Richard Johnson
richardj@microsoft.com

Opening Questions

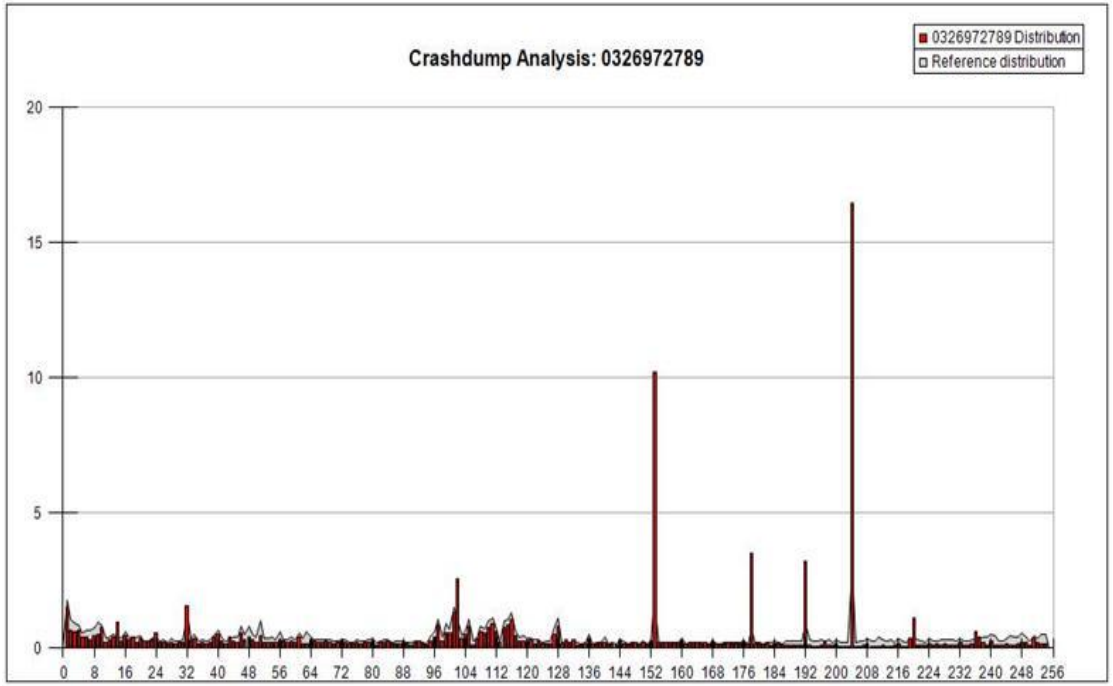
- How can we use the visualization tools we currently have more effectively?
- How can the Software Development Lifecycle benefit from visualizations?
- What is the impact of visualizations on our software security processes?

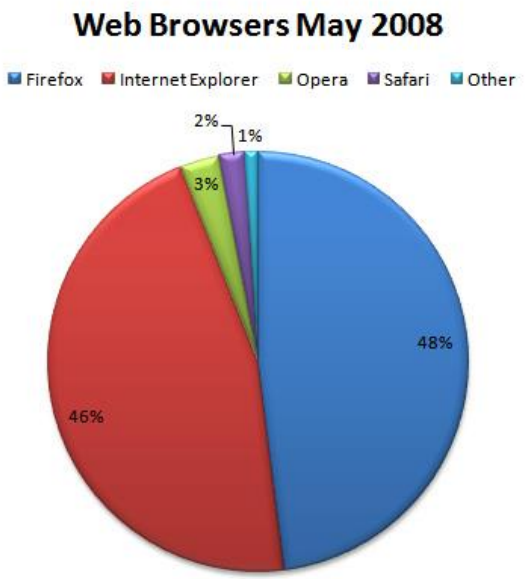
Visualization 101

- What is visualization?
  - Information transmission through imagery
- Why is visualization important?
  - Visualizations utilize the mind's most perceptive input mechanism
- What are the challenges in visualization?
  - Create intuitive spatial mappings of non-spatial data
  - Retain clarity while presenting highly dimensional data

