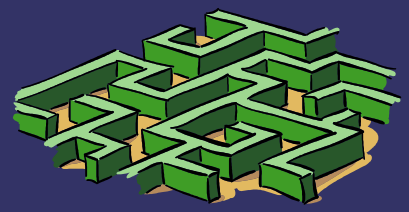
Making a client entry for AP

```
client 206.915.906.293 {
        secret          = bonbon
        shortname       = yippityskippity
        nastype         = other
}
```



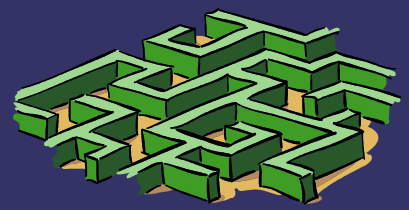
Adding Users

- By default supplicants are defined in the users file.
  - This is defined in the files {...} module
  - An LDAP DSA, RDBMS, DC, or KDC can also be used for authentication.
    - Use users to get our EAP/RADIUS working, then define you alternative authentication source.
    - Add your '*real*' authentication source AFTER you get RADIUS working.
      - “ldap”



users file

- The users file is read from top to bottom.
  - The special DEFAULT user can be used to define attributes for subsequent users.
  - The first line of a users entry specifies criteria that must be matched.
  - Subsequent lines define attribute value pairs to be returned to the client.
    - There is a large number of attributes that can be assigned to an account.



users

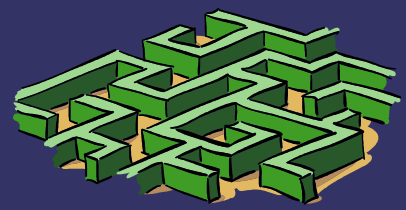
```
DEFAULT Auth-Type = Local
        Reply-Message = "Hello, %u",
        Fall-Through = Yes
DEFAULT Service-Type == Framed-User
        Framed-IP-Address = 255.255.255.254,
        Framed-MTU = 576,
        Service-Type = Framed-User,
        Fall-Through = Yes
DEFAULT Framed-Protocol == PPP
        Framed-Protocol = PPP,
        Framed-Compression = Van-Jacobson-TCP-IP
awilliam        Password == "eaptest"
```

Check Items

Reply List

Username

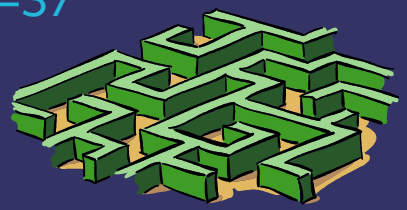
Clear text password.



radtest

- The radtest utility is used to test authentication to your RADIUS server.

```
# radtest awilliam eaptest localhost 10 testing123
Sending Access-Request of id 74 to 127.0.0.1:1812
        User-Name = "awilliam"
        User-Password = "eaptest"
        NAS-IP-Address = localhost.localdomain
        NAS-Port = 10
rad_recv: Access-Accept packet from host 127.0.0.1:1812, id=74, length=37
        Reply-Message = "Hello, awilliam"
$ radtest awilliam badpassword localhost 10 testing123
Sending Access-Request of id 78 to 127.0.0.1:1812
        User-Name = "awilliam"
        User-Password = "badpassword"
        NAS-IP-Address = localhost.localdomain
        NAS-Port = 10
rad_recv: Access-Reject packet from host 127.0.0.1:1812, id=78, length=37
        Reply-Message = "Hello, awilliam"
```

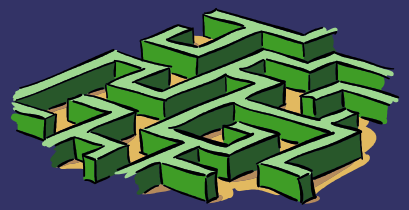


wpa_supplicant

- SuSe 10.0 provides wpa_supplicant.
  - wpa_supplicant -d -D madwifi -i ath0 -c {*config file*}
    - ctrl_interface_group=root
      network={
      ssid="WMMI.NET"
      scan_ssid=1
      key_mgmt=WPA-EAP
      eap=PEAP
      pairwise=CCMP TKIP
      group=CCMP TKIP
      identity="awilliam"
      password="********"
      phase1="peaplabel=0"
      phase2="auth=MSCHAPV2"
      }

