Application White-listing with Bit9 Parity

Mike Weeks

Application White-listing with Bit9 Parity

GIAC (GSEC) Gold Certification

Author: Michael Weeks, mweeks9989@gmail.com
Advisor: Dominicus Adriyanto

Accepted: October 5, 2014

Abstract

Endpoint protection based solely on Anti-Virus (AV) signature-based technology is fundamentally flawed. AV technology will only protect against known malware the AV provider has identified and written signatures for. The problem with classic AV protection is it does not detect unknown or unseen malware because it is signature based. A better method is to identify secure software and only allow that software to run. This technique is called application white-listing. Bit9's security solution, Parity, is a leader in this arena. This paper will delve deep in the implementation, configuration, and deployment of the product and explore how the data provided by the product can be utilized to significantly enrich the metrics of a security operations team.

1. Introduction

Antivirus is a requirement for a host of compliance standards and is championed to be a critical component for any security baseline (PCI-DSS 3.0-5.1). A recent google search for "Cyber Security Breaches" in Google News shows 16,700 results in Google News. Even NIST has stated that that AV is not an adequate control (Artes, et al, 2013). The basis for this argument is that AV, even with heuristics, looks for methods or signatures that are known to the specific AV vendor. Bit9 Parity goes a step further and restricts the execution of any executables or applications to those only allowed by the product (Bit9 Datasheet, 2013). Parity has a host of benefits as well as some significant drawbacks, but with proper and careful implementation, a deployment of Parity can be successful.

Parity has multiple methods to manage and control an environment. Parity is deployed with a server, database and console to control and manage Parity Agents. The deployed agents are a package of executables and configuration files that contain a kernel module that sits on the hardware layer and proxies the raw system calls from the user layer to those resources. For this reason it makes manipulation of the agent from the user layer very difficult. There is also a management console to manipulate the server that controls all agents on endpoints. The server has a backend database that stores data to be used by its processes and to administer the environment. The data in the back-end database has tremendous use, not just for managing the product, but also as a source for a security operations team to pull information. Carbon Black, which was also recently purchased by Bit9, functions as a detection mechanism to support Parity's control mechanisms. According to the Bit9 site: "Records of execution, file system modifications, registry modifications, network connections, and a copy of every unique binary" (Bit9 and Carbon Black are Now One Company 2014).

Some weaknesses in the product are its dependency on the software-signing certificates for ease of administration and user acceptance. This was a significant issue when Parity was compromised by advanced actors, and the Bit9 software signing certificates were compromised (Krebs, 2013). This can be corrected by disabling the trust of Bit9's software-signing certificates. In addition, the execution prevention can be bypassed by exploits and attacks that stay transient in memory as well as injecting into running processes. One specific method for bypassing the product, outlined at Shmoocon in 2012, was to inject into a .DLL and then migrate to the notifier.exe tool, in the Parity agent, in order elevate privileges. The vulnerability was patched in version 6.0.0. (Shaffer 2012).

Despite these few weaknesses, Bit9 Parity can be deployed to greatly enhance endpoint protection as well as protect against most unknown threats. However, in order to protect against unknown threats, and prevent significant negative impact on operations it must be deployed correctly.

Michael Weeks, mweeks9989@gmail.com

1.1. Pre-Deployment

During pre-deployment, the first thing that must be decided is where it will be deployed. Bit9 would recommend that the product be deployed on all systems in an environment. However, this is not feasible as the cost of the product and the complexity of most environments makes 100% immediate deployment difficult. Parity takes a default deny approach (Bit9 Data Sheet, 2014). This is a good method for protection but can make deployments difficult. To deal with this situation it is a good idea to deploy the product in homogenous environments first.

Therefore, in planning deployment it is best to identify and group environments by their similarity and their levels of criticality. The most critical could be where the protection needs to go first. However an additional risk of deploying the product in critical environments is that by description they are critical to the business. So the product must deployed with care, proper planning and testing.

There are two major methods of deployment. The product must first be deployed with one of two goals in mind, protection, or control of the environment.

1.1.1. To Protect the Environment (Client-side)

Protection and prevention is absolutely ideal when it comes to deployment of Parity. When working with dynamic and non-homogenous environments the product should be deployed in this mindset. An excellent environment for deploying to protect would be a desktop or laptop (client side) environment.

With this method of deployment a few extra deployment options must be configured. For instance a few extra white-listing rules to allow for dynamic approvals of a multitude of applications, specifically .net and other development tools. Also enabling trust approvals to a certain level will ensure any software signed by a certificate trusted by Bit9 Parity's Database. This will remove administrative overhead significantly by allowing those files that are "known good: to run in environment. One caveat to that being, that a lot of DLLs and other libraries are not signed by a software certificate and will have to be identified by administrators of the product.

Some organizations have purchased Parity purely for the external device control. Parity has excellent control for the denial of non-approved USB keys and can prevent this type of threat. Installation and enabling of this feature prior to any lockdown of executables can be a quick milestone accomplishment in Parity deployment plan.

Another consideration is that Parity comes with two types of licenses: Visibility which will provide file and event tracking, provide file bans and device blocking, as well as the Parity Suite which is designed for getting systems into full lockdown.

Figure 1 – Trust by Certificates

1.1.2. To Control the Environment (Servers)

In order to protect an environment administrators and security personnel must control and understand their environment. However methods of deployment can differ with these underlying goals in mind. Deploying to control should be applied in specific environments that have rigorous change control and a low level of change. This would be server environments or other systems that are running on end-of-life operating systems, such as Supervisory Control and Data Acquisition (SCADA) systems, as well as some Point of Sale Systems (POS).

