21

Computer Security

Olga Ohrimenko
Brown University

Charalampos Papamanthou
University of California, Berkeley

Bernardo Palazzi
Brown University and Italian National Institute of Statistics

21.1 Introduction

As the number of devices connected to the Internet continues to grow rapidly and software systems are being increasingly deployed on the Web, security and privacy have become crucial properties for networks and applications. Because of the complexity and subtlety of cryptographic methods and protocols, software architects and developers often fail to incorporate security principles in their designs and implementations. Also, most users have minimal understanding of security threats. While several tools for developers, system administrators, and security analysts are available, these tools typically provide information in the form of textual logs or tables, which are cumbersome to analyze. Thus, in recent years, the field of security visualization has emerged to provide novel ways to display security-related information, thus making such information easier to understand.

Securing computers and cyberspace is one of today's grand challenges for science and engineering. Computers and networks are under continuous threat from attackers who want to steal credit card numbers, intellectual property, and other sensitive information. Also, massive distributed denial of service attacks can impair even the largest of companies and government organizations.

Computer security research aims at developing methods and associated protocols to analyze and defend against a growing number and variety of attacks. The development of

security tools is a continuous process that keeps on reacting to newly discovered hardware and software vulnerabilities and newly deployed technologies.

21.1.1 Motivation

Both the discovery of vulnerabilities and the development of security protocols can be greatly assisted by visualization. For example, network traffic can be naturally displayed as a graph whose nodes are hosts and whose edges are associated with packets going from one host to another. Also, a visual representation of a complex multiparty security protocol can give experts better intuition of its execution and security properties. Traditionally, instead, computer security analysts read through large logs produced by applications, operating systems, and network devices. Inspecting such logs is quite cumbersome and often unwieldy, even for experts. Motivated by the growing need for automated visualization methods and tools for computer security, the field of *security visualization* has recently emerged as an interdisciplinary community of researchers with its own annual meeting (VizSec).

For basic background on computer security, see the textbook by Goodrich and Tamassia [GT11]. The book by Raffael Marty [Mar08] provides an excellent introduction to methods and tools for visualizing computer networks to analyze their security.

21.1.2 Chapter Organization

In this chapter, we give a survey of approaches to the visualization of computer security concepts that use graph drawing techniques. We consider a variety of fundamental security and privacy issues, focusing on network security, access control, and attack strategies. We show how graphs can be used as an effective modeling tool in computer security and we give examples of how several classic graph drawing techniques have been used in current security visualization prototypes. Finally, we mention an approach for privacy-preserving drawing in a cloud computing scenario.

Thanks to their versatility, graph drawing techniques are one of the main approaches employed in security visualization. Indeed, not only computer networks are naturally modeled as graphs, but also data organization (e.g., file systems) and vulnerability models (e.g., attack trees) can be effectively represented by graphs. In particular, we consider the following security visualization problems:

1. *Network monitoring.* (Section 21.2) The visualization of network traffic helps network administrators identify anomalous patterns, such as scans, worm infections, and hosts trying to gain unauthorized access to the network. Thus, it is an effective component of intrusion detection systems. Also, traffic visualization can be used to identify unusually heavy network activity and quickly track down machines that generate or receive a large volume of packets. Early detection is crucial when defending against denial of service attacks. It is also interesting to monitor and visualize the evolution of highly dynamic services on the Internet, such as the root name servers.
2. *Border gateway protocol (BGP).* (Section 21.3) This protocol manages reachability between hosts in different Autonomous Systems, i.e., networks controlled by Internet Service Providers. The visualization of BGP-related information is important to ensure that routing in the Internet has not changed and has not been tampered with. In particular, displaying the topology of Autonomous Systems and the evolution of BGP routing patterns can assist the detection of disruptions in Internet traffic caused by attacks or router configuration errors.

3. *Access control.* (Section 21.4) Access to resources on a computer system or network is regulated by organizational policies and enforced with technological mechanisms for authentication and authorization. Resources need to be protected not only from malicious activity by outside attackers but also from accidental disclosure to unauthorized legitimate users. Access control mechanisms for file systems, databases, and distributed applications are complex and tricky to configure. Visualization helps both users and administrators gain an intuitive understanding of the vast set of permissions that are in place in the system and allows them to efficiently spot sensitive resources that are insufficiently protected. Also, visualizing flows of information in a system can help keeping sensitive data private and defend against the leakage of confidential information. Access control is especially challenging in distributed environments without centralized administrative control. An aspect of access control that is gaining increasing importance is the management of privacy settings by users of a social network.
4. *Attack graphs.* (Section 21.5) Starting with a vulnerable component of a system, an attacker can compromise other components to reach the desired goal. Attack graphs are used to describe dependencies between vulnerabilities in a system. They characterize the paths through the system that can be followed by an adversary. The visualization of attack graphs helps computer security analysts identify and remedy vulnerabilities.

Sections 21.2 through 21.5 are organized around the four security topics mentioned above. For each topic, we overview visualization tools that employ graph drawing techniques. Table 21.1.2 classifies the papers surveyed in these sections according to the security topic addressed and the graph drawing method used.

	Force Directed	**Layered Drawing**	**Bipartite Drawing**	**Circular Drawing**	**Treemap or Gmap**	**3D**
Network Monitoring	[MMK07, TN00, GB98, MMB05, DSN12]		[YYT+04, BFN04, Con07]	[Tol]	[DSN12, BvO09]	[XMB+06]
BGP	[BMPP04, TRNC06]			[TRNC06]		[OKB06]
Access Control	[MLA12]	[MFG+06, Yee06, YSTW05]			[HPPT08]	
Attack Graphs		[NJKJ05, NJ04]			[CIL+10]	

Table 21.1 Classification of the papers on security and privacy visualization surveyed in this chapter according to the security topic addressed and the graph drawing method used.

Finally, in Section 21.6, we take a different perspective and consider the subject of privacy protection when a client outsources the task of drawing a graph to a server in the cloud. We present a technique that provides a high level of privacy, going beyond encryption, and is computationally efficient.

Chapter 24 overviews related work on the visualization of computer networks.

21.2 Network Monitoring

In this section, we overview selected papers on graph-based visualization techniques for network monitoring. Related work includes, e.g., [FMK+08, MFK+09].