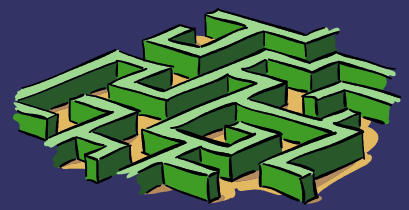
Your wireless interface

Your wireless chipset



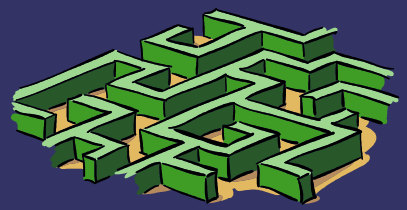
WPA drivers

- wpa_supplicants supports the following chipsets:
  - hostap = Host AP driver (Intersil Prism2/2.5/3) [default]
  - hermes = Agere Systems Inc. driver (Hermes-I/Hermes-II)
  - madwifi = MADWIFI 802.11 support (Atheros, etc.)
  - atmel = ATMEL AT76C5XXx (USB, PCMCIA)
  - wext = Linux wireless extensions (generic)
  - ndiswrapper = Linux ndiswrapper
  - broadcom = Broadcom wl.o driver
  - ipw = Intel ipw2100/2200 driver
  - wired = wpa_supplicant wired Ethernet driver
  - bsd = BSD 802.11 support (Atheros, etc.)
  - ndis = Windows NDIS driver
- wpa_supplicant home page:
  - http://hostap.epitest.fi/wpa_supplicant/



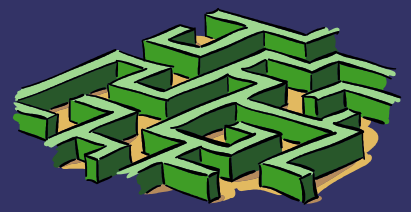
ldap

- ldap {
  server = "localhost"
  identity = "*bindDN*"
  password = *bindPassword*
  basedn = "dc=whitemice,dc=org"
  filter = "(&(objectclass=account)(uid=%{Stripped-User-Name:-%{User-Name}}))"
  base_filter = "(objectclass=radiusprofile)"
  start_tls = no
  default_profile = "cn=Default Profile,ou=RADIUS,ou=Sub...."
  dictionary_mapping = ${raddbdir}/ldap.attrmap
  ldap_connections_number = 5
  edir_account_policy_check=no
  groupname_attribute = cn
  groupmembership_filter = "(&(objectClass=GroupOfNames)(member=%{Ldap-UserDn}))"
  timeout = 4
  timelimit = 3
  net_timeout = 1
  }



ldap.attrmap

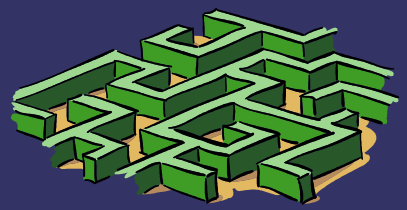
- The ldap.attrmap file maps RADIUS attributes to LDAP attributes
  - checkItem LM-Password sambaLMPassword
  - checkItem NT-Password sambaNTPassword
  - replyItem Idle-Timeout radiusIdleTimeout
  - replyItem Session-Timeout radiusSessionTimeout
- dn: cn=Default Profile,ou=RADIUS,ou=SubSystems,...
  objectClass: top
  objectClass: radiusprofile
  objectClass: ipService
  cn: Default Profile
  ipServicePort: 1812
  ipServiceProtocol: udp
  radiusIdleTimeout: 1800
  radiusSessionTimeout: 28800



NTLM

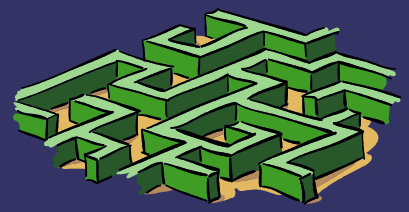
- An alternative to LDAP is to use NTLM authentication to your CIFS DC
  - mschap {
    authtype = MS-CHAP
    use_mppe = yes
    require_encryption = yes
    require_strong = yes
    with_ntdomain_hack = no
    ntlm_auth = "/usr/bin/ntlm_auth --request-nt-key --username=%{Stripped-User-Name:-%{User-Name:-None}} --domain=*BACKBONE* --require-membership-of=*BACKBONE\\wireless* --challenge=%{mschap:Challenge:-00} --nt-response=%{mschap:NT-Response:-00}"
    }

All one long line.



Testing NTLM Auth

- You can run the ntlm_auth command manually to make sure that it works
  - /usr/bin/ntlm_auth --username=adam \
    --domain=BACKBONE
    --password=**********
    --require-membership-of=BACKBONE\\wireless
  - Should return:
    NT_STATUS_OK: Success (0x0)

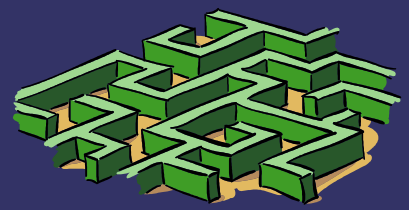


users

- Remove users from the users file
  - DEFAULT Ldap-Group == "WPA Wireless"

    DEFAULT Fall-Through = 1

    DEFAULT Auth-Type := Reject
    Reply-Message = "Please call the helpdesk."
  - Try authenticating against the RADIUS server now.
- Restart the RADIUS service
  - rcradius restart



Debugging FreeRADIUS

- To debug FreeRADIUS run the server with the “-X -A” options.
  - Server will run in the foreground.
  - Will write enormous amounts of information to standard out.
- Server writes logs to /var/log/radius/radius.log
- Accounting information is written to /var/log/radius/radacct/{*client-IP*}/{*session-id*}