This method of deployment must become an integral part of the change control process, because no changes can be made without the Parity administrator getting involved in the change. Limiting software approvals by certificate trust level must be used sparingly. Any approvals based on trusted directories, users and other trust levels baked into the product should not be allowed. Rules should specify a particular process, specific directory not ending in an *, as well as a not using the "everyone" user.

This method of deployment (as well as the protection deployment) also heavily utilizes policies. Levels of Lockdown should be created to move systems in and out of maximum lockdown. In this method of deployment there is heavy reliance on policy, because the servers will be migrated in and out of the policy in conjunction with the change control policy.

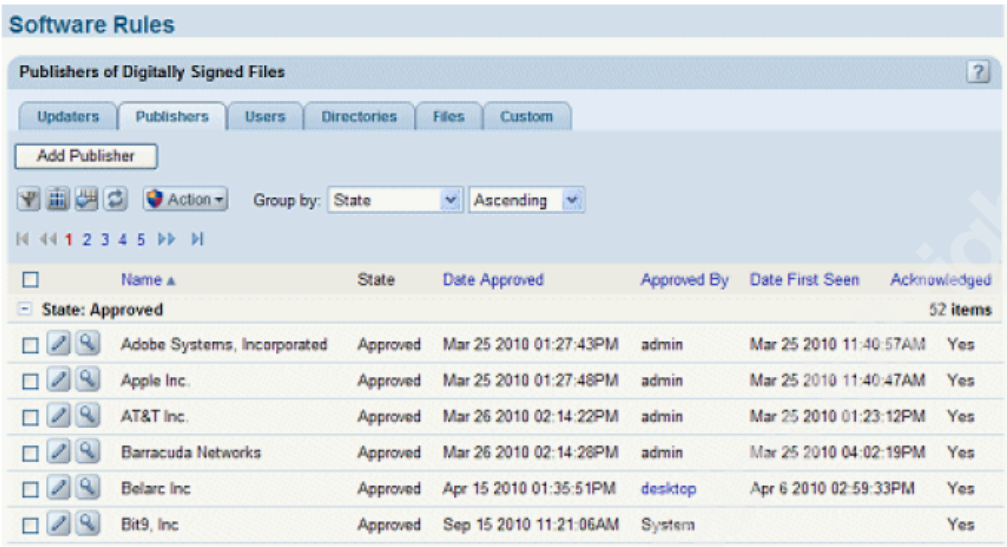


Figure 2 – Trusted Installers

It is highly recommended to deploy separate servers for each of these methods of deployment. To ensure that over-permissive software rules do not accidentally give excessive permissions to those servers in the control deployment.

2. Deployment

After deciding what environment to start, it is time to build out the Parity Server and console. According to the Bit9 installation guide, the server should have a SQL server available or a new SQL server database, either 2005 or 2008 deployed and configured prior to installation. (Parity 6.0 Deployment Guide, 2013) The server will also need .net framework 3.5 and a host of other web application Microsoft requirements. All should be included with a current version of Server 2008. [Appendix A]. Prior to installation ensure that all servers meet local hardening procedures.

It is recommended to use a database on another server for administration of more than 300 systems. After the database has been identified (or installed) the service account that is identified will need create and owner rights for the database. The database will be called DAS – this is prepopulated by the Parity install scripts, as well as the schema, and all tables. Logging in as the service account and running the installation will greatly simplify deployment.

Steps for installation are detailed in the Parity Deployment guide from Bit9, the installation is simple and can be walked through by most experienced administrators. Ensure to install with best practices and ensure that all drive installations are on a separate drive from the operating system. Logs should also be on a separate drive, especially if the database is on the same server. The full document should be retrieved from Bit9's support portal for installation guides for the version in use.

2.1. Configuration

After the server has been installed, it should be simple to browse to the https://localhost which will direct to the Parity console if logging on locally. Browsing from another system to https://servername which will direct the administrator to the Parity console. The default credentials should be username admin and password admin. As always, best practices, change immediately!