Visualization Taxonomy

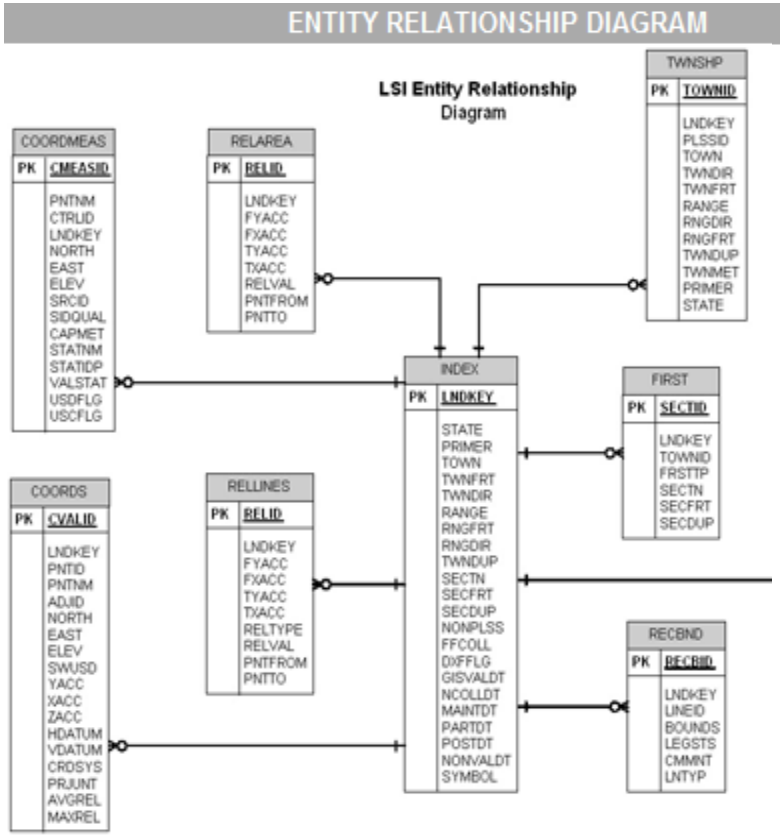
- Data Visualization

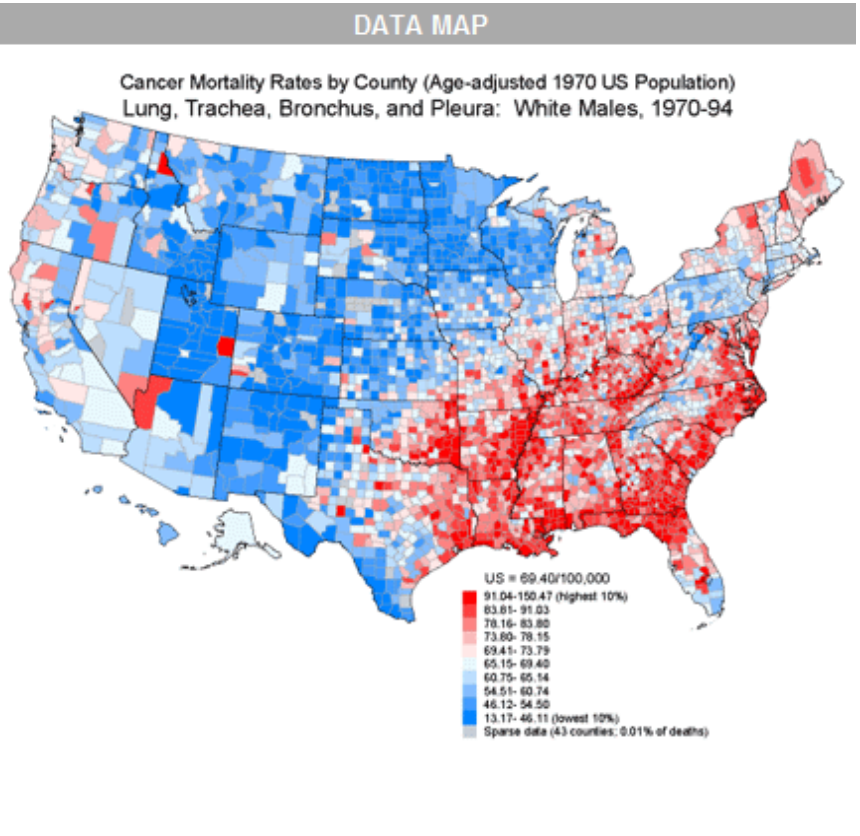




Visualization Taxonomy

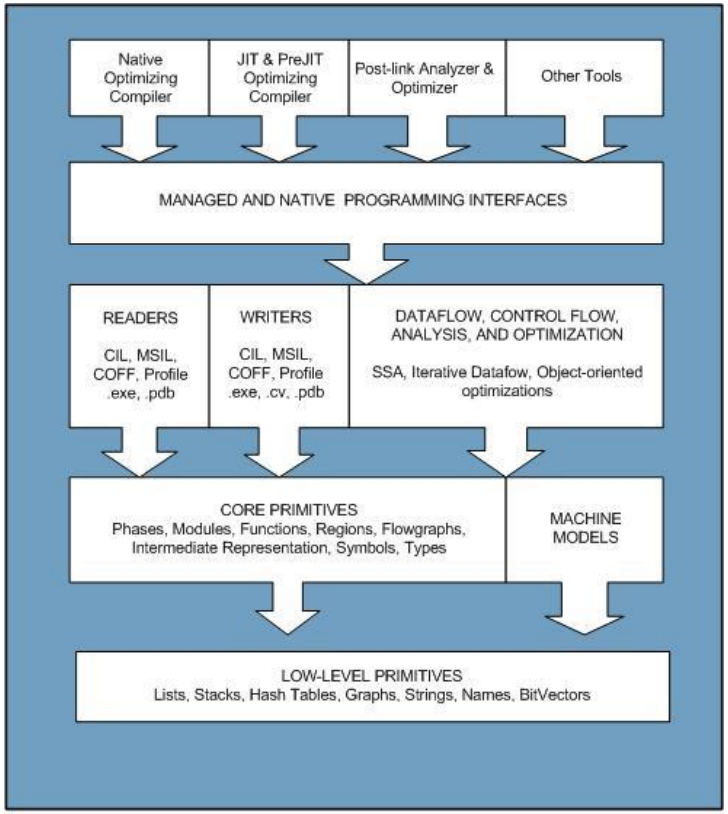
- Information Visualization

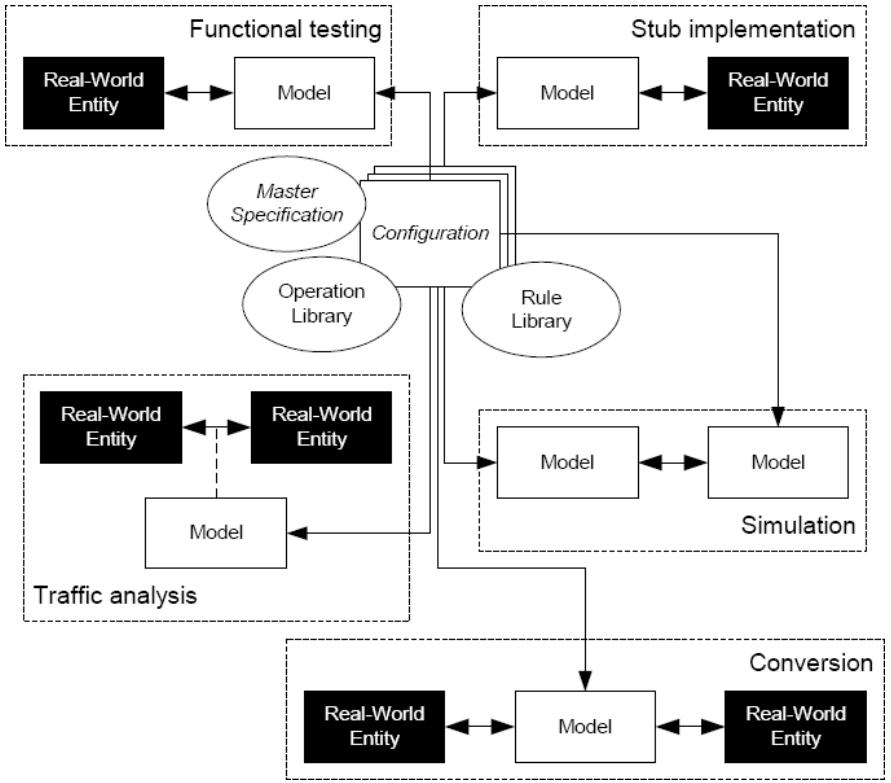




Visualization Taxonomy

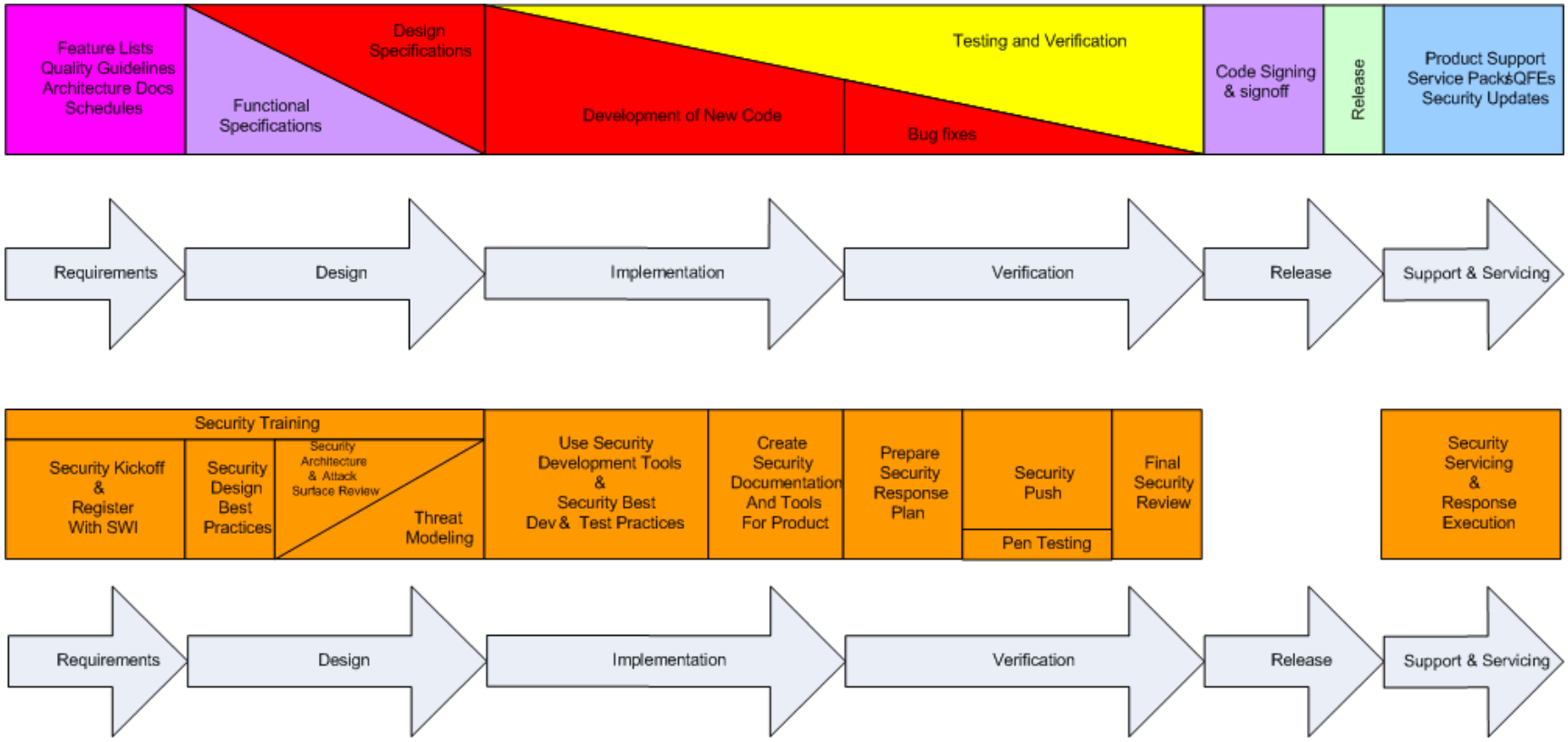
- Concept Visualization





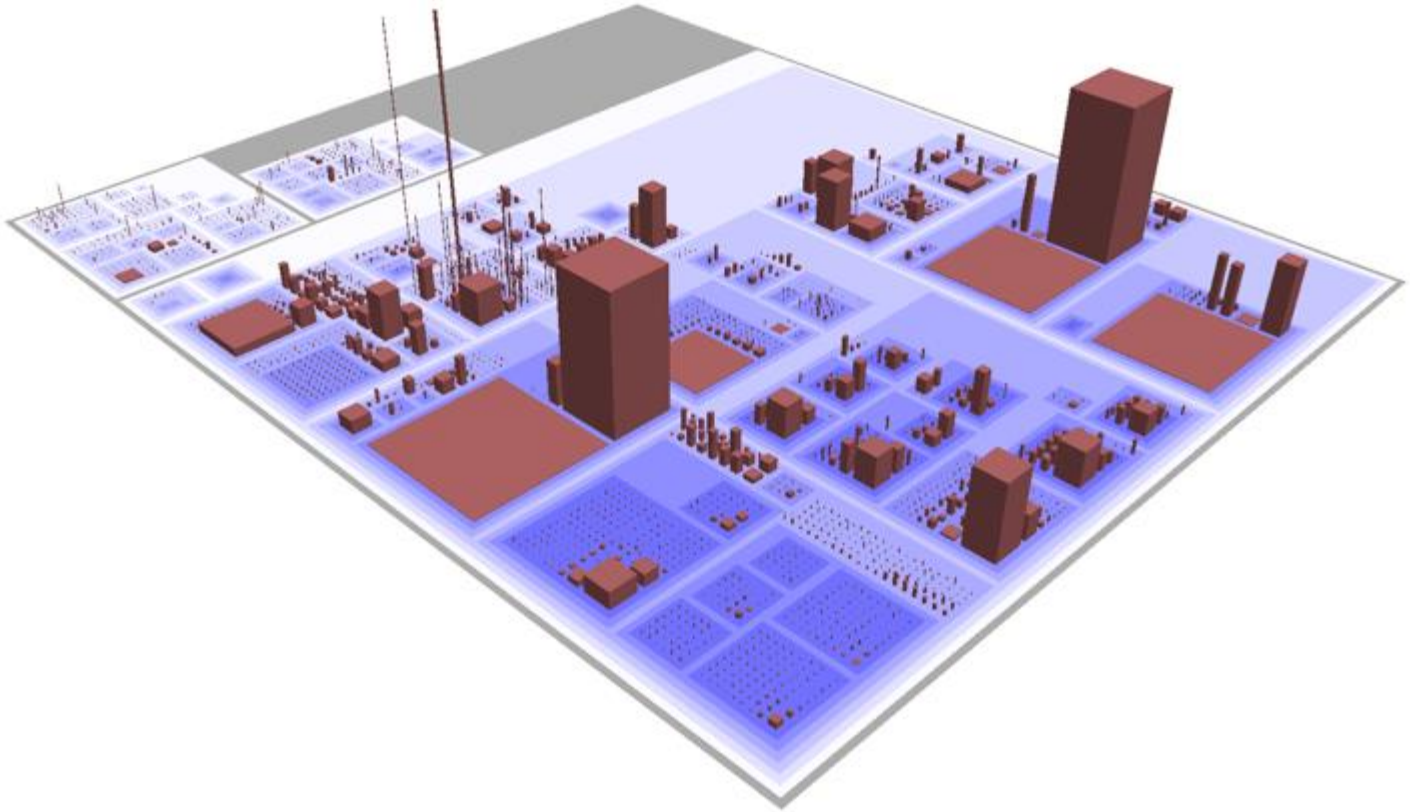
Visualization Taxonomy

- Strategy Visualization



Visualization Taxonomy

- Metaphor Visualization



Software Visualization

- Problem Space
  - Program Visualization
  - Algorithm Visualization
- Sourcing Data
  - Static vs Dynamic data
  - Inaccurate analysis tools
- The goal is always: Reduce Complexity!