21.2.1 Intrusion Detection

In [TN00], the authors use a combination of force-directed drawing, graph clustering, and regression-based learning in a system for intrusion detection (see Figure 21.1). Their system consists of the following components:

- a packet collecting module;
- a graph construction and clustering module;
- a visualization module; and
- an event generation module.

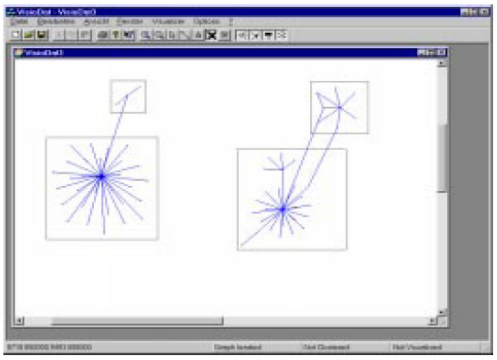


Figure 21.1 Force-directed clustered drawing for intrusion detection (thumbnail of image from [TN00]).

The authors model the computer network with a graph where the nodes are computers and the edges are communication links with weight proportional to the network traffic on that link. The clustering of the graph is performed with a simple iterative method. Initially, every node forms its own cluster. Next, nodes join clusters that already have most of their neighbors, breaking ties at random. The resulting graph is a simplified version of traffic exchanges where entities that communicate often are joined into clusters. Two clusters A and B are connected by an edge if there is at least one edge between some node of cluster A and some node of cluster B.

The classic force-directed spring embedder method [Ead84, FR91] is used to determine the layout of the graph of clusters and of the graph within each cluster. Since forces are proportional to the weights of the edges, if there is a lot of communication between two hosts, their nodes are placed close to each other.

Various features of the clustered graph (including statistics on the node degrees, number of clusters, and internal/external connectivity of clusters) are used to describe the current state of network traffic and are summarized by a feature vector. Using test traffic samples and a regression-based strategy, the system learns how to map feature vectors to intrusion

detection events. The visualization of the clustered graph can help a security analyst in assessing the severity of the intrusion detection events generated by the system.

21.2.2 Traffic Analysis

A tool for visualizing the evolution over time of the volume and type of network traffic using force-directed graph drawing techniques is described in [MMK07] (see Figure 21.2). Since there are different types of traffic protocols (HTTP, FTP, SMTP, SSH, etc.) and multiple time periods, this multidimensional data set is modeled by a graph with two types of nodes: *dimension nodes* represent traffic protocols and *observation nodes* represent the state of a certain host in a given time interval. Edges are also of two types: *trace edges* link observation nodes of consecutive time intervals and *attraction edges* link observation nodes with dimension nodes and have weight proportional to the traffic of that type.

The layout of the above graph is computed starting with a fixed placement of the dimension nodes and then executing a modified version of the Fruchterman-Reingold force-directed algorithm [FR91] that aims at achieving uniform edge lengths. The authors show how intrusion detection alerts can be associated with visual patterns in the layout of the graph.

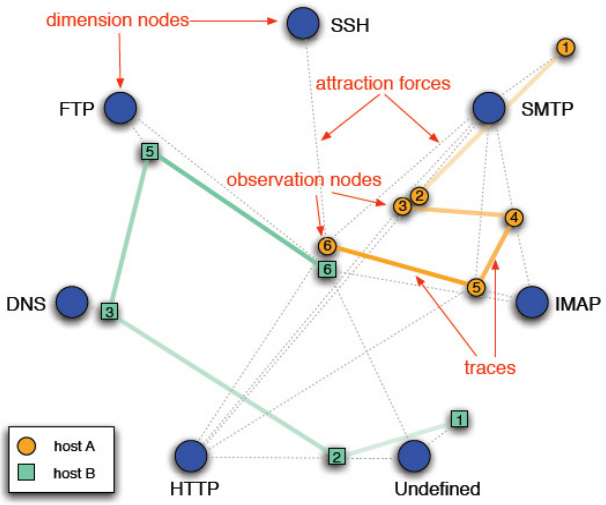


Figure 21.2 Evolution of network traffic over time (thumbnail of image from [MMK07]): dimension nodes represent types of traffic and observation nodes represent the state of a host at a given time.

EtherApe [Tol] shows traffic captured on the network via the pcap interface (Figure 21.3). A simple circular layout places the hosts around a circle and represents network traffic between hosts by straight-line edges between them. Each protocol is distinguished by a different color and the width of an edge shows the amount of traffic. This tool allows to quickly understand the role of a host in the network and the changes in traffic patterns over time. Beyond the graphical representation, it is also possible to display detailed traffic statistics of active ports.

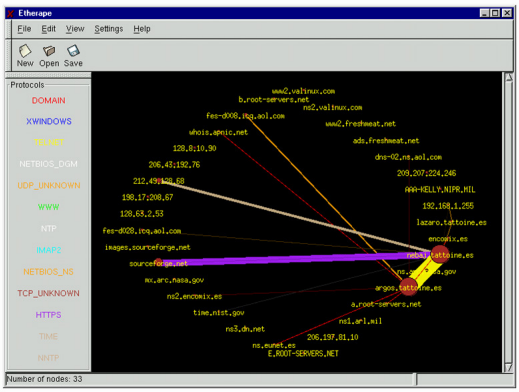


Figure 21.3 Traffic monitoring with Etherape (thumbnail of image from [Tol]). The size of the nodes and the thickness of the edges are proportional to the traffic volume. The color of an edge denotes the prevalent protocol of the associated traffic.

RUMINT [Con07] system (named after RUMor INTelligence) is a free tool for network and security visualization (Figure 21.4). It takes captured traffic as input and visualizes it in various unconventional ways. The most interesting visualization related to graph drawing is a parallel plot that allows one to see at a glance how multiple packet fields are related. An animation feature allows to analyze various trends over time.