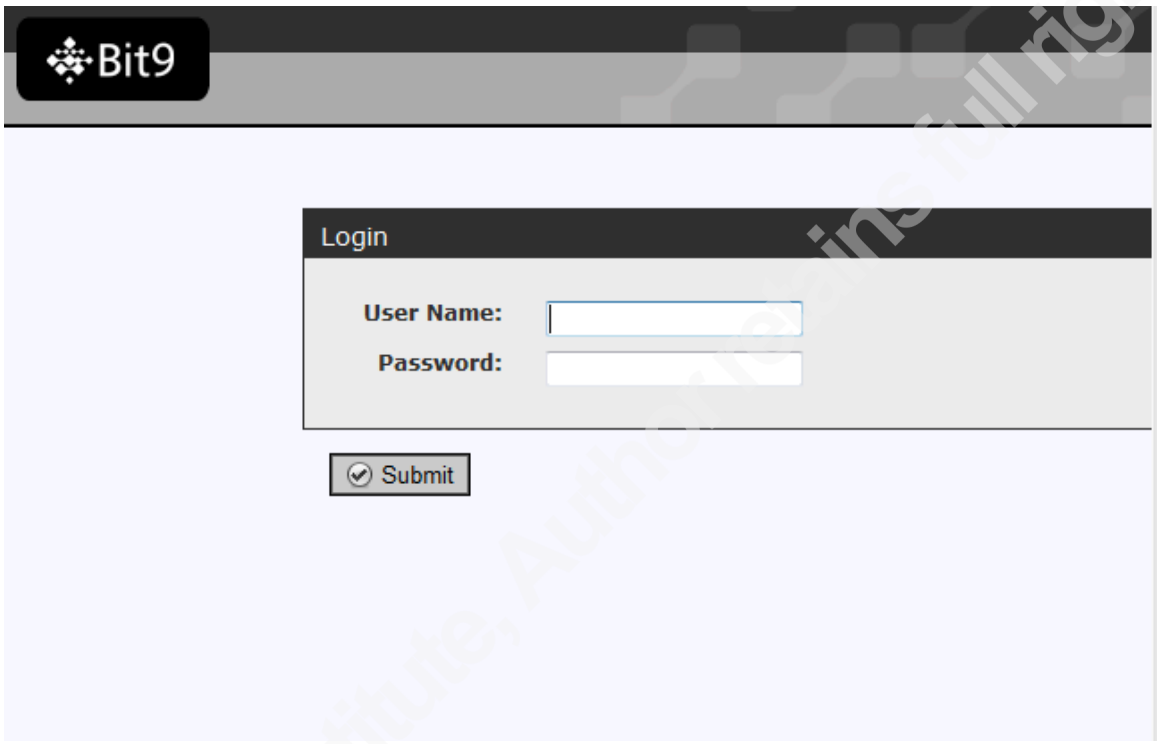
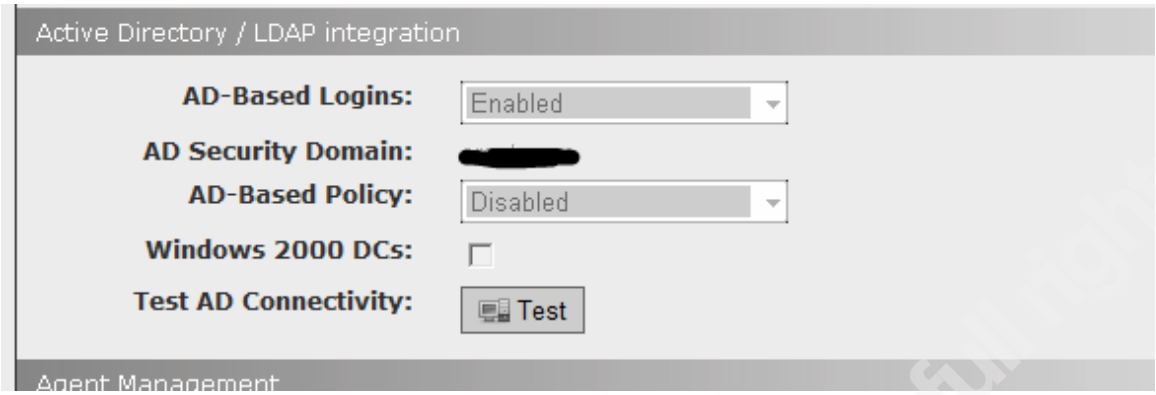


Figure 4 – Login Screen

After logging in and changing default username and password, an excellent next step is to enable AD logins for administrators. Using local change control procedures request three security groups with exactly the following naming convention: cn="Bit9 Administrators" – full administrators to the console, cn="Bit9 Power Users" – Power Users, and cn="Bit9 ReadOnly Users" – users that can access the console and query data. All other users will have unauthorized access. The security groups will have to be on the same domain that the Parity server is joined to. Administrators should be individuals familiar with the product that, know how to maintain and own the product and its deployment. PowerUsers should be those member of the helpdesk/desk side in a Deployed to Protect (client side) environment or System Administrators in a Deployed to Control (server) environment. ReadOnly Users should be those in management or possibly some security analyst that need to research executables in the environment.

After the security groups have been configured, it is a simple process of enabling the AD authentication under the Administration Tab > General Tab under Active Directory / LDAP integration. This configuration should be carefully considered in the Deploy to Control method for deployment, with this type of configuration the security of Active Directory must be considered absolute.



2.1.1. Bit9 Knowledge Base

Another critical component is the Bit9 knowledgebase. The Bit9 knowledgebase is one of the single largest collection of known good executables available commercially. This will require outbound connectivity to the Bit9 knowledgebase servers on port 443 from the Parity server. It will also require a license from Bit9 knowledgebase. There is an open API to query the data through a restful API. (Script attached – Appendix B) The knowledgebase can be configured in the Administration tab > Licensing > Parity Knowledge Activation.

2.1.2. Other System Administration

On the system administration tab there are a host of other setup actions that can be accomplished on this tab as well. On the mail tab, the SMTP settings for alerts can be configured to send alerts for status of systems. The advanced options has the ability to back-up the database, configure automated updates, log out times for the parity console, file uploads configuration, old computer cleanup, software rule completion, and certificate options. Most of these options are not of much concern, however the cleaning up of old agents should be configured.

The Security tab under Administration has a critical configuration. The Parity agents communicate with the Parity Console through certificate authentication. The date the certificate expires should be noted, when this happens the agents will not be able to communicate with the server. In Deploy to Protect configuration, this can be production impacting because if agents cannot communicate with the Parity Console then the memory will spike on the servers causing resource depletion. Which is not something that will make operations personnel very happy. A script is attached to move agents from lockdown policy in case a situation like this occurs. [Appendix B]

The events tab have two important configurations. Event Log Management need to be well thought out and minimized to ensure system performance is not significantly impacted. The External Event Logging can send events to a SIEM or other syslog server for off-box storage and retention of logs. This can be set to send in CEF, syslog, and other formats.