Static Software Properties

- Structural Connectivity
  - Execution & Data Flow
  - Class Hierarchies
- State Machine Models
  - Memory profile
  - Algorithm Complexity
- Revision History
  - Age and authorship
  - Milestones in quality assurance

Dynamic Software Properties

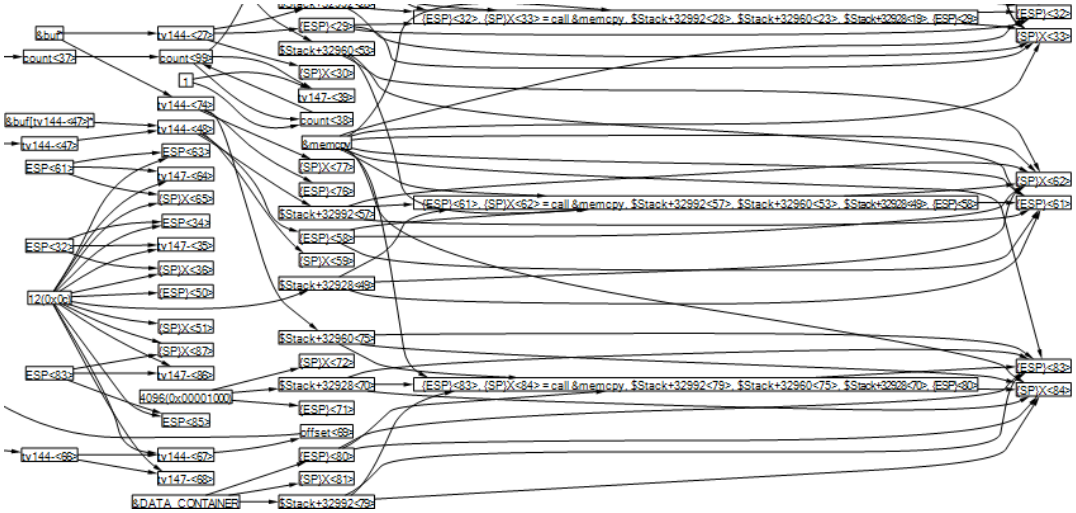
- Execution tracing
  - Code coverage
  - Indirect relationships
  - Dynamic dependencies
- Memory tracing
  - Heap management patterns
  - Object instances
  - Taint propagation
- Environment

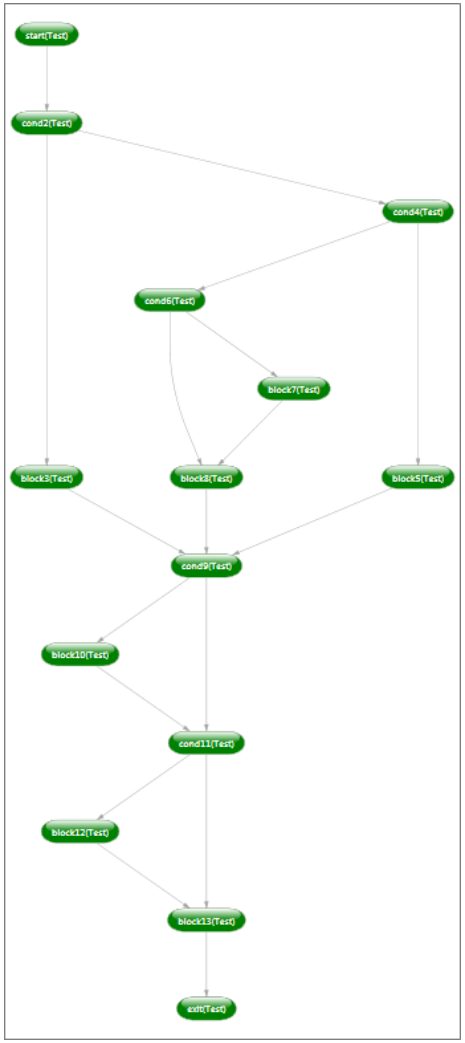
Software Security Properties

- Attack Surface Area
  - Dataflow entry points
  - Privilege boundaries
- Exploitability
  - Execution environment
  - Compiler security
  - Reachability
- Implementation Flaws
  - Arithmetic flaws
  - Comparison flaws
  - Unchecked user input
- History
  - Code age
  - Author credibility

Graph Visualization

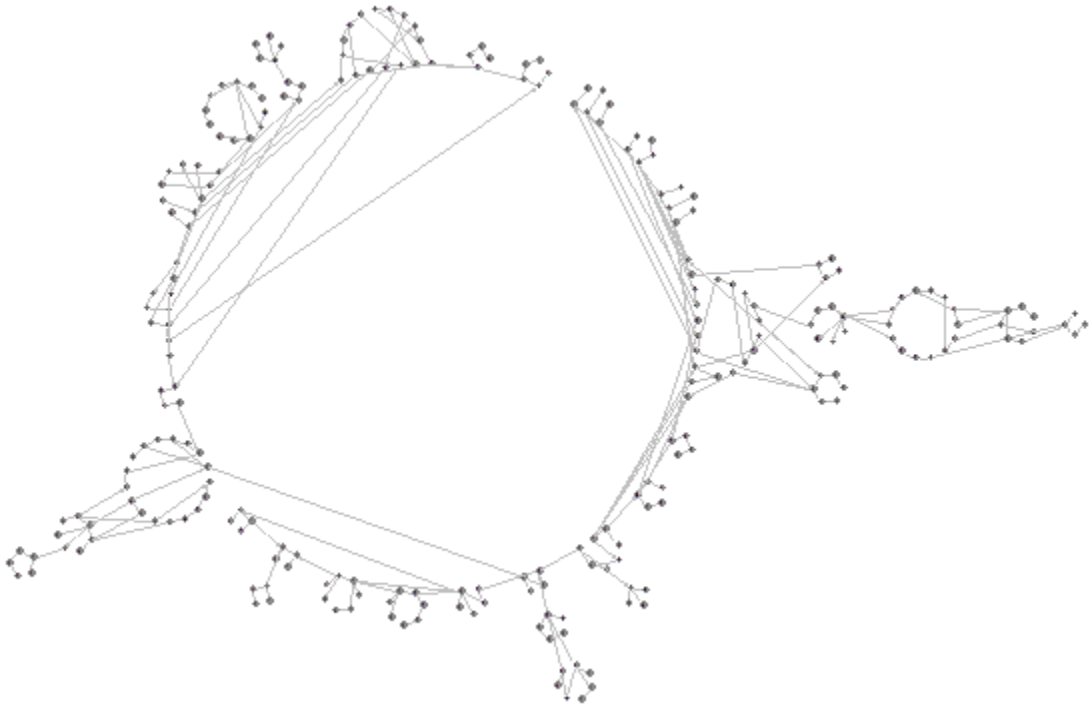
- Hierarchical Layout
  - Layered by order of connectedness
  - Not for highly connected graphs