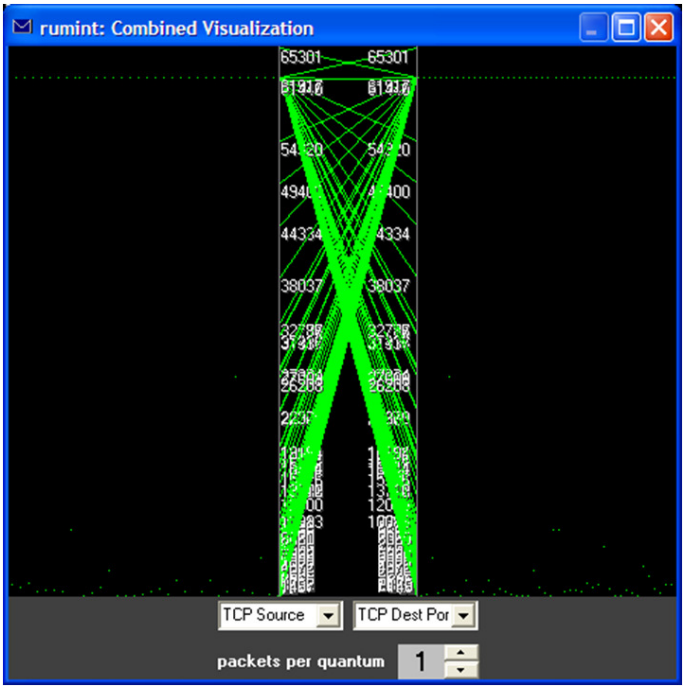


Figure 21.4 Visualization of an NMAP scan with RUMINT (thumbnail of image from [Con07]).

21.2.3 Internal vs. External Hosts

In [YYT+04], the authors apply a simple bipartite drawing technique to provide a visualization solution for network monitoring and intrusion detection (see Figure 21.5). The nodes, representing internal hosts and external domains, are placed on three vertical lines. The external domains that send traffic to some internal host are placed on the left line. The domains of the internal hosts are placed on the middle line. The external domains that receive traffic from some internal host are placed on the right line. Each edge represents a network flow, which is a sequence of related packets transmitted from one host to another host (e.g., a TCP packet stream). Basically, the layout represents a tripartite graph. The vertical ordering of the domains along each line is computed by the drawing algorithm with the goal of minimizing crossings.

The tool uses a slider to display network flows at various time intervals and provides three views. In the global view, the entire tripartite graph is displayed to show all the communication between internal and external hosts. In the internal view and domain view, the tool isolates certain parts of the network, such as internal senders and internal receivers, and correspondingly displays a bipartite graph. The domain view and internal view are easier to analyze and provide more details on the network activity being visualized but on the other hand, the global view produces a high-level overview of the network flows. The authors apply the tool in various security-related scenarios, such as virus outbreaks and denial-of-service attacks.

In [BFN04], the authors use a matrix display combined with a simple graph drawing method in order to visualize the traffic between domains in network and external domains (see Figure 21.6). To visualize the internal network, the authors use a square matrix: each entry of the matrix corresponds to a host of the internal network. External hosts are represented by squares placed outside the matrix, with size proportional to the traffic sent or received.

Straight-line edges represent traffic between internal and external hosts and can be colored to denote the predominant direction of the traffic (outgoing, incoming, or bidirectional). The placement of the squares arranges hosts from subnets of the same size along the same vertical line and attempts to reduce the number of edge crossings. Further details on the type of traffic can be also displayed in this tool. For example, vertical lines inside each square indicate ports with active traffic. This system can be used to visually identify traffic patterns associated with common attacks, such as virus outbreaks and network scans.

21.2.4 Similarity Analysis for Traffic Logs and Scans

In [GB98], the authors present a technique to visualize log entries obtained by monitoring network traffic. Each log entry stores a multidimensional vector whose elements correspond to features of the network traffic, including origin IP, destination IP, and traffic volume. The authors build a weighted similarity graph for the log entries using a simple distance metric for two entries given by the sum of the differences of the respective elements. The force-directed drawing algorithm of [Cha96] is used to compute a 2D drawing of the similarity graph of the entries, which shows clusters of similar entries (see Figure 21.7). For example, this visualization allows to focus on entries associated with small clusters, which denote unusual events that could be associated with anomalous behavior of the network or a security breach.

The work by [MMB05] considers network scans, often used as the preliminary phase of an attack. The authors develop a visualization system that shows the relationships between different network scans (see Figure 21.8). The authors set up a graph where each node

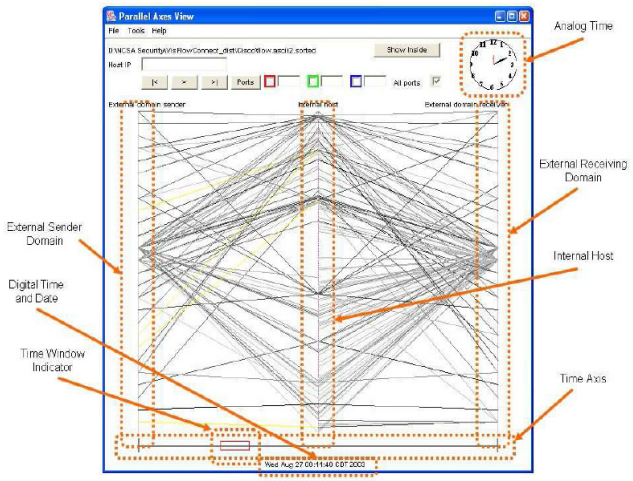


Figure 21.5 Global view of network flows using a tripartite graph layout: nodes represent external domains (on the left and right) and internal domains (in the middle) and edges represent network flows (packet streams) between domains (thumbnail of image from [YYT+04]).

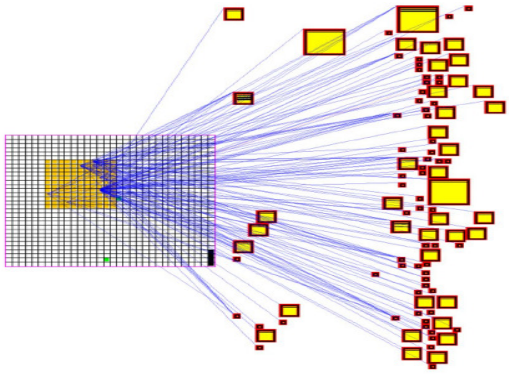


Figure 21.6 Visualization of internal vs. external hosts using a matrix combined with a straight-line drawing. Internal hosts correspond to entries of the matrix while external hosts are drawn as squares placed around the matrix. The size of the square for an external host is proportional to the amount of traffic from/to that host (thumbnail of image from [BFN04]).

represents a scan and the connection between them is weighted according to some metric (similarity measure) that is defined for the two scans. Features taken into consideration for the definition of the similarity measure include the origin IP, the destination IP, and the time of the connection. To avoid displaying a complete graph, the authors define a minimum weight threshold, below which edges are removed. The LinLog force-directed layout method [Noa04] is used for the visualization of this graph. In the drawing produced, sets of similar scans are grouped together, thus facilitating the visual identification of malicious scans.