2.1.3. Policy Configuration

Designing the policies in Parity is absolutely critical to having a successful deployment. The default policies that come with the product are a good place to start. “Default Policy” which is designed for the agents to go to once the agent is initially installed. The “Local Approval Policy” which is designed to approve any running executables on the system. The “Template Policy” which is designed to be copied and configured for new policies.

Initially four new policies need to be created for management of agents. “Lockdown Policy” must be created to replace the Default Policy and to be the final stop for agents during configuration. “Lockdown Reporting” policy which will be configured on systems to report as if they were in lockdown without actually blocking, and a “Monitoring Policy” to start hashing and collecting execution information on systems. “Disabled Policy” should also be created to for the installation of the agents, and removal of the agents if necessary.

The cycle of agents to move for installations should be as follows:

- “Disabled Policy” for installation, configuration will allow systems to be installed and uninstalled easily without issue. Also with initial installation of disabled agents on systems, agents can be deployed in mass to a list of systems.
- “Monitor Policy” agents will start to hash all files on the system. This activity is processor intensive and should be implemented on systems during downtime, or on one system at a time.
- “Lockdown Reporting” agents will behave and report as if the agent is in lockdown mode. This should be utilized for monitoring of agents to remove false positives during tuning.
- “Lockdown” agents will lockdown and not allow any approved executables to run.

Michael Weeks, mweeks9989@gmail.com

Figure 5,6,7 – Policy Configurations

The next critical point to configuration and deployment are software rules. Software rules are a unique feature in Parity and can be the key to a successful deployment or can possibly, undo any protections Parity may provide. After configuring the policies and any trusts for software certificates, software rules should be configured. If the environment is a windows environment some of the default rules should be turned on. Updates for Windows should be enabled, and depending on the environment (Server/Client) all other available software updates that are available on the systems. Most software can be approved through certificates by watching for executions, researching the execution and approving. Some

software creates dynamic executables or DLLs, or .pyc python executables, an example being is the .Net framework.

The .Net framework creates dynamic DLLs inside multiple directories. There are multiple paths for file creation with specific care to ensure that only DLLs can be written and the specific process is csc.exe.

