The Center for Internet Security - Critical Controls

Security Basics On a Zero Dollar Budget

Rob VandenBrink
rob@coherentsecurity.com
Good Afternoon – whoami $me =

• Rob VandenBrink, Coherentsecurity.com
• In the industry since mainframe / PC1 / DOS 1.0 days (1981)
• Blog at the Internet Storm Centre (https://isc.sans.edu)
• Editor for Center for some CIS Security Benchmarks: Palo Alto, Cisco Nexus, Cisco FirePower
• $dayjob?
  • Networking, Virtualization, Security and Automation / Orchestration.
  • Design, Build, Fix, Audit, Attack, Defend
CIS Critical Controls

• 20 Critical Controls that highlight how Blue Teams can better secure most environments. [https://www.cisecurity.org/controls/](https://www.cisecurity.org/controls/)

• Hundreds of sub-controls under the 20 (great reference at [https://www.auditscripts.com/download/4229/](https://www.auditscripts.com/download/4229/))

• More or less independent of Operating System or Vendor

• Roughly in order of impact

• Today we’ll cover off how several of these controls can be used to secure an organisation, **focusing on using PowerShell to deliver**
CIS Critical Controls – “Basic” Controls

• Covering the basic controls generally means that your organization is no longer the slowest gazelle in the herd

• This is the “If you are not doing this, you should start” list
  • Hardware Inventory
  • Software Inventory
  • Continuous Vulnerability Management
  • Control Admin Rights
  • Set Secure Configuration Templates for Servers, Workstations, Laptops, Mobile
  • Logging
CIS Critical Controls - Foundational

• The Best Practices/ Smart Move list:
  • Email and Browser Protections
  • Malware Defenses
  • Limit Network Ports, Protocols and Services (network segmentation)
  • Data Recovery Capabilities
  • Secure config for Network Devices – Firewalls, Routers, Switches etc
  • Boundary Defense
  • Data Protection
  • Controlled Access (Need to Know)
  • Wireless Access Control
  • Account Monitoring
CIS Critical Controls - Organizational

• This list focuses more on people and processes
• These tend to involve training, transitioning or hiring people ($BUDGET)
• More difficult and slower to implement in most organizations
  • Security Awareness Training
  • Application Software Security
  • Incident Response
  • Penetration Test / Red Team Exercises (Threat Hunting)
CC#1 – Hardware Inventory

• Easy to inventory hardware in AD – PowerShell is your friend
• You can’t protect / patch / backup what you don’t know is there
• Adding an NMAP scan allows you to inventory non-AD components, or at least get a start on that
• Looking ahead to controls down the list – keeping Student / Guest / SCADA / ICS / Medical Equipment networks segregated and firewalled away from production makes this much easier!
• Active scanning Student / Guest owned gear ≈ legal issues
• Scanning Medical gear in particular = liability issues
CC#1 – Hardware Inventory

• A one-liner in PowerShell, then dump to Excel, db or whatever

• For many situations, no need to buy anything – “AD knows”:

  # OS / Stale Computer Acct inventory from AD
  $pcs = Get-ADComputer -filter * -property Name, OperatingSystem, OperatingSystemVersion, LastLogonDate, IPV4Address
  $pcs | export-csv -path ./AD-OSInventory.csv

• Can also “loop” through AD member list (or a subnet for non-domain computers) and query hw directly with WMI

• Unless you are creating a target list, inventorying EOL OS’s just feeds the dumpster fire (XP, S2K3, S2K8, W7)
CC#1 – Hardware Inventory

• For more (hw specific) inventory info, use “get-wmiobject” / “gwmi”
• Basic HW:

```powershell
$computerhw = Get-WmiObject -Class Win32_ComputerSystem -ComputerName $Computer | select Name,Manufacturer, Model, SystemSKUNumber, TotalPhysicalMemory
```

• CPU:

```powershell
$computerCPU = gwmi win32_processor -ComputerName $Computer | select DeviceID,Name
```

• OS:

```powershell
$computerOS = gwmi Win32_OperatingSystem -ComputerName $Computer | select Version, Caption
```
CC#1 – Hardware Inventory

• Memory

$computerRAM = gwmi Win32_PhysicalMemory -ComputerName $Computer | select DeviceLocator,Manufacturer,PartNumber,Capacity,Speed

$computertotalRAM = (gwmi Win32_PhysicalMemory | Measure-Object -Property capacity -Sum).sum /1gb

• Disk:

$computerDisks = gwmi -Class Win32_LogicalDisk -Filter "DriveType=3" -ComputerName $Computer | select DeviceID,VolumeName,Size,FreeSpace
CC#1 – Hardware Inventory

• Serial Number:

```powershell
$computerBIOS = gwmi win32_bios -ComputerName $Computer
```

• GPU:

```powershell
$computerGPU = Get-WmiObject Win32_VideoController | Select description, driverversion, CurrentHorizontalResolution,CurrentVerticalResolution
```

