Be Like Eric

Secure Like a Programmer Codes

Jimmy Ray Purser

Twitter: @llumio, @jimmyray_purser
Which one?
The Art of Unix Programming

Eric S. Raymond
Rule of Modularity: Write simple parts connected by clean interfaces.

Rule of Clarity: Clarity is better than cleverness.

Rule of Composition: Design programs to be connected to other programs.

Rule of Separation: Separate policy from mechanism; separate interfaces from engines.

Rule of Simplicity: Design for simplicity; add complexity only where you must.

Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.

Rule of Transparency: Design for visibility to make inspection and debugging easier.

Rule of Robustness: Robustness is the child of transparency and simplicity.

Rule of Representation: Fold knowledge into data so program logic can be stupid and robust.

Rule of Least Surprise: In interface design, always do the least surprising thing.

Rule of Silence: When a program has nothing surprising to say, it should say nothing.

Rule of Repair: When you must fail, fail noisily and as soon as possible.

Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.

Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.

Rule of Optimization: Prototype before polishing. Get it working before you optimize it.

Rule of Diversity: Distrust all claims for “one true way”.

Rule of Extensibility: Design for the future, because it will be here sooner than you think.
Data Centers are becoming more SD….

It is because it’s just easier?

Or better?

Agility

Or the latest Buzzword Bingo
Rule of Modularity

*Write Simple Parts Connected by Clean Interfaces*
Rule of Clarity

Clarity is Better Than Cleverness

VLAN 22
VXLAN VNID 1020
SGT:3
ACL 113
Rule of Clarity

Clarity is Better Than Cleverness

VLAN 22

VXLAN VNID 1020

SGT:3

ACL 113
Rule of Clarity

Clarity is Better Than Cleverness

<table>
<thead>
<tr>
<th>APPLICATION</th>
<th>ROLE</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERP</td>
<td>Web</td>
<td>US-West</td>
</tr>
<tr>
<td>Ordering</td>
<td>Processing</td>
<td>Azure</td>
</tr>
<tr>
<td>HRM</td>
<td>Database</td>
<td>EU-London</td>
</tr>
</tbody>
</table>
Rule of Separation

Separate Policy from Mechanism. Separate Interfaces from Engines

- ~800: Average number of rules
- 9 rules added, only 4 removed each month
- 10%: average rulebase churn
- 70%: Number of FW admins that claim undetected errors
- 95%: Number of FW breaches caused by misconfigurations, not flaws
Rule of Separation

Separate Policy from Mechanism. Separate Interfaces from Engines
Rule of Simplicity

Design for Simplicity. Add Complexity Only Where You Must
Rule of Simplicity

Design for Simplicity. Add Complexity Only Where You Must
Rule of Transparency

Design for Visibility to make inspection and debugging easier
Rule of Transparency
Rule of Representation

*Fold Knowledge into Data so Program Logic can be Stupid and Robust*

Application Whitelisting
Rule of Economy

*Programmer Time is Expansive. Conserve it in Preference To Machine Time*

```javascript
var table="<tr><th>Artist</th><th>Title</th></tr>
        var x = xmlDoc.getElementsByTagName("CD");
```

```javascript
function myFunction(xml) {
    var i;
    var xmlDoc = xml.responseXML
```
Rule of Economy

*Programmer Time is Expansive. Conserve it in Preference To Machine Time*

Application Whitelisting can Really suck
Rule of Economy

Programmer Time is Expansive. Conserve it in Preference To Machine Time

Multi Dimensional
Rule of Economy

*Programmer Time is Expansive. Conserve it in Preference To Machine Time*

A graph can be drawn without lifting the pencil off the paper and without drawing any line twice.
Rule of Extensibility

Design for the Future, because it will be here sooner than you think.
Rule of Extensibility

<table>
<thead>
<tr>
<th></th>
<th>’90s</th>
<th>Late ’90s–’00s</th>
<th>Today</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servers</td>
<td>Static</td>
<td>Dynamic</td>
<td>Dynamic and Distributed</td>
</tr>
<tr>
<td>Environment</td>
<td>Static</td>
<td>Dynamic</td>
<td>Dynamic and Distributed</td>
</tr>
<tr>
<td>Change</td>
<td>Low</td>
<td>High</td>
<td>Continuous</td>
</tr>
<tr>
<td>Ownership</td>
<td>Full</td>
<td>Full</td>
<td>Partial / Rented</td>
</tr>
<tr>
<td>Security</td>
<td>Gateway ACLs</td>
<td>ACL Sprawl</td>
<td>?? ?? ??</td>
</tr>
</tbody>
</table>
Rule of Generation

Avoid Hand Hacking. Write Programs to Write Programs when you can.

MESOSPHERE

ANSIBLE

puppet labs

chef

2016 The Year of the API
Rule of Generation

Avoid Hand Hacking. Write Programs to Write Programs when you can

API Keys

Entropy  Speed  Reduced Exposure  Rotation

TLS at minimum  OAuth 1.0a
Rule of Optimization

Prototype Before Polishing. Get it Working Before you Optimize it

Everybody has a plan until they get punched in the face
Rule of Optimization

Prototype Before Polishing. Get it Working Before you Optimize it

I Hate Whitelisting! Deny All

Model EVERYTHING
Rule of Diversity

Distrust All Claims for “One True Way”
Rule of Being Groovy

References Man

**Whitelisting:**  https://www.nsa.gov/ia/_files/factsheets/Application_Whitelisting_Trifold.pdf  

**Graph Theory:**  http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2010/video-lectures/lecture-8-graph-theory-ii-minimum-spanning-trees/

**Awesome Careers!**  http://www.illumio.com
Thank The Maker

@esrtweet

Computer science education cannot make anybody an expert programmer any more than studying brushes and pigment can make somebody an expert painter.

— Eric S. Raymond —

http://www.catb.org/esr/
< End of Line >