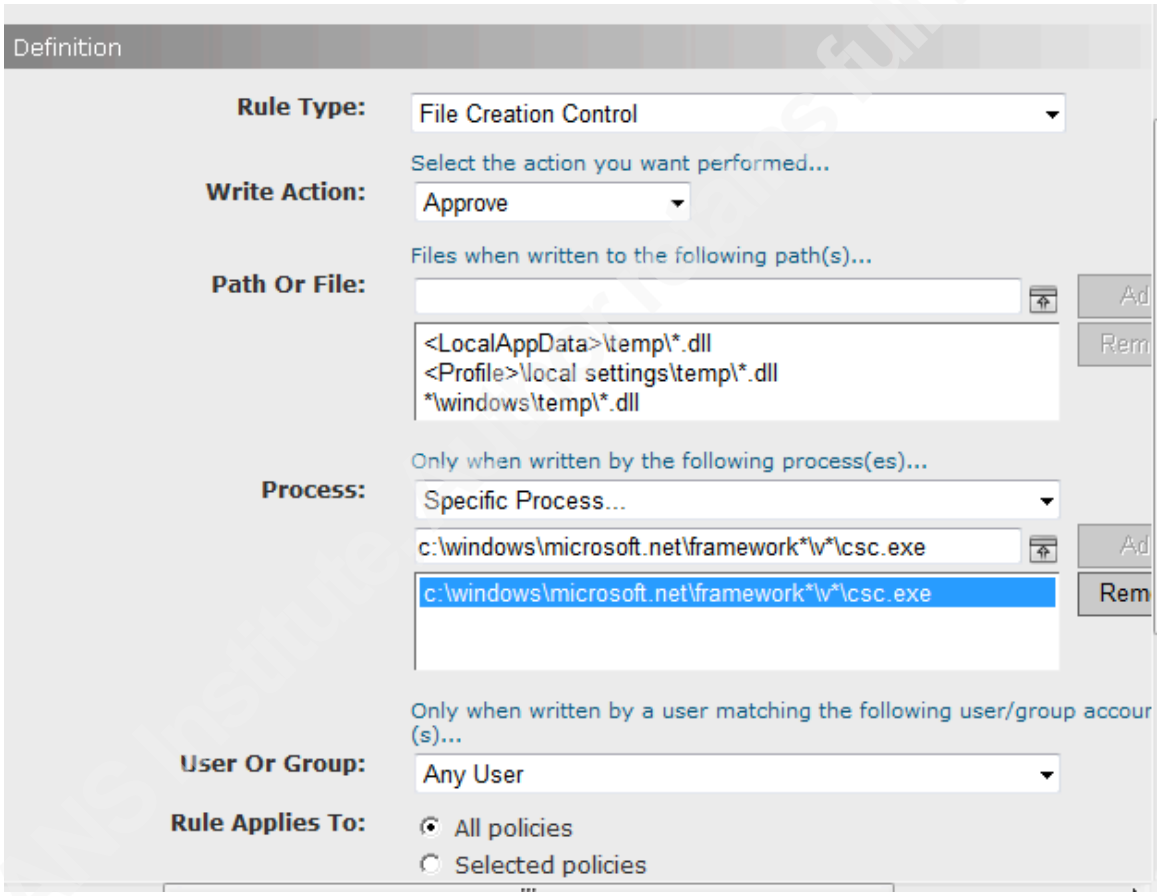


Figure 5 - .Net framework file creation

This will allow the .DLLs to be written as needed to the specified directories as well as only by the specified process. This protects the creation of the files but does not automatically approve the files created on disk. This is only one half of the equation, it is necessary to add the execution control of DLLs on the .net framework. Ensuring that only <Reg:HKLM\Software\Microsoft\.NetFramework\InstallRoot>*\mscorsvw.exe and <Reg:HKLM-SoftwareX86\Microsoft\.NetFramework\InstallRoot>*\mscorsvw.exe are allowed to run DLLs will go a long way towards protecting the environment from methods of exploiting the .net framework.

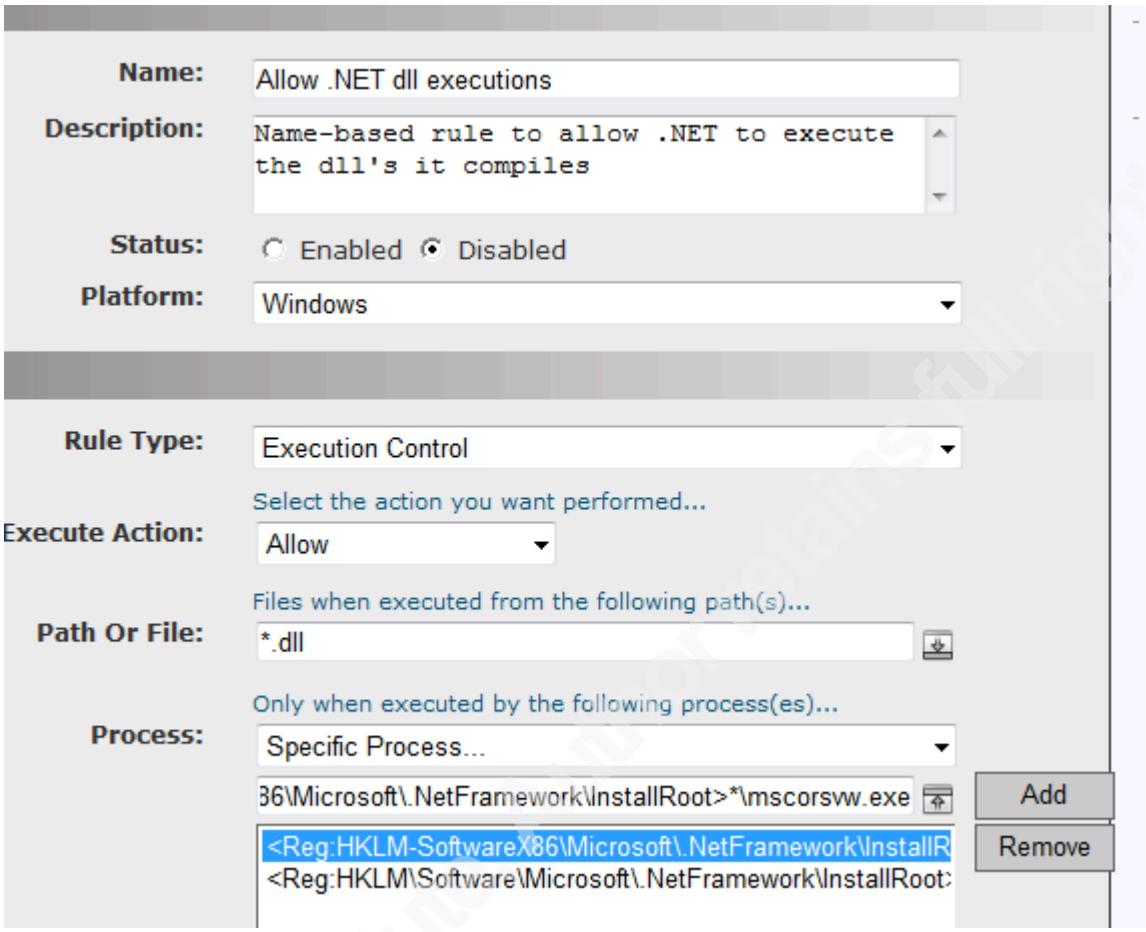


Figure 6 - .Net Framework Execution Rule

This is one of the more unique rules that required to be created in the majority of Windows environments. For most rules, a good rule of thumb is to monitor systems in lockdown reporting and review the blocks as block events come in. If the software was a legitimate piece of software, copy the path, and the process that executed the software. Then add a rule to the software rules that allows the software to execute with the specified process. In higher security environments (server) it may be ideal to enter a username as well.

Scripts and software become difficult because it becomes necessary to ensure those scripts can execute as well. There are pre-built rules for the execution of scripts however they can be somewhat permissive. It is slightly better to designate specific directories and not allow those scripts to start other executable processes, or even better utilize script signing and sign scripts that are allowed to run in an environment.

2.2. Deploying Agents

After all the agent configuration policies have been created and some basic software rules like the .net software rule, it is time to start deploying agents. The agents can be downloaded from https://parityserver/hostpkg/. It is best to start with an agent disabled policy.