• Putting this all together (demo):
CC#2 – Software Inventory

• Again, you can’t patch what you don’t know is out there
• You can’t hunt down and delete things if you don’t know what is there (Java, Silverlight, Flash, Adobe PDF reader)
• Yup, “KeyGen” / “Free” / “Jailbreak” often just means malware
• You can get a decent software inventory across AD just with Powershell
• You can also remotely delete offending software in Powershell
CC#2 – Software Inventory

• Example collection script for standard Windows installs
• Note the “are you there” check – this saves a bundle of time
• Often collect CC1 and CC2 info together
• Demo (sort by name and version for outlier versions or apps):

```powershell
$pcs = get-adcomputer -filter * -property Name,OperatingSystem,OperatingSystemVersion,LastLogonDate,IPV4Address
$patchinfo = @( ) ; $i=0
foreach ($pc in $pcs) {
    $i+=1; write-host $i
    if (Test-Connection -ComputerName $pc.DNSHostName -count 2 -Quiet) {
        $appsonpc = Get-WmiObject -Class Win32_Product -computername $pc.DNSHostName | select pscomputername, vendor, name, version, installdate
        $domainapps += $appsonpc
    }
}
$domainapps | export-csv -path ./domainapps.csv
```
CC#2 – Software Inventory (Actually CC#8)

• Special Case: Antivirus / Endpoint Protection app
• WMI Query against the MS SecurityCenter2 namespace
• Look for version mismatches, or user / helpdesk installed software
• Again, “sort” is your friend!

$wmiQuery = "SELECT * FROM AntiVirusProduct"

$AntivirusProduct = Get-WmiObject -Namespace "root\SecurityCenter2" -Query $wmiQuery @psboundparameters # -ErrorVariable myError -ErrorAction 'SilentlyContinue'
CC#3 – Vulnerability Management

• With hardware / software inventories done, time to patch
• OS / MS Office patching is automated by default
• Important because criminals have many patches reversed and incorporated into malware in days or even hours.
• Usually the approach is a bit more structured than “let it just happen”:
  • On patch Tuesday, monitor news sources for issues (ISC Stormcast, vendor newsgroups etc)
  • On Wednesday, patch a representative group of pilot servers and workstations
  • Patch the rest of the workstations on Thursday
  • Progressively patch all servers, with completion by Saturday at the latest
To get “last patch date” (ie – which stations got missed?):

```powershell
$pcs = get-adcomputer -filter * -property Name,OperatingSystem,OperatingSystemVersion,LastLogonDate,IPV4Address
$patchinfo = @() ; $i=0
foreach ($pc in $pcs) {
    $i+=1 ; write-host $i
    if (Test-Connection -ComputerName $pc.DNSHostName -count 2 -Quiet) {
        $tempval = new-object psobject
        $lasthf = get-hotfix -computername $pc.dnshostname | sort InstalledOn | select -last 1
        $tempval | add-member -membertype noteproperty -name Name -value $pc.dnshostname
        $tempval | add-member -membertype noteproperty -name PatchDate -value $lasthf.installedon
        $tempval | add-member -membertype noteproperty -name OperatingSystem -value $pc.OperatingSystem
        $tempval | add-member -membertype noteproperty -name OperatingSystemVersion -value $pc.OperatingSystemVersion
        $tempval | add-member -membertype noteproperty -name IpAddress -value $pc.IPV4Address
        $tempval | add-member -membertype noteproperty -name LastLogonDate -value $pc.LastLogonDate
        $patchinfo += $tempval
    }
}
$patchinfo | export-csv -path ./patchdate.csv
```
CC#3 / CC#11 – Infrastructure Patching / Vuln Mgt

• Lots of infrastructure is outside of the “Servers & Workstations” pool
• These often get missed in the patching policy:
  • Hypervisors
  • BladeCenters
  • Server Mgt such as rDRAC, iLo, BMC (web app vulns, default creds)
  • Backup Servers – often unpatched, frequently ransomware (OUCH)
  • Storage Switches, SANs, Routers, Switches, Wireless
• For instance, both if not patched Vmware vCenter and Cisco ISE are vulnerable to CVE 2017-5638 (think Equifax Breach)
• [https://isc.sans.edu/forums/diary/Struts+Vulnerability+CVE20175638+on+VMware+vCenter+the+Gift+that+Keeps+on+Giving/24606/](https://isc.sans.edu/forums/diary/Struts+Vulnerability+CVE20175638+on+VMware+vCenter+the+Gift+that+Keeps+on+Giving/24606/)
CC#3 – Vulnerability Management - Apps

• Software Patching is tougher
• MS Software can roll right into Windows Update of course
• Other vendors (Adobe, Oracle, Apple) also publish on a regular schedule, and often can be automated within the app.
• Software inventory will catch all the versions, but won’t highlight patch dates
• This is where actual inventory software can help
• Most orgs “have an app for that”
CC#3 – Vulnerability Management - Apps

• Your “patch all the things” app will miss stuff
• Tools like Nessus or OpenVAS can run authenticated scans
  • Can cross-check installed applications and configurations against known vulnerabilities
• Tools like Burp Suite or ZAP can be configured to assess web apps or web app component versions
  • For example, old versions of **jquery**, dijits, struts, Django, Wordpress, Wordpress Plugins should pop right up
  • Problems like this result in websites getting compromised at scale
• Got Admins?