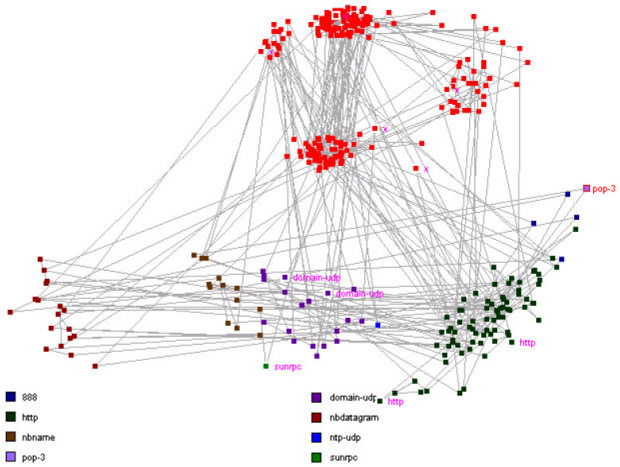


Figure 21.7 Similarity graph of traffic log entries (thumbnail of image from [GB98]).

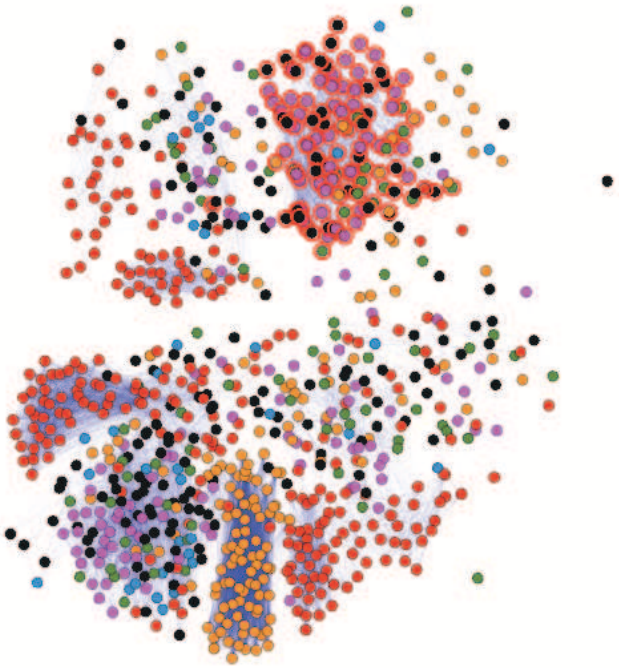


Figure 21.8 Similarity graph of network scans (thumbnail of image from [MMB05]). Nodes represents scans. Only edges with weight (similarity) above a certain threshold are displayed.

21.2.5 Visualization of Address Space

The shift in the address space from IPv4 to IPv6 requires new visualization tools for viewing network activity [BvO09]. The authors of [BvO09] observe that interesting information can be drawn from patterns in the number of IPv6 packets that go between same source and destination addresses. Since the size of each address is 128 bits, it is not trivial to visualize this information. However, IPv6 addresses are allocated in a standardized hierarchical manner in order to keep global routing tables efficient. The work of [BvO09] exploits this assignment pattern to visualize packet information from 4.5 hours of network traffic using treemaps.

In Figure 21.9, the destination addresses of IPv6 packets are displayed. Each rectangle is split into levels that represent the hierarchical nature of the addresses. The size of every rectangle is determined by the number of packets routed to this address. The protocols of the packets are distinguished with colors, e.g., dark gray represents TCP. This visualization of the address space can be used to find frequent destinations, sources that have similar destination behavior, as well as patterns in the type of traffic routed.

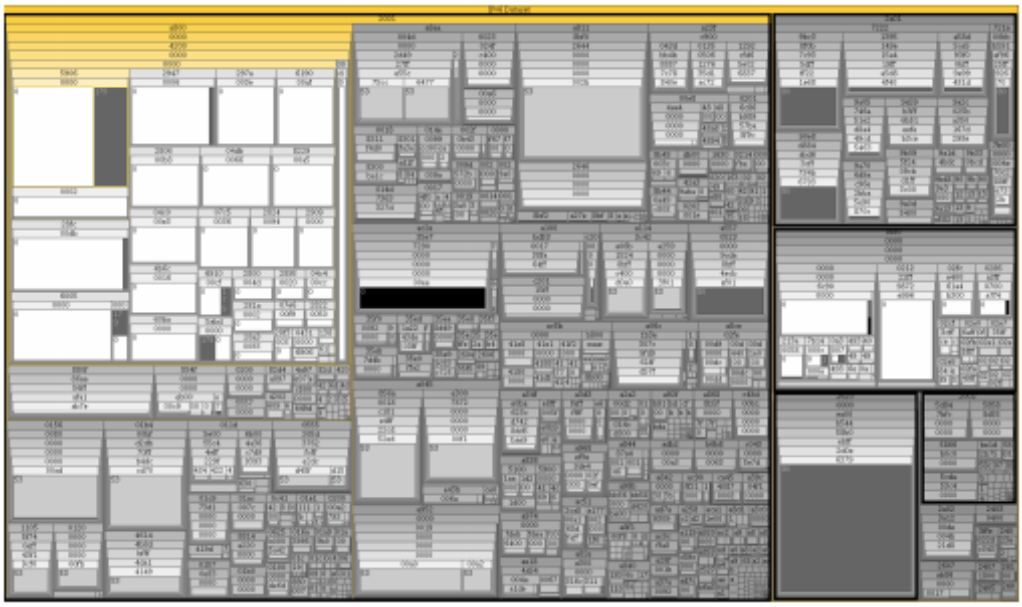


Figure 21.9 A treemap visualization of IPv6 source addresses, destination ports, and packet count of network traces. The colors are used to display the protocol of each packet (thumbnail of image from [BvO09]).