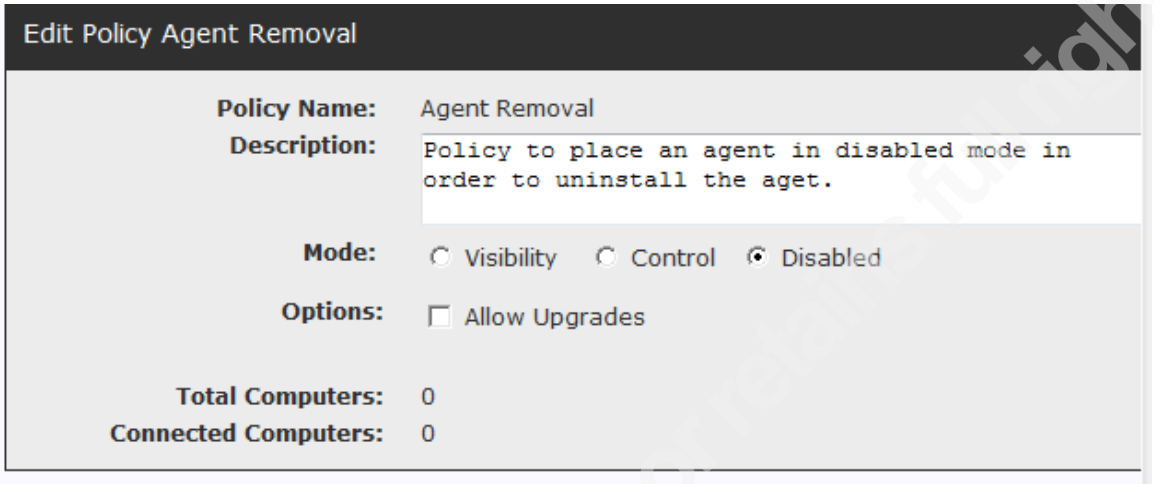


Figure 7 – Agent Removal Policy

Installing the agent can be done on all systems through multiple methods, GPO, software packaging and through scripting. Scripting is beneficial, because it can be scheduled and the output can be collected for error checking. See appendix B for an example installation script.

Installing the agents is a slow process which requires getting a list of all devices, verifying in the Parity Console the assets are available and the communication level of the agent. Something to consider is that any Windows version after Server 2008 and Windows 7 should deploy the agents without the need for a reboot. However older versions will require a reboot. If the agents are not communicating with the Parity Server ensure that agents can reach the server on TCP port 41002 or reboot the system if necessary.

2.3. Locking Down the Agents

After ensuring that all agents are deployed it is time to start locking down agents. This can be accomplished by selectively moving agents into the "Monitoring Policy". This step in the installation process has the most impact on the system therefore it is best to move agents into this policy during times of less usage and only move a few agents at a time. The Parity agent will start to hash every file on the system at this time and will utilize CPU and memory. If deploying to client systems, initiate after hours. If it is a server environment attempt to stagger it between failover between high-availability environments, or during maintenance windows.

After all agents have been moved to the "Monitor Policy" utilize the same method to move the agents into "Lockdown Reporting". As systems start reporting blocked executables slowly approve individual executables as well as create software rules to ensure that valid executables are approved. This

process can be slow, however it is crucial to ensure that the false block rate is a low as possible in order reduce the impact to business operations. Once the block rate has been reduced to an acceptable level it is time for the most difficult portion of the deployment process.

2.3.1. Policies and Procedures

Before moving any systems into lockdown (other than testing systems) it is time to ensure there is a process for addressing blocked executables that users/administrators need to run on the systems. It is likely that any organization that is going to deploy Parity will have methods and processes for IT workflow. This is an ideal method for dealing with end user issues with Parity blocks of potentially useful and needed executables. This should be communicated with the user population to ensure that users know where to go in case they have Parity block.

User notifications can be configured in the Parity Console to inform the user where to go if they do receive a block, as well as very basic workflows in the product. The workflows in the product are not entirely complete and will require some type of process to track changes and to ensure that software is approved for use. Initial deployment will be painful, but with proper project management and communication skills this process can go well. See appendix C for example Policy/Procedures documents.

For server deployments, to ensure that false blocks are mitigated, use the change control system. In the change control process there should be a method for changing systems. The Parity administrator and team will have to get involved with the change control process. An excellent method is to move the agents into the "Local Approval" policy during maintenance, and then back into lockdown when the maintenance is completed. For emergency changes the Parity Administrators will have to have on-call rotations, or if resources are limited ensure that System Administrators are trained to deal with situations where Parity could prevent the execution of duties.

Using these methods and working slowly and methodically, Parity can be locked down efficiently and with minimal incidents.

3. Operational Uses for Parity

There are many other uses for Parity other than just to protect the environment. It is an excellent source of information showing exactly what is running in an environment. By querying the data in Parity, a Security Analyst could research to find if a downloaded malicious file actually reached the endpoint system or not. An Analyst could also upload a hash from doing analysis on another system to Parity to block across the install base. The server actually has a very simple SOAP API utilizing JSON that can be called very simply from web posts. (Example Script Attached Appendix A)

Another consideration is the recent acquisition of Carbon Black by Bit9. The purchase of Carbon Black also has tremendous potential for Security Analytics utilizing the visibility that Parity has into an environment. Its ability to delve into the memory could tremendously offset some of the places that Parity

fall-short in prevention. Some organizations may want to purchase this additional tool to augment their security analysis.

As mentioned earlier, another very significant protection that Parity provides is the blocking of USB devices. This happens at the kernel level, so a huge sector of USB attacks can be mitigated using Parity.