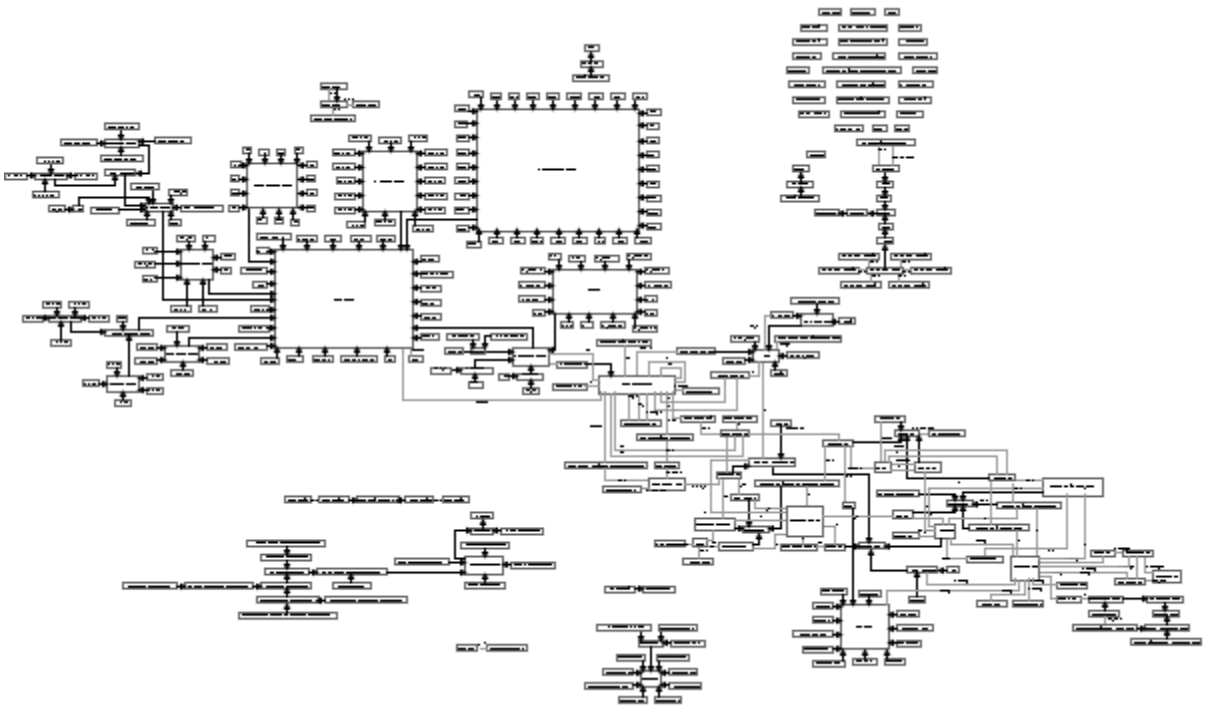
Graph Visualization

- Circular
  - Nodes aligned on circles
  - Clustering



Graph Visualization

- Orthogonal
  - Edges aligned on axes
  - Clustering

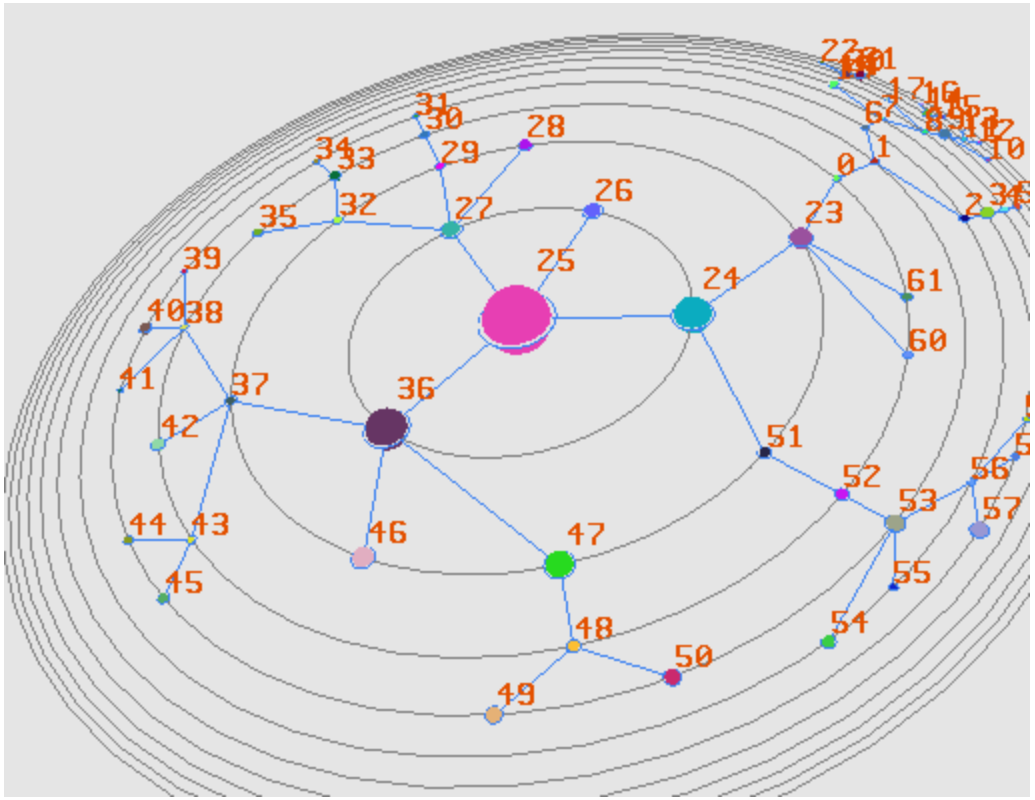


Graph Visualization

- Force Directed
  - Spring, Magnetic, and Gravitational force
  - Packing

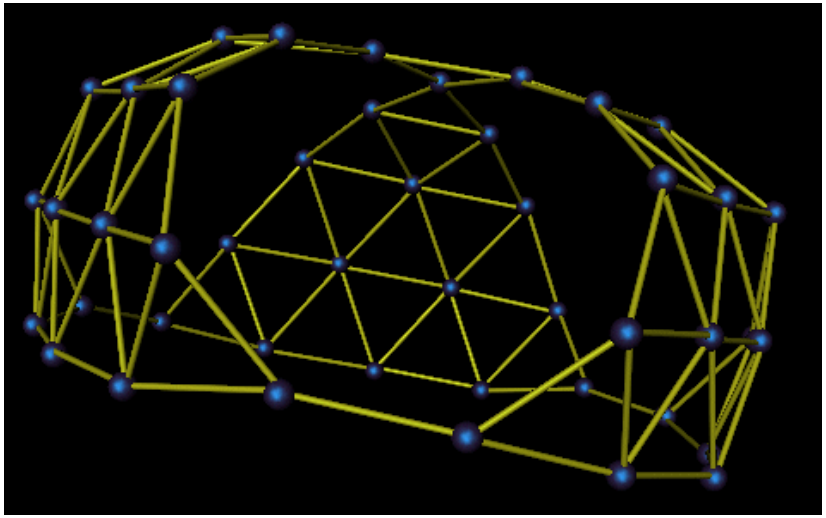
Improved Graph Visualization

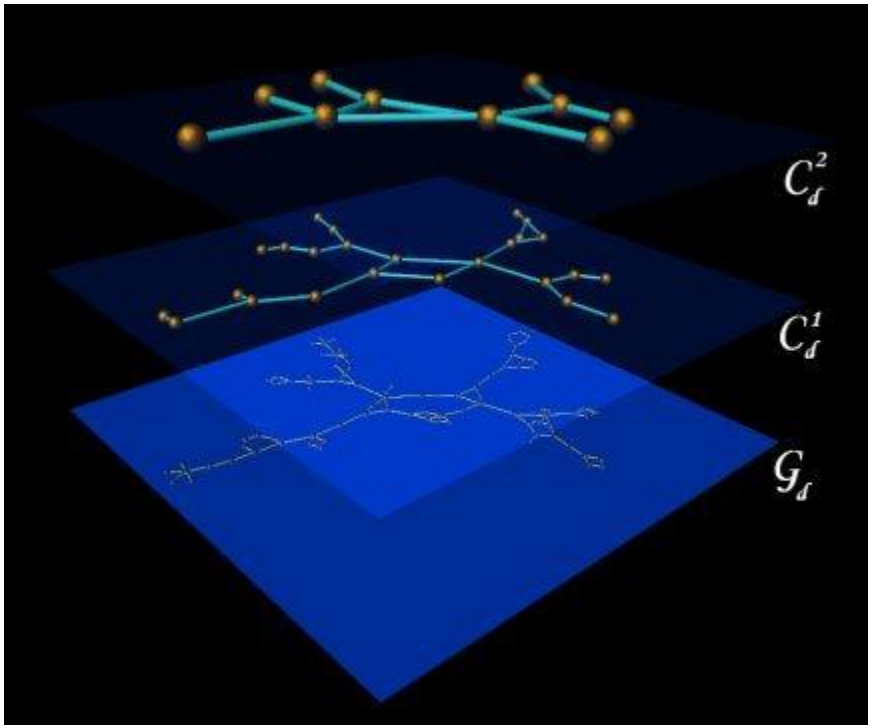
- Hyperbolic Space
  - Clarity on center focus
  - Packing



Improved Graph Visualization

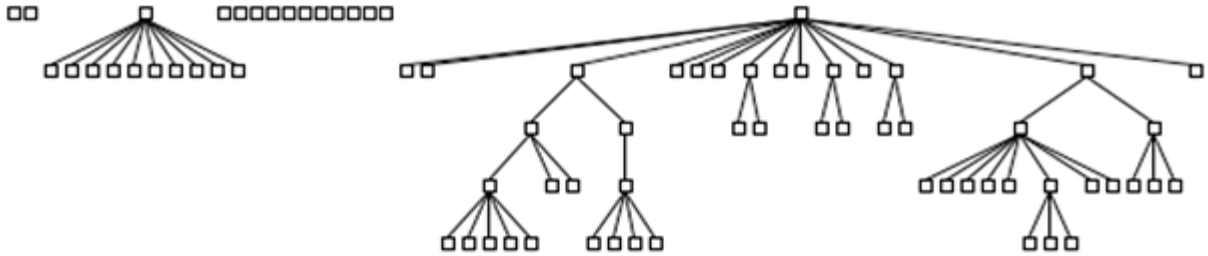
- Higher Dimensional Space
  - Clarity with high connectivity
  - Multi-level views





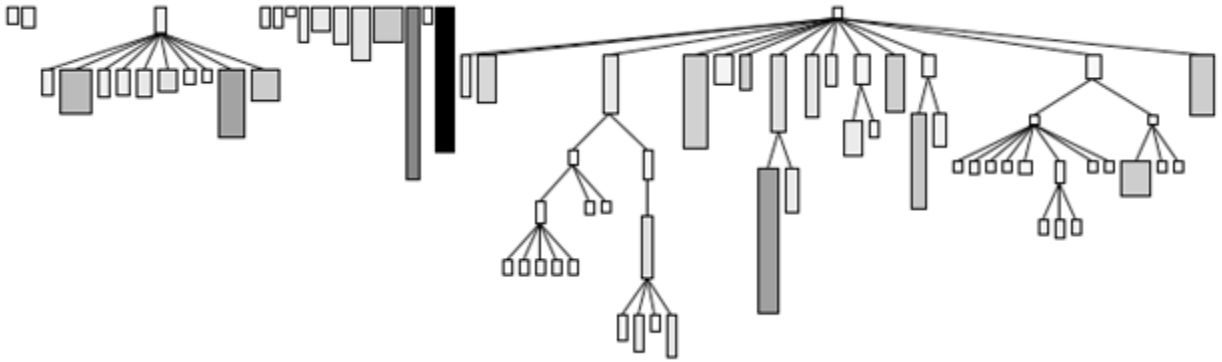
Visual Attributes

- Nodes
  - Spatial coordinates
  - Spatial extents
  - Color
  - Shape
- Edges
  - Color
  - Shape
  - Width
  - Style