21.2.6 Visualization of Name Server Migration

Recall that a name server finds the IP address of a domain name queried by a computer. To find this IP address, a name server queries other name servers, including root servers. A root

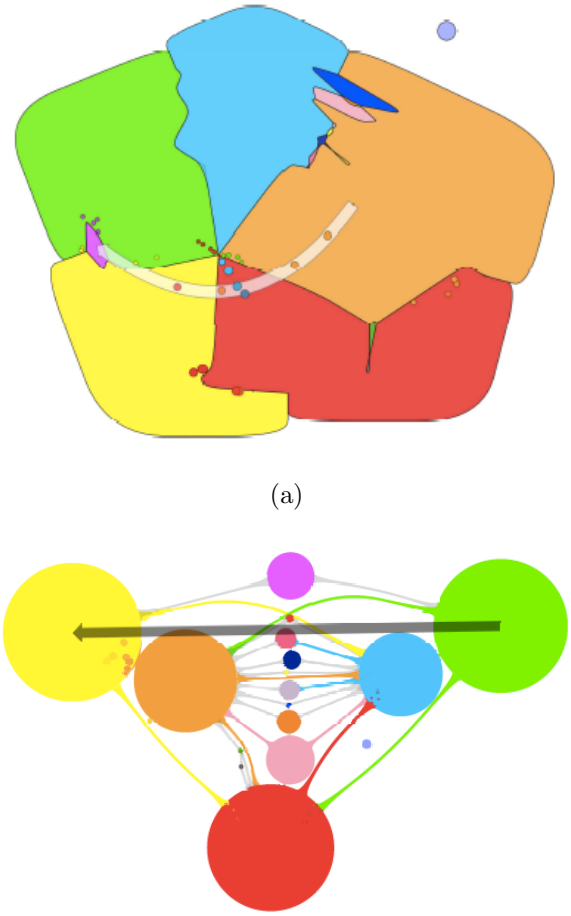
server is responsible for the root zone of the domain name space. Specifically, root servers keep a database of the authoritative name servers for the top-level domains (e.g., .com, .edu, .net, .org, etc.). Since the number of root servers is small and their role is very important in resolving domain names, each of them is implemented via a number of computers, called instances, that provide efficiency and resilience for that root server. Hence, when a name server is trying to answer a query, its request is sent to one of the instances depending on the status of the routing. Migration happens when the same name server is served by different instances of the same root server across time.

The authors of [DSN12] describe an animated visualization of migration of name servers between instances of a single root server, the K-root server. We refer to name servers that query the K-root server as clients. The migration process is visualized between instances by measuring the number of clients served and the total number of queries received by each instance. This information allows to monitor changes in migration patterns and in the workload of each instance over time. Moreover, it also helps to observe any anomalies in changes. For example, one instance is suddenly flooded by requests or clients of a specific Internet Service Provider change root servers in a suspicious pattern.

Two interesting animated visualization techniques are proposed to observe migration between the instances. Both visualizations are based on a *migration graph* G where each node is an instance of the root server and two nodes are connected if migration between the two is considered to be usual. The first visualization technique, *country map*, uses a geography-based layout to show the migration of clients (see Figure 21.10(a)). Each instance is represented as a bounded region of a distinct color, and its size is proportional to the number of clients that it currently serves. If two instances are exchanging clients, then they are adjacent on the map. Unusual migration of clients is represented via a flow traversed by bubbles with size proportional to the amount of flow. The second visualization, *octopus map*, represents instances as circles connected by "tentacles" to show the flow of usual migration (see Figure 21.10 (b)). The width of a tentacle is proportional to the amount of flow between the two instances it connects, while the color is related to the colors of the corresponding instances. Unusual changes are represented by arrows connecting non-adjacent instances.

Drawing of the first visualization consists of constructing a planar graph from the migration graph G (if G is not planar already) as a backbone for the final graph. A straight-line drawing of the backbone preserving its planar topology is drawn in such a way that each vertex has enough area around it to fit the average number of the clients that it serves in a given time period. This is achieved by using a spring embedder algorithm [DETT99] where the charge and the lengths depend on the number of clients an instance and its adjacent instances serve. For smoother visual transition between time intervals in the animation, the drawing of the backbone is modified to maximize the angles between adjacent edges. This step is done by adding new edges to the drawing and using constrained Delaunay triangulation. The skeleton obtained from this step is then adjusted for each specific time interval during animation.

Octopus map drawing involves computing a topology for the migration graph G such that the number of crossings between its edges is minimized. Then a straight-line drawing for G respecting the computed topology is built. During the animation steps, the vertices and edges of G are substituted by circles and tentacles, respectively, and any intersections between the shapes are removed. The challenge is then to scale the drawing to fit into an area where the animation is projected. For this purpose, a constrained spring embedder [DLR11] is used to preserve the original planar topology and ensure that no intersections between the shapes appear.



(a)

(b)

Figure 21.10 Snapshots of animations of client migration between instances of the K-root server. (a) A country map drawing where instances are represented as shapes of distinct colors and borders signify usual exchanges of clients between the instances. Unusual migration is displayed via a flow traversed by bubbles of size proportional to the number of clients migrated. (b) An octopus map drawing of the same data as in the country map drawing above. Each circle is an instance of the K-root server, a tentacle represents an expected migration and its width is proportional to the number of clients exchanged, while a gray arrow shows an unusual migration (thumbnails of images from [DSN12]).

21.3 Border Gateway Protocol

The *Border Gateway Protocol* (*BGP*) manages the routing of IP packets across different Autonomous Systems (AS), which can be informally viewed as collections of hosts under the same administrative control. In this section, we survey selected visualization methods for the Border Gateway Protocol that can be used to discover attacks, anomalies, and faults in the routing network.

21.3.1 Topology of Autonomous Systems

VAST (Visualizing Autonomous System Topology) [OKB06] is a tool that uses 3D straight-line drawings to display the BGP interconnection topology of Autonomous Systems (see Figure 21.11). The goal of the tool is to allow security researchers to quickly extract relevant information from raw routing datasets.