4. Conclusion

When evaluating any technology technologist and security practitioners should carefully analyze with due care the technologies, especially those that will require employee time and energy as well as significant capital expenditure. Bit9's Parity will take significant time, funds, and energy to deploy. It will take a concerted effort from senior leadership to decide on the product and then organizational push to deploy it.

The approach that Application-Whitelisting takes is a simple one, trust only what is known and all other executables and binaries are not trusted and are not allowed to run. If an organization believes that they may be targeted by an advanced actor then the advanced protection provided by an approach like Application-Whitelisting should be evaluated.

The decision is a risk decision, the protections Parity offers are significant. If deployed properly, malware will not be able to gain a persistence on a network, as well a huge number of other attacks will be mitigated. If an organization deems that they need the level of security, the costs and energy that Parity takes to deploy are well worth the efforts.

Michael Weeks, mweeks9989@gmail.com

5. References

Artes, Francisco, Baylor, Ken PhD Phatak, Vikram (2013), NSS Labs – NIST: Cyber Security Framework RFI, Retrieved from: http://csrc.nist.gov/cyberframework/rfi_comments/040813_nss_labs.pdf, on 08/31/2014

Beachey, Jim, (2010). "Application Whitelisting Panacea or Propaganda?", SANS Reading Room, Retrieved from: http://www.sans.org/reading-room/whitepapers/application/application-whitelisting-panacea-propaganda-33599 on 08/21/2014

Krebs, Brian, (2013), "Security Firm Bit9 Hacked, Used to Spread Malware", KrebsonSecurity, Retrieved from: http://krebsonsecurity.com/2013/02/security-firm-bit9-hacked-used-to-spread-malware/ on 08/21/2014

Krebs, Brian, (2013), "Bit9 Breach Began in July 2012", KrebsonSecurity, Retrieved from: http://krebsonsecurity.com/tag/bit9-breach/ on 08/21/2014

Shaffer, Curt, (2012), "Raising the White Flag-Bypassing Application Whitelisting", Foreground Security, Retrieved from: http://foregroundsecurity.com/resources/blog/164-raising-the-white-flag-bypassing-application-white-listing, on 09/02/2014

Unknown (2014), "DATA SHEET | The Bit9 Security Platformv7", Bit9, Retrieved from: https://www.bit9.com/download/data-sheets/Bit9-Security-Platformv7.pdf on 08/21/2014

Unknown (2013), Payment Card Industry (PCI) Data Security Standard – Requirements and Security Assessment Procedures, Retrieved from: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf on 08/31/2014

Bit9 Inc., *Installing Parity,* Document Version: 6.0.2c, September 8, 2011

Unknown (2014), Bit9 and Carbon Black are Now One Company, Retrieved from: https://www.bit9.com/solutions/bit9-and-carbon-black-one-company/, on 08/31/2014

6. Appendix A - From "Installing Parity" V. 6.0.2

Physical Single-Tier System < 5000 Clients			Physical 2-Tier System > 5000 Clients	
Requirement	**Combined Parity Server/ SQL Server Specifications**		**Parity Server Specifications**	**SQL Server Specifications**
	< 300 Clients	**300 - 5000 Clients**		
Processor	Dual Core Server Class	Dual Core or Dual Processor Server Class	Dual Core Server Class (<10000 Clients) Quad Core Server Class (>10000 Clients)	Dual or Quad Core Server Class
Disk space	40 GB	2 drives: 40GB for Parity, 72GB for SQL	40 GB	72 GB+
RAM	2-4 GB	4 GB	4 GB+	4 GB+
Network	1 GB NIC	1 GB NIC	1 GB NIC	1 GB NIC
IP address	Static IP address only (no DHCP) with an assigned FQDN or alias. Computers running the Parity Agent recognize the server by either its fixed IP address, DNS-name lookup or an alias. IPv4 must be the default protocol for the server.			

If you are using IIS 7.0, in the **IIS Roles Manager**, verify the following configuration:

- Common HTTP Features: All
- Application development:
  - ASP.NET
  - .NET Extensibility
  - CGI - ISAPI Extensions
  - ISAPI Filters
- Health & Diagnostics:
  - HTTP Logging
  - Logging Tools
  - Request Monitor
  - Tracing
- Security:
  - Basic Authentication
  - Windows Authentication
  - URL Authorization
  - Request Filtering
  - IP and Domain Restrictions
- Performance: None
- Management Tools:
  - IIS Management Console
  - IIS Management Scripts and Tools

- Management Service
• FTP Publishing Service: None

(This is not a full excerpt for server requirements please follow Bit9 – Parity Installation Guide from vendor for all information)