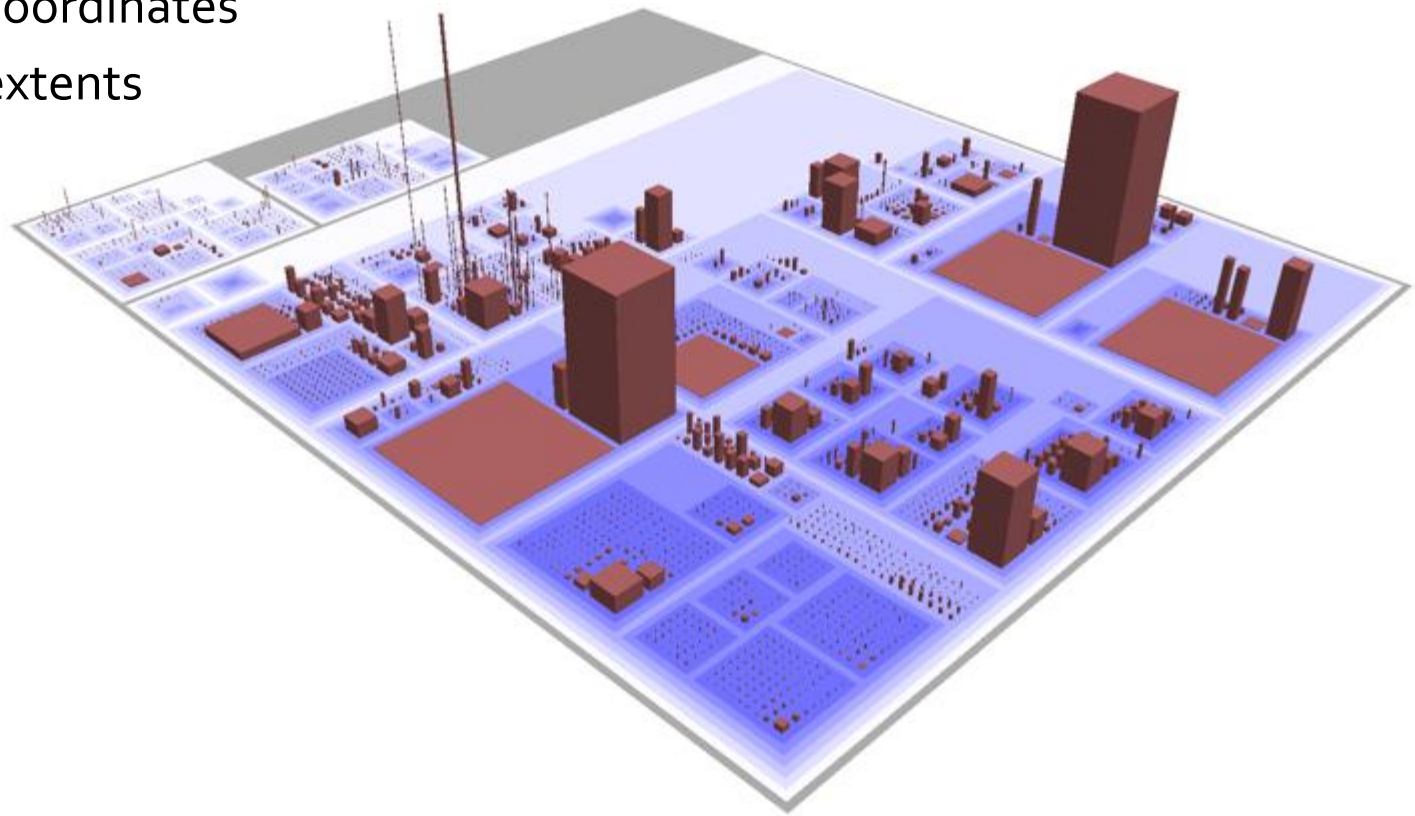
Visual Attributes

- Nodes
  - Spatial coordinates
  - Spatial extents
  - Color
  - Shape
- Edges
  - Color
  - Shape
  - Width
  - Style



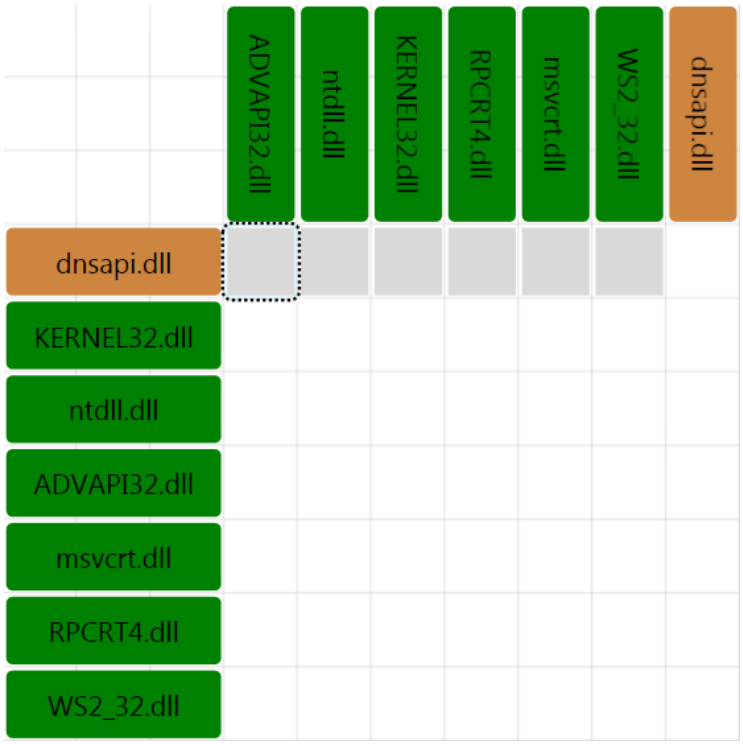
Visual Attributes

- Nodes
  - Spatial coordinates
  - Spatial extents
  - Color
  - Shape
- Edges
  - Color
  - Shape
  - Width
  - Style