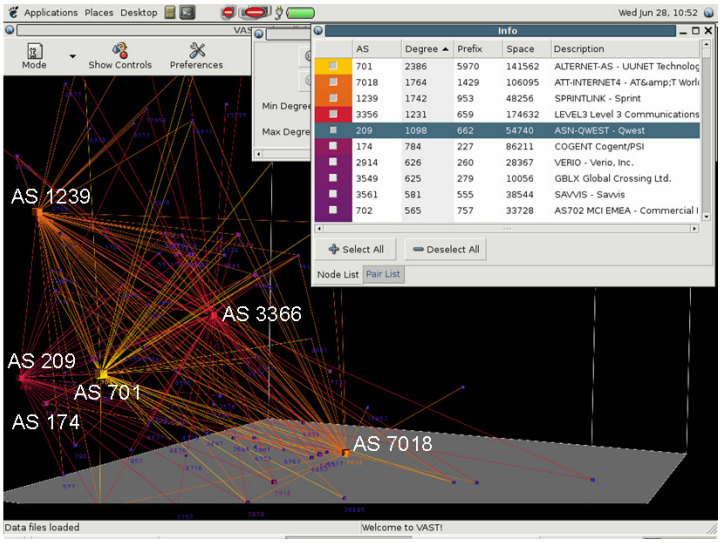


Figure 21.11 Some large autonomous systems in the Internet visualized with VAST (thumbnail of image from [OKB06]).

VAST employs a quad-tree to show information about an Autonomous System and an octo-tree to represent relationships between multiple Autonomous Systems. The visualization allows users to efficiently detect routing malfunctions and sensitive points, including the following ones:

- address-space hijacking attacks, where an Autonomous System maliciously sends routing announcements crafted to attract to itself traffic destined to IP addresses that belong to a different Autonomous System, e.g., to create alternative versions of popular websites;
- anomalous routing announcements, which accidentally cause portions of the Internet to become temporarily unreachable; and
- critical portions of the Internet topology, which are essential for its reliable operation.

The authors have also developed another tool, called *Flamingo*, that uses the same graphical engine as VAST but is used for real-time visualization of network traffic.

21.3.2 BGP Monitoring

BGP Eye [TRNC06] visualizes in real time the status of BGP activity with easy-to-read layouts (see Figure 21.12) and supports root-cause analysis of BGP anomalies. Its main objective is to track the healthiness of BGP activity, raise an alert when an anomaly is detected, and indicate its most likely cause. In particular, the authors show how BGP Eye can be used to analyze two Internet anomalies. First, they use the tool to study a worm outbreak, detecting the ASes that contributed the most to the spread of the infection. In the second use case, BGP Eye visualizes how prefix hijacking affects various ASes and routing tables over time.

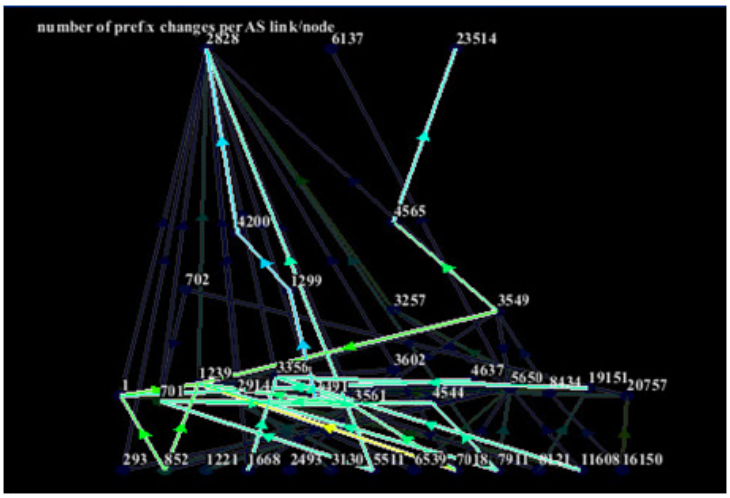
BGP Eye provides two different types of visualization of BGP dynamics:

- The *Internet-centric view* uses layered straight-line drawings to display the interaction between Autonomous Systems (AS) in terms of BGP announcements exchanged. The colors indicate the deviation of current values in the system from the historic ones, allowing to notice any anomalies. Displaying the global view also helps to track the propagation of a problem through the entire Internet, e.g., its growing rate and spreading.
- The *Home-centric view*, which uses a radial drawing, is designed to present BGP activity from the perspective of a specific Autonomous System. In this visualization, the granularity is increased to the router level. The inner ring contains the routers of an Autonomous System and the outer ring contains their peer routers, belonging to other Autonomous Systems. In the outer ring, the layout method groups together routers belonging to the same Autonomous System and uses a placement algorithm that reduces the distance between connected nodes.
  In Figure 21.12 (b) the size of each AS represents the moving average of the number of BGP events originated by the AS. The thickness of AS-AS links represents the number of BGP events traversing this link. The color of lines and nodes gives information on the deviation of the current sample from its historical trend. The minimum deviation value is shown with a blue color while red is the maximum deviation.

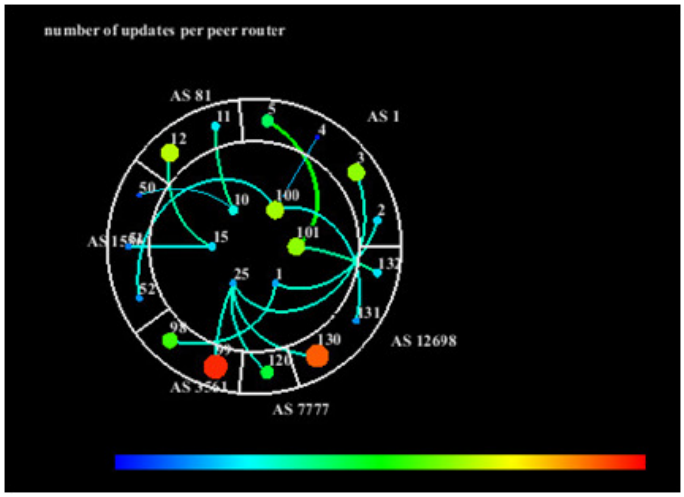
21.3.3 BGP Evolution

BGPlay [BMPP04] and *iBGPlay* provide animated graphs of the BGP routing announcements for a certain IP prefix within a specified time interval (see Figure 21.13). Both visualization tools are targeted to Internet Service Providers. Each node represents an Autonomous System, and paths are used to indicate the sequence of Autonomous Systems needed to be traversed to reach a given destination according to a given announcement. The resulting graphs can be used to discover faults in the links traversed by the traffic flows and to check the consistency of router configurations.