7. Appendix B

```
#!/usr/bin/powershell
# Pass API calls to console
# post hash to parity and blockify it
function Upload-HashToParity($hash,$ParityURL,$feed){
    #Requries pshell 3.0
    $ParityLoginUrl = "$ParityURL/login.php"
    $cred = Get-Credential
    $username = $cred.GetNetworkCredential().UserName
    $Credentials = New-Object System.Management.Automation.PSCredential -
ArgumentList $UserName, $adPW
    $password = $Credentials.GetNetworkCredential().Password
    $useragent = "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Win64; x64;
Trident/4.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729;
.NET4.0C; .NET4.0E; InfoPath.3)"
    #$headers = @{}

    Invoke-WebRequest -Uri $ParityLoginUrl -UserAgent $useragent -Method Get -
SessionVariable Parity

    $loginPostData =
@{username=$username;password=$password;oldURL='/home.php';oldURLParams='';dbuse=
'das';submit='Submit'}

    Invoke-RestMethod -Uri $ParityLoginUrl -UserAgent $useragent -WebSession
$parity -Method POST -Body $loginPostData

    #let's see if we have a cookie set
    if ($Parity.Cookies.Count -eq 0) {
        Write-Host "fail to connect"
        break
    }
    else
    {
        $rule = "$feed-$dte"
        $description = "Malware+Artifacts"
        $fileUrl = "$ParityURL/file-policy-details.php"
        $uploaddata = @{
            fileRuleName=$rule
            ruleActionSelect='3'
            nameOrHash='hash'
            Platform='1'
            fileName=''
            hashType='MD5'
            hash=$hash
            antibody_id_list=''
            antibody_id=''
            description="Malware+Artifacts"
            policyScope='allHostGroups'
            formAction='create'
            returnUrl='/approvals.php?tab=fileRules'
            fromFileDetails='false'
        }
        $upload = Invoke-WebRequest -Uri $fileUrl `
            -ContentType application/x-www-form-urlencoded `
            -UserAgent $useragent `
            -Method post `
            -WebSession $Parity `
            -body $uploaddata
    }
}
```

```
#############agent Removal Script##############
Param ([string]$password)

$computer = get-content .\listofcomputers.txt

invoke-command -ComputerName $computer -ScriptBlock {
    c:\Program Files (x86)\Bit9\Parity Agent\dascli.exe password $password
    c:\Program Files

 (x86)\Bit9\Parity Agent\dascli.exe enable
    c:\Program Files (x86)\Bit9\Parity Agent\dascli.exe setpolicy 'agent
removal'
dards
}
##################end Script################################


############################agent installation Script#########################
$servers = Get-Content .\servers.txt

foreach ($server in $servers)
{
    invoke-command -ComputerName $server -ScriptBlock {
        cmd.exe "\\uncpathtoparitymsi"
        }
}

#########################end Script#########################################
```

Appendix C

Sample Policy and Procedures:

Desk-Side Parity Procedures

1. Purpose

Anti-virus software has been significant tool in combating malicious software and unwanted software. However the organization has decided that the protections AV provide are insufficient and have taken an application whitelisting approach. Application whitelisting is an approach to only allowed approved software to be allowed to run in the environment.

2. Scope

The organization has chosen Bit9's product Parity for the implementation on end-point systems. The agent will be installed on all end-point systems to protect the organization against unwanted software. The end goal is to have all endpoint systems in a lockdown configuration.

3. Policy

All systems that users log into directly to work that are client side system and managed by the desk-side support team will have the agent installed. The agent will be installed in the following fashion:

- When a new system is built out, and on all current systems the agent will be installed. This will be accomplished within 5 days of system build.
- After the agent is installed the agent will be moved into monitor mode to hash all endpoints on the system. This will be accomplished within 5 business days after installation.
- Once the hashing of all files are complete the agent will be moved into lockdown reporting. This will be accomplished within 5 business days after moved into monitor mode.
- Configuration will be handled by the Parity Administrator team to ensure that false positives are mitigated to a <5% level. This will be accomplished within 20 business days of the installed system.

4. Enforcement

Any systems that do not meet the compliance date will be restricted from network communication until lockdown is complete.

Procedure for Software Approvals

The software approval list is at the following location: http://internal-corporate-site/approvallist.organization.com. All software on this list will be configured and allowed to run. If there are any additional items requested from the user population the following procedures will be met:

- User will submit a helpdesk ticket requesting the additional software.
- The user's supervisor will review the software for an approved business use.
- Ticket will then be routed to the Desk-Side Support Team to ensure the software can be supported.
- The ticket will be routed to the Security Operations team will review the software for malicious sources.
- Finally the Parity Administration team will review the ticket and either whitelist the installation file and/or create any software rules to allow the software to run.

Deploy to Protect (Server) Example Policy and Procedures:

Change Control Process Utilizing Parity

1. Purpose

The change control monitoring for the organization is a rigorous and important process to protect the environment from dangerous changes as well as security vulnerabilities. For this reason has decided to use an integrity control application from the Bit9 called Parity.

2. Scope

The software will be deployed on all production servers providing services to customers.

3. Policy

The Parity agent will be deployed to any production servers in a disabled policy. Any new servers will receive the agent according to the in-house build process. The agent will then be moved to the Monitor policy in order to hash files on the system. This change will be scheduled and change controlled. After all files have been hashed the agent will then be moved into Local Approval, to approve any files on the system. After all files on the system have been approved, the agents will be moved into Lockdown Reporting for the Parity Administration team to create any software rules that are necessary for the server function. Finally the server will be moved into Lockdown to prevent any unauthorized changes.

After the completion of the Change Control Process – see http://internal-corporate-site.organization.com/ChangeControlProcess.doc. The Parity Administrator will move the servers into local approval to approve any changes that will occur during the change window. After the change window is complete the Parity Administrator will move the server into Lockdown Reporting to monitor any potential blocks of valid executables that may have changed after the installation. After a sufficient amount has passed but no more than 24 hours, the Parity Administrator will move the system back into lockdown.

For any emergency or exception changes the Parity Administrator will be available on-call to change the server into the appropriate policy to perform the maintenance necessary. The emergency or exception change will be reviewed according to the change control policy.

4. Enforcement

Any employee or administrator that does not follow the rules in this policy may be subject to disciplinary action, up to and including termination of employment.

Michael Weeks, mweeks9989@gmail.com