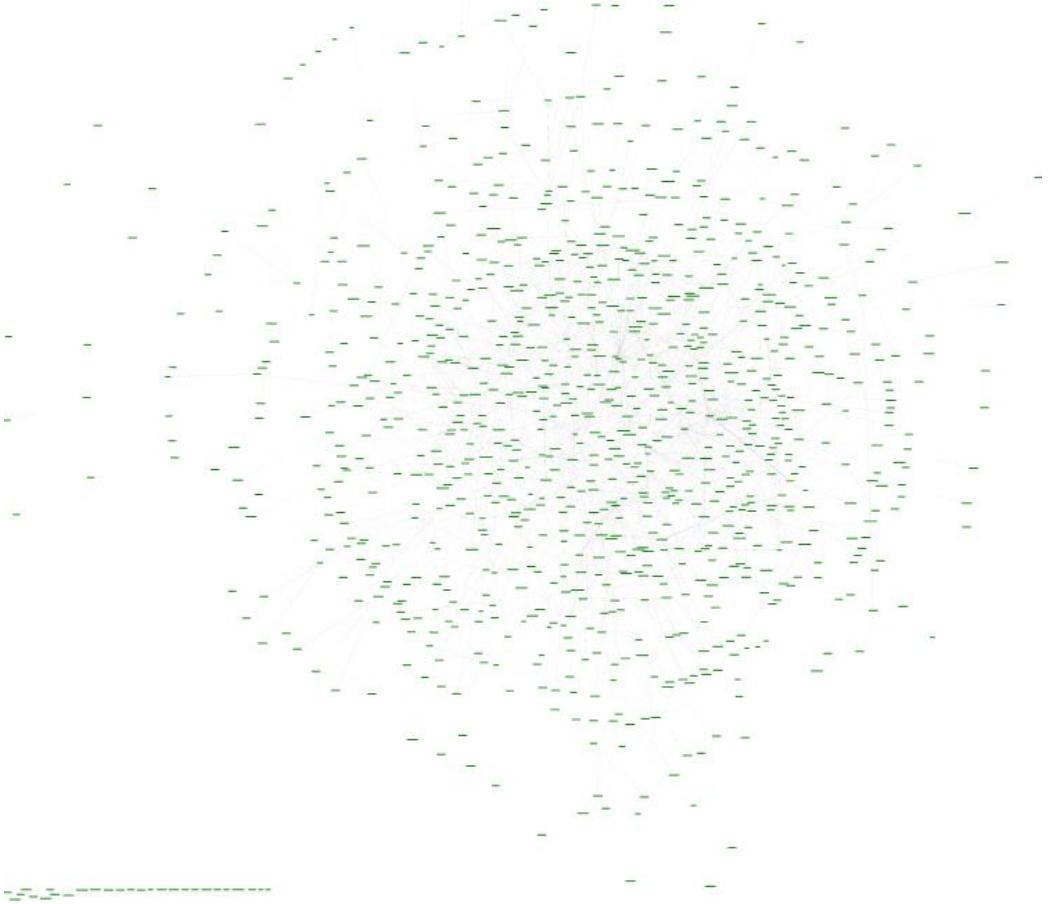
Visualizing Software Security

- Observe binary interdependencies



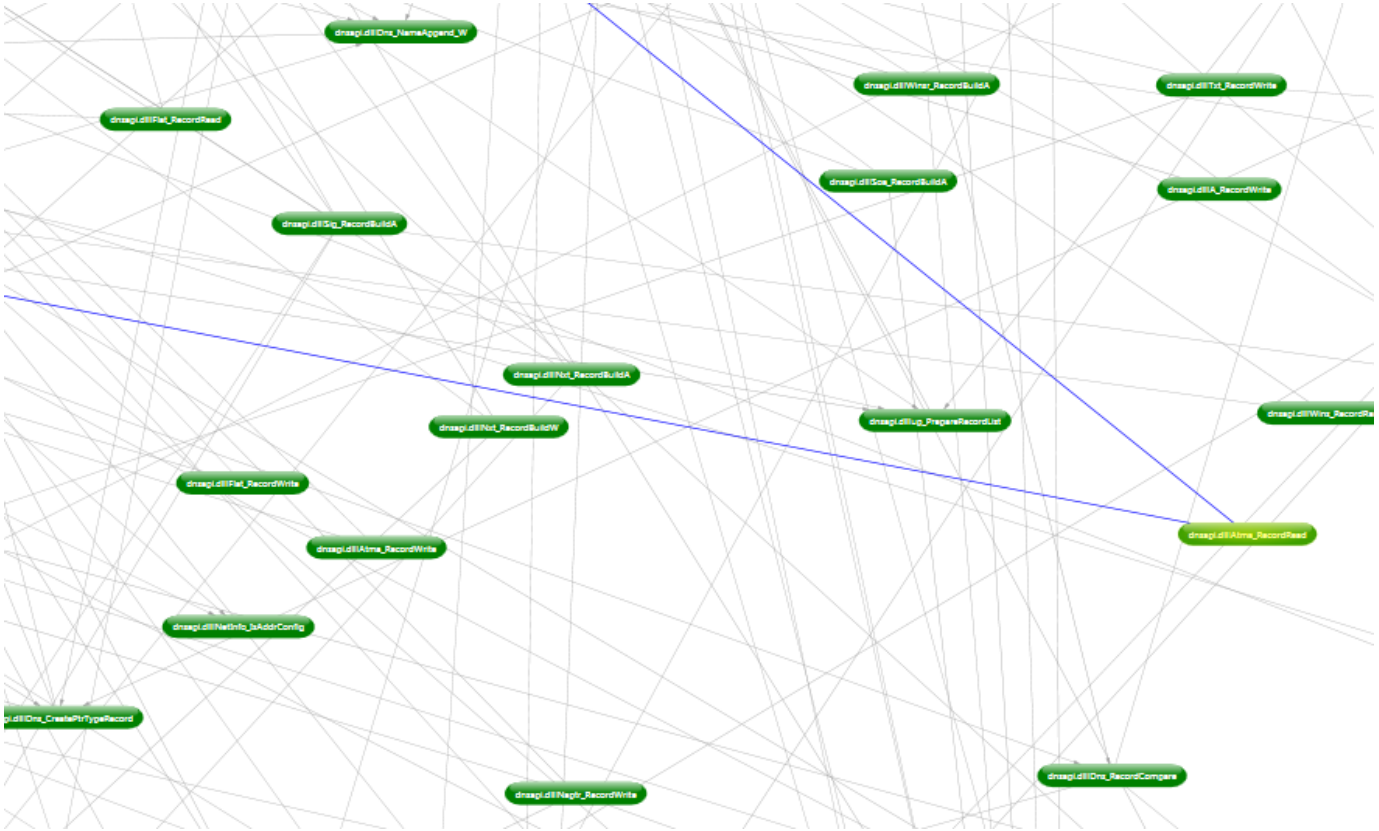
Visualizing Software Security

- Acquire a method level control flow graph



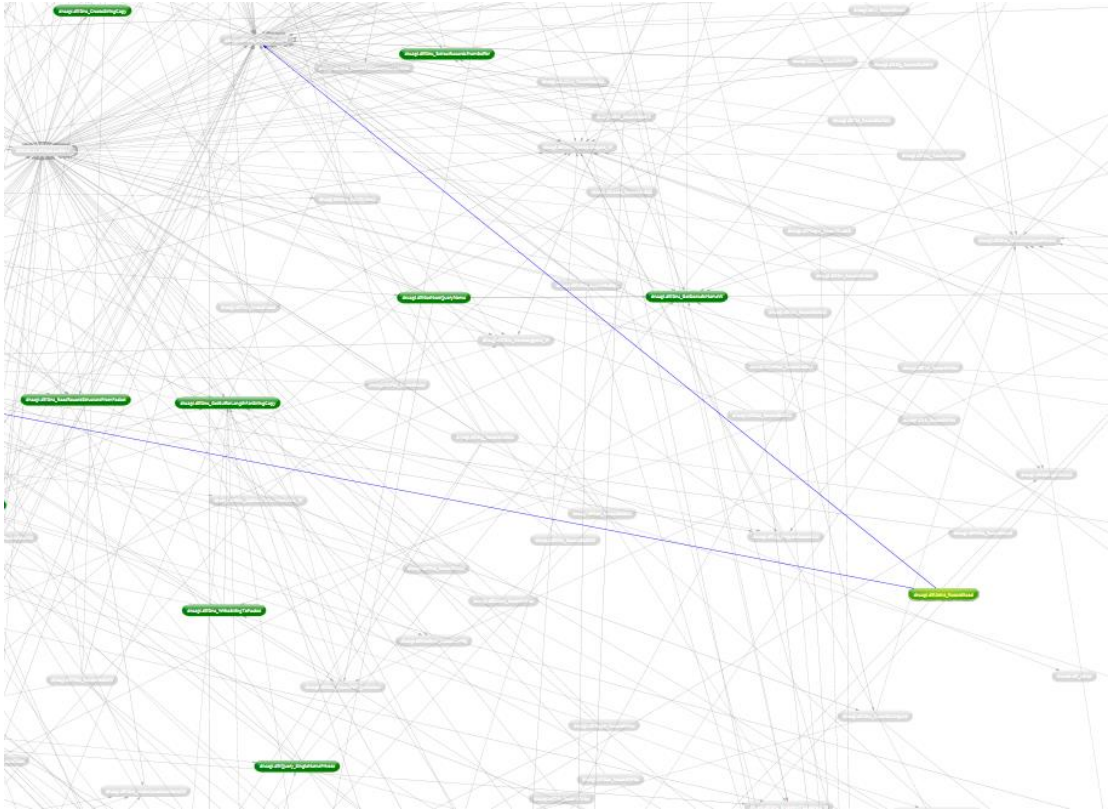
Visualizing Software Security

- Acquire a method level control flow graph



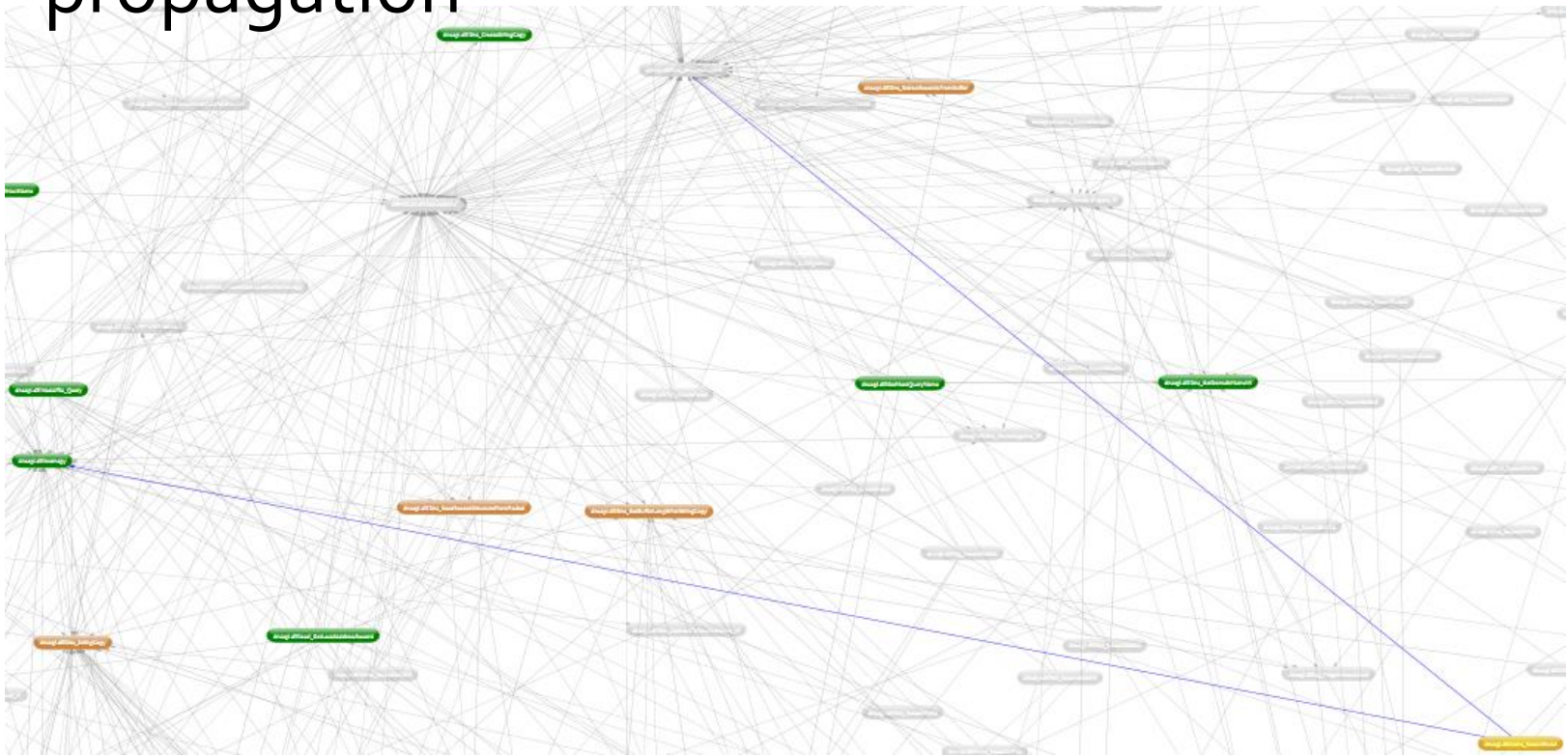
Visualizing Software Security

- Reduce graph using code coverage data