$b = @()
$a = $()
'Domain Admins', 'Administrators', 'Enterprise Admins', 'Schema Admins', 'Server Operators', 'Backup Operators' | ForEach-Object {
    $groupName = $_
    $a = Get-ADGroupMember -Identity $_ -Recursive | Get-ADUser | Select-Object Name, samaccountname, DisplayName, @{n='GroupName';e={ $groupName } }
    $b += $a
}
$b | export-csv alladmins.csv
• Got Local Admins?

```powershell
function get-localadmin {
    param ($strcomputer)
    $admins = Get-LocalAdmin $strcomputer
    $admins = $admins |? {$_.groupcomponent -like '*"Administrators"*'}
    $admins |% {
        $_.partcomponent -match "\.+Domain\=\(.+)\,Name\=\(.+)" > $null
        $matches[1].trim('"') + "\" + $matches[2].trim('"')
    }
}

$i = 1
$targets = Get-ADComputer -Filter * -Property Name
foreach ($targethost in $targets) {
    write-host $i $targethost.name
    $vallist = @()
    $admins = get-localadmin $targethost.name
    foreach ($a in $admins) {
        $val = new-object pobject
        $val | add-member -membertype NoteProperty -name Hostname -value $targethost.name
        $val | add-member -membertype NoteProperty -name AdminID -value $a
        $vallist += $val
    }
    $vallist | export-csv -append adminusers.csv
```
CC4 – Admin Rights / CC 16-Account Monitoring

• Got Dead Accounts?
• Password Policy Exceptions?
• Legacy Group Members?
• Other “Problem Accounts”?

$probusers = get-aduser -filter * -properties * | select samaccountname, name, enabled, scriptpath, passwordlastset, passwordexpired, passwordneverexpires, passwordnotrequired, lockedout, lastlogon, lastlogondate, lastlogontimestamp, lockedout, cannotchangepassword, accountexpirationdate, mobilephone, officephone, telephoneNumber
$probusers | export-csv accountinfo.csv
CC4 – Admin Rights / CC 16 -Account Monitoring

• Microsoft Service Accounts:
  • LocalSystem
  • NT AUTHORITY\LocalService
  • NT AUTHORITY\NetworkService

• No passwords, so cannot be used by malware or attackers for lateral movement

• No interactive session access

• Service account password hashes can be extracted directly from the registry (using nishang for instance), or Kerberoasted (look for SPN)
CC4 – Admin Rights / CC 16-Account Monitoring

• Got Service Accounts?
  get-adcomputer -filter * | foreach { Get-WmiObject Win32_service -computer $_.name } | select-object systemname, displayname, startname, state | export-csv services.csv

• (“startname” is the account that starts the service)
• Filter out the “good” lines to find the problematic ones
• “long tail analysis” to find outliers (sort /count, look at 1’s and 2’s)
• Use the “offenders list” to contact vendors, fix configs, or find better products (or find malicious services)
Logging should detect common “lateral movement” scenarios:

- Single credentials connecting to multiple stations – esp Local Admin / Domain Admin
- Be sure to account for admin scripts and helpdesk
- Service Accounts (if they exist) shouldn’t be logging in interactively
- A good job for your SIEM

Microsoft LAPS makes a great defense here!

- Sets Unique Local Admin passwords, change periodically
- Careful though, security tools like LAPS can also be attacked
- Also be sure not to LAPS your DCs!
CC 9 – Limit and Control Network Ports

• Should define workstation Firewall rules in GPO
  • Only “trusted hosts” need tcp/445 or tcp/3389 to workstations
• Should do the same for Servers
  • Ditto, nobody needs to map a drive to your SQL Server
• Powershell Audit of Firewall State:

Get-NetFirewallProfile -PolicyStore activestore | select name, enabled, defaultinboundaction, DefaultOutboundAction

• Firewall Rules:

get-NetFirewallRule -policystore activestore
CC 9 – Limit and Control Network Ports

• Should also expand access controls to infrastructure
• Access Classes on network gear are easily done but seldom seen
• ACLs to prevent access to vCenter, BMC, iLo, SAN Admin etc
• Your receptionist does not need access to vCenter, but for some reason they all seem to have it
• Neither does your Guest Wireless SSID
Next Steps?

• Many of the other Critical Controls can be implemented using Group Policies (also $free)
• Others such as implementing 802.1x require other MS components, but are still free
• **Look for a “Part 2” to this presentation** outlining these
• Budget dollars are often attached to:
  • Many (but not all) of the Foundational Controls - Browser and Email protections, Malware defenses etc
  • Almost all of the Organizational Controls - with the exception of “getting started” phases of User Education
Blogs and Software Repo

• Repo: https://github.com/robvandenbrink/Critical-Controls
  • All scripts and supporting blog links

• Watch for new content at https://isc.sans.edu
Questions?

- Repo: https://github.com/robvandenbrink/Critical-Controls
- Blog: https://isc.sans.edu

- https://www.coherentsecurity.com