BGPlay shows the paths to the chosen destination (prefix) that appear in announcements collected by observation points spread over the Internet. iBGPlay shows data privately collected by one ISP. The ISP can obtain from iBGPlay visualizations of outgoing paths from itself to any destination. The drawing algorithm is a modification of the force-directed approach that aims at optimizing the layout of the paths.



(a)



(b)

Figure 21.12 Visualizations in BGP Eye: (a) Internet-centric view; and (b) Home-centric view (thumbnails of images from [TRNC06]).

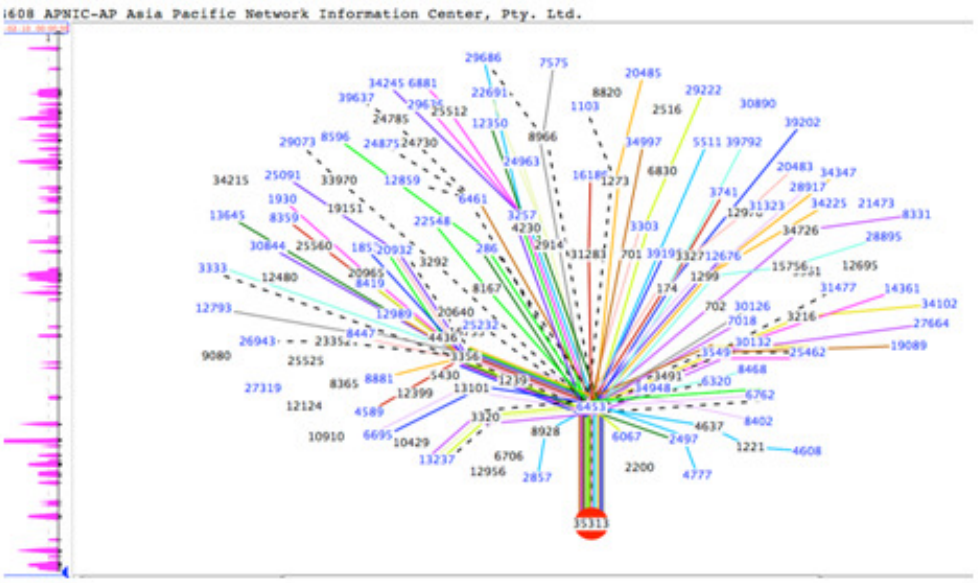


Figure 21.13 In BGPlay, nodes represent Autonomous Systems, and paths are sequences of Autonomous Systems to be traversed to reach the destination (thumbnail of image from [BMPP04]).

21.4 Access Control

This section considers selected graph-based visualization techniques for several aspects of access control.

21.4.1 Rule-Based Access Control

The RubaViz system [MFG+06] is a graphical tool for managing and querying rule-based access control systems (see Figure 21.14). RubaViz makes it easy to answer questions like

> "What group has access to which files during a given time span?"

The system constructs a graph whose nodes are subjects (people or processes), groups, resources, and rules. Directed edges go from subjects/groups to rules and from rules to resources to display allowed accesses. The layout is straight-line and upward.

21.4.2 File System Access-Control

TrACE [HPPT08] is a tool for visualizing file permissions in the NTFS file system (Figure 21.15). TrACE allows a user or administrator to gain a global view of the permissions in a file system, thus simplifying the detection and repair of incorrect configurations leading to unauthorized accesses.

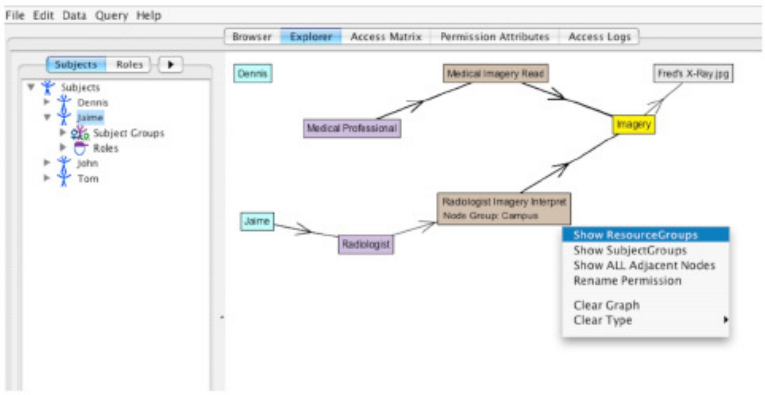


Figure 21.14 The RubaViz system for rule-based access control (thumbnail of image from [MFG+06]).

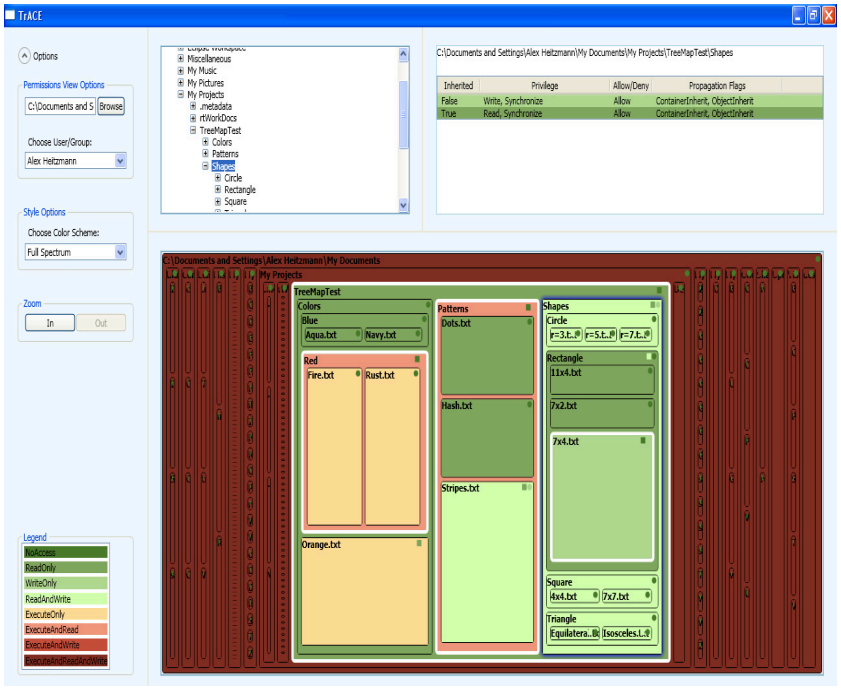


Figure 21.15 Visualization of permissions in the NTFS file system with TrACE (thumbnail of image from [HPPT08]).