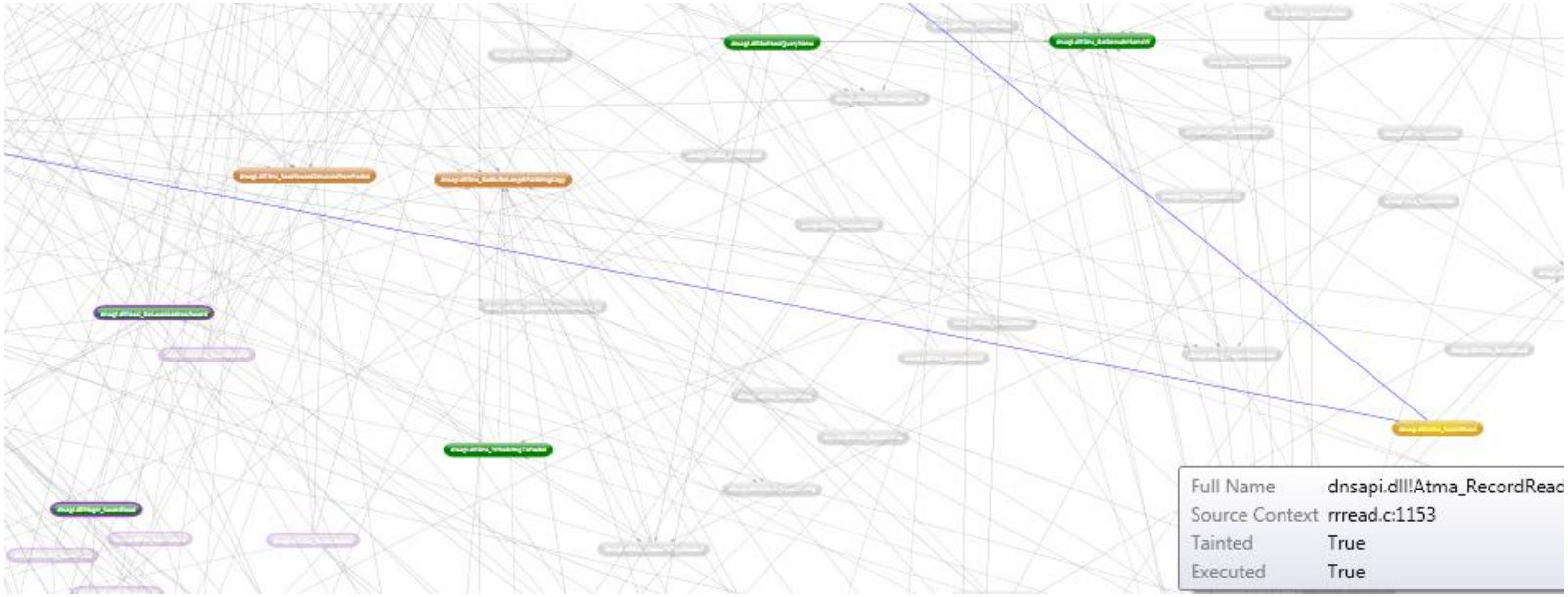
Visualizing Software Security

- Trace dataflow dependency to discover taint propagation



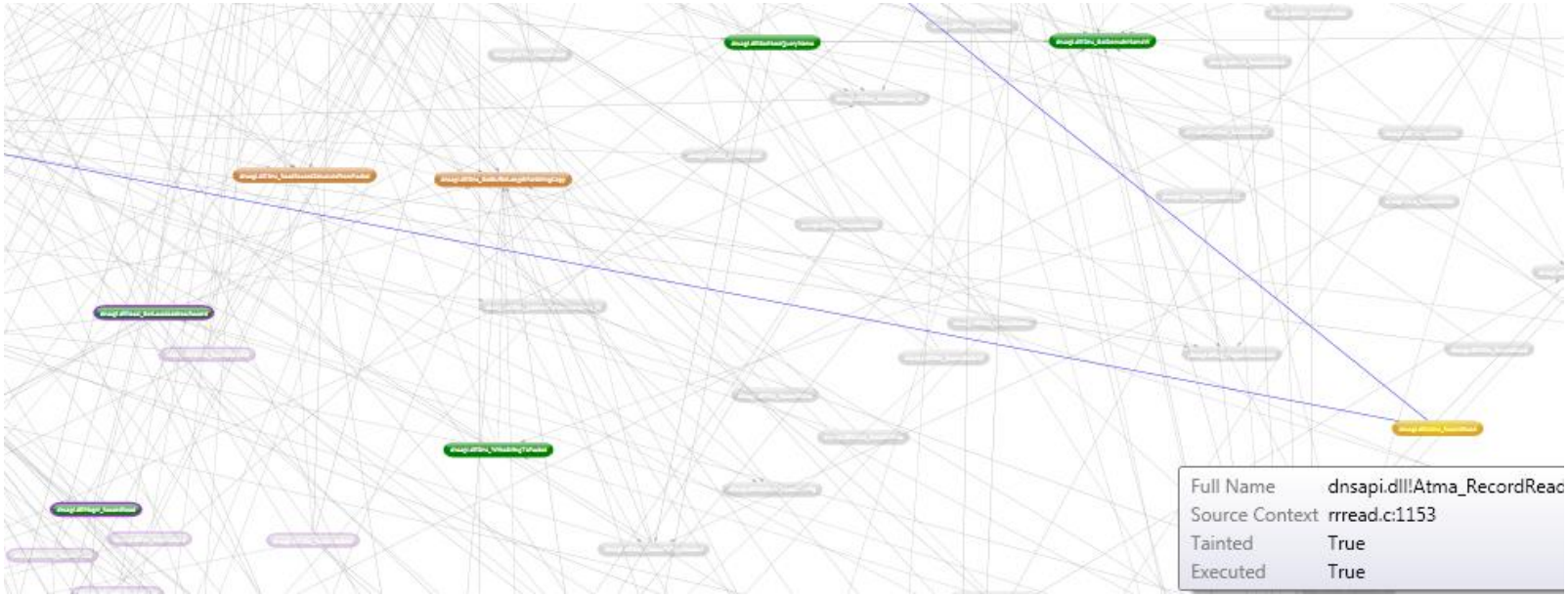
Visualizing Software Security

- Use static analysis plugins to derive security properties such as GS and SafeSEH



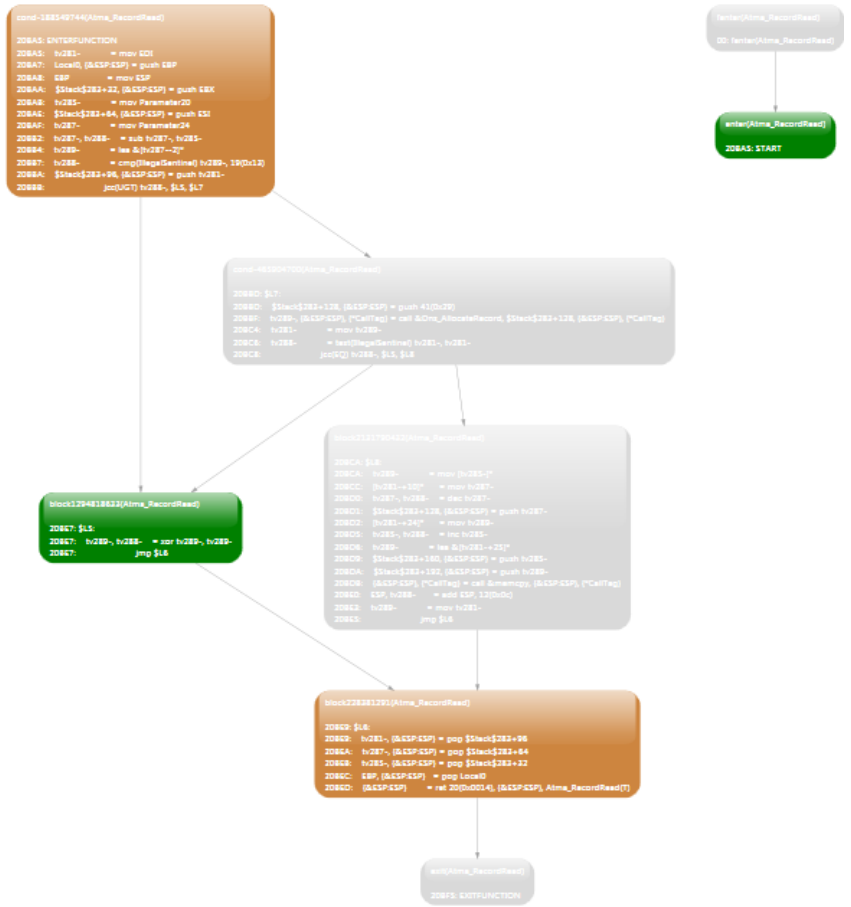
Visualizing Software Security

- Use static analysis plugins to derive security properties such as GS and SafeSEH



Visualizing Software Security

- Analyze non-covered paths in tainted functions

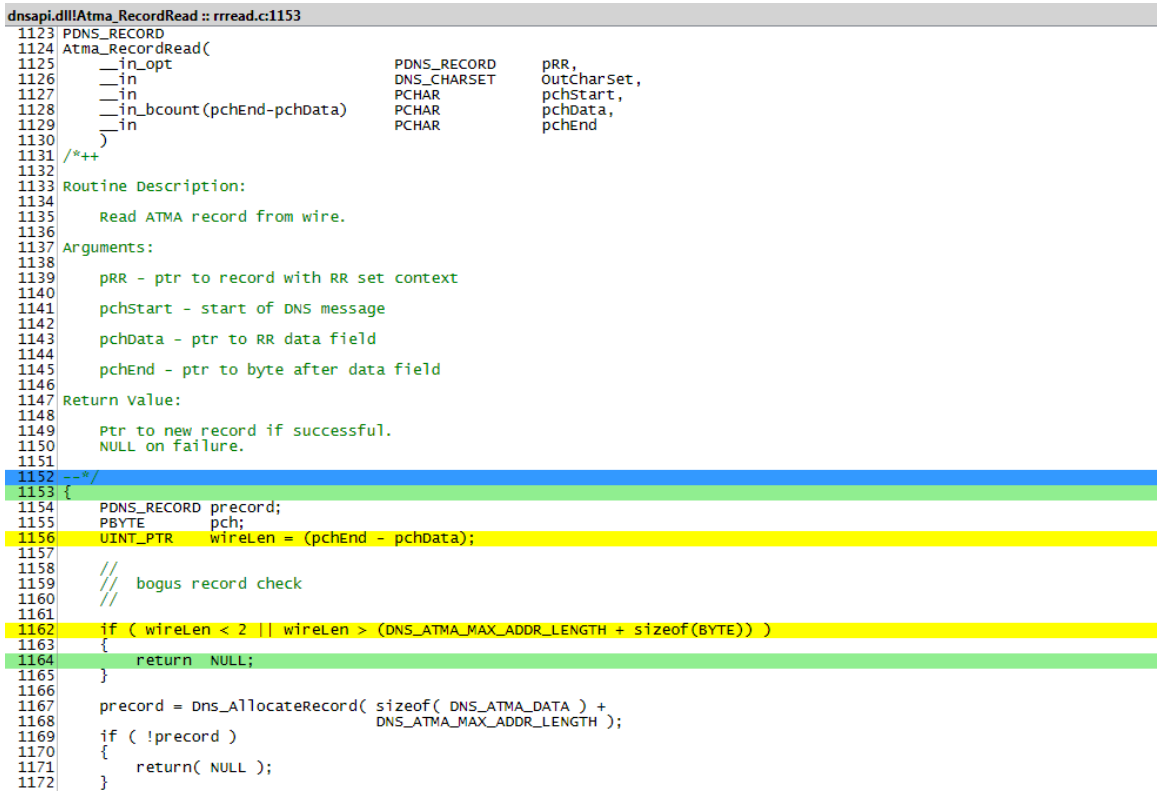


Visualizing Software Security

- Analyze non-covered paths in tainted functions

Visualizing Software Properties

- Examine source code where correlations occur

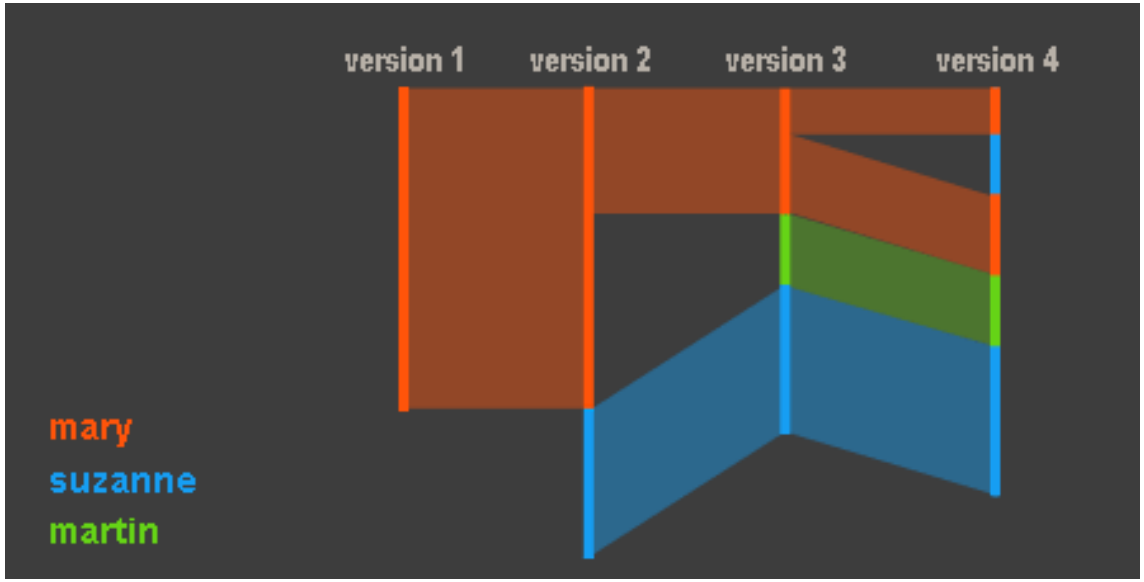


dnsapi.dll!Atma_RecordRead :: rrread.c:1153

```
1123 PDNS_RECORD
1124 Atma_RecordRead(
1125     __in_opt                        PDNS_RECORD     pRR,
1126     __in                            DNS_CHARSET     OutCharSet,
1127     __in                            PCHAR           pchStart,
1128     __in_bcount(pchEnd-pchData)     PCHAR           pchData,
1129     __in                            PCHAR           pchEnd
1130     )
1131 /*++
1132
1133 Routine Description:
1134
1135     Read ATMA record from wire.
1136
1137 Arguments:
1138
1139     pRR - ptr to record with RR set context
1140
1141     pchStart - start of DNS message
1142
1143     pchData - ptr to RR data field
1144
1145     pchEnd - ptr to byte after data field
1146
1147 Return Value:
1148
1149     Ptr to new record if successful.
1150     NULL on failure.
1151
1152 --*/
1153 {
1154     PDNS_RECORD precord;
1155     PBYTE       pch;
1156     UINT_PTR    wireLen = (pchEnd - pchData);
1157
1158     //
1159     //  bogus record check
1160     //
1161
1162     if ( wireLen < 2 || wireLen > (DNS_ATMA_MAX_ADDR_LENGTH + sizeof(BYTE)) )
1163     {
1164         return  NULL;
1165     }
1166
1167     precord = Dns_AllocateRecord( sizeof( DNS_ATMA_DATA ) +
1168                                   DNS_ATMA_MAX_ADDR_LENGTH );
1169     if ( !precord )
1170     {
1171         return( NULL );
1172     }
```

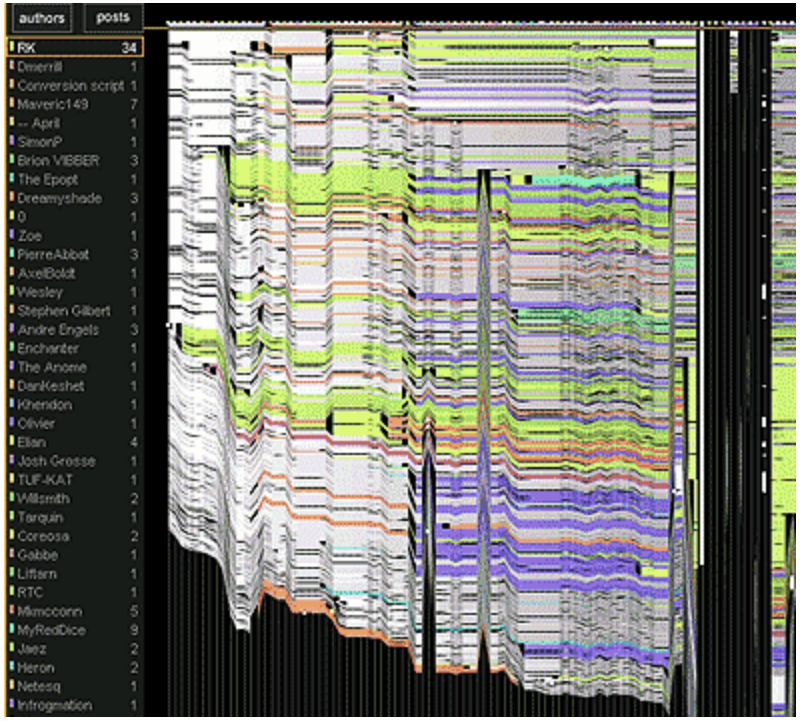
Beyond Graphs

- Source Code Revision History
  - History Flow



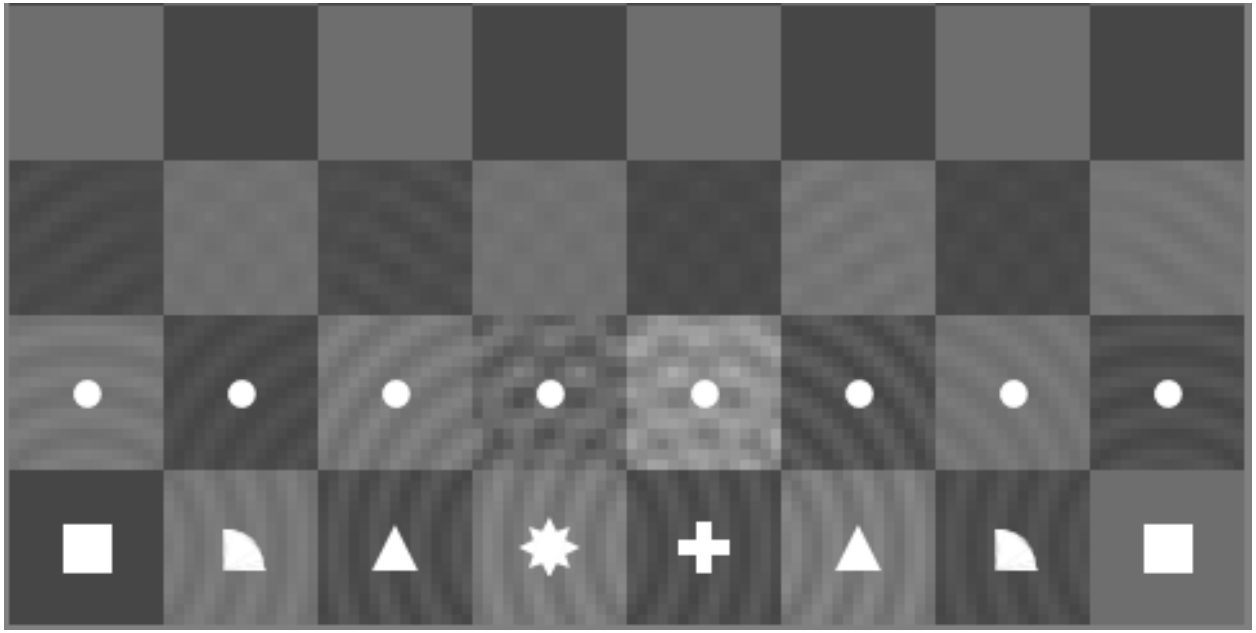
Beyond Graphs

- Source Code Revision History
  - History Flow



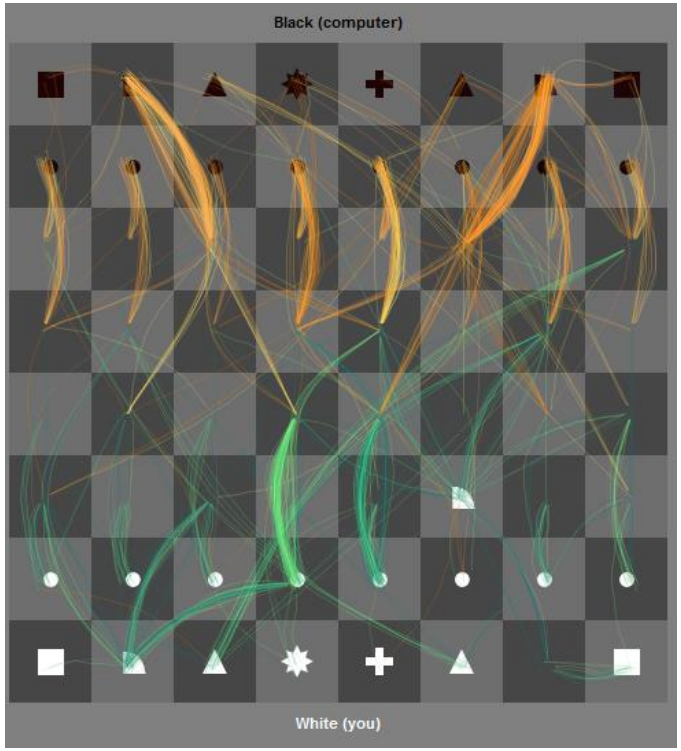
Beyond Graphs

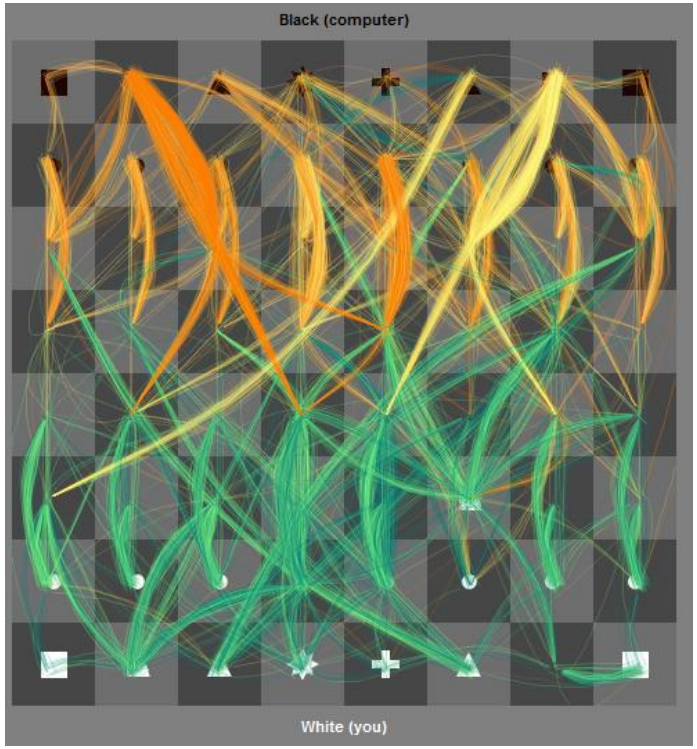
- State Machine Models
  - Thinking Machine



Beyond Graphs

- State Machine Models
  - Thinking Machine





Richard Johnson
richardj@microsoft.com

Richard Johnson
richardj@microsoft.com