In the NTFS file system, there are three types of permissions:

- *explicit* permissions are set by each user or members of a group;
- *inherited* permissions are dynamically inherited from the explicit permissions of the ancestor folders; and
- *effective* permissions are obtained by combining the explicit and inherited permissions.

TrACE uses a treemap layout [JS91] to draw the file system tree and colors the tiles with a palette denoting various access levels. The size of a tile indicates how much the permissions of a folder/file differ from those of its parent and children. Advanced properties, such as a break of inheritance at some folder, are also graphically displayed. The tool makes it easy to figure out explicit and inherited permissions of which nodes affect the effective permissions of a given node in the file system tree.

21.4.3 Trust Negotiation

Using a Web service requires an initial setup phase where the client and server enter into a negotiation to determine the service parameters and the cost by exchanging credentials and policies. Trust negotiation is a protocol that protects the privacy of the client and server by enabling the incremental disclosure of credentials and policies. Planning and executing an effective trust negotiation strategy can be greatly aided by tools that explore alternative scenarios and show the consequences of possible moves.

In [YSTW05], the authors use a layered upward drawing to visualize automated trust negotiation (ATN) (Figure 21.16). In a typical ATN session, the client and the server engage in a protocol that results in the collaborative and incremental construction of a directed acyclic graph, called *trust-target graph*. This graph represents credentials (e.g., a proof that a party has a certain role in an organization) and policies indicating that the disclosure of a credential by one party is subject to the prior disclosure of a set of credentials by the other party [WL02]. A tool based on the Grappa system [BML97], a Java port of Graphviz [EGK+04], is used to generate successive drawings of the trust-target graph being constructed in an ATN session.

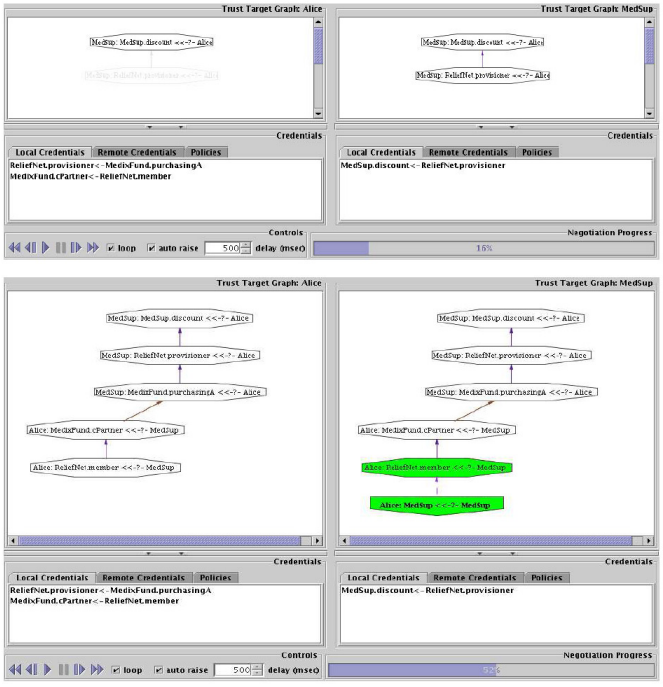
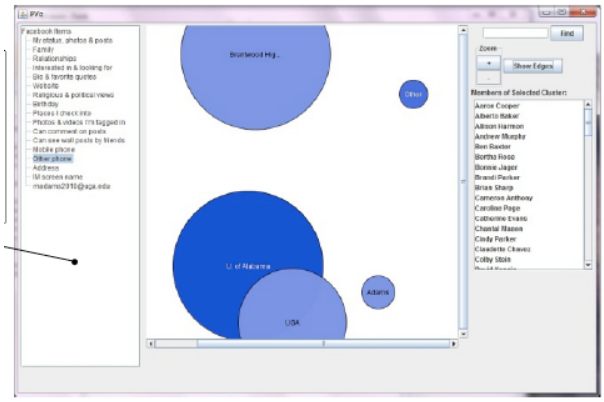


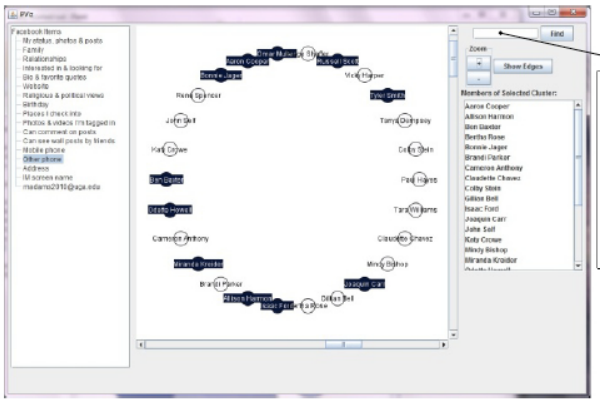
Figure 21.16 Drawing of the trust-target graph generated by a trust negotiation session (thumbnail of image from [YSTW05]).

21.4.4 Privacy Settings in Social Networks

User privacy in social networks is of concern to the users and companies providing this service. To this end, social networking companies are creating more and more tools to help users manage their privacy settings. The authors of [MLA12] describe a visual tool for users of social networks to assess the visibility of their data among their friends in a more accessible way. The tool parses the data of the user and creates a graph of groups and subgroups from a list of friends of the user, where friends are split into groups using modularity optimization. Hence, a node in the final graph represents a group of friends. The user can then query this graph via zooming or direct queries to see the visibility of his data among his friends. Authors chose a force-directed approach to display nodes in the graph, and the color of the nodes represents the privacy level (see Figure 21.17).



(a)



(b)

Figure 21.17 Visualization of privacy settings in a social network. (a) Circles represent groups of friends that have the same visibility of user's data and the color shows the privacy level. (b) The granularity is increased to the level of specific information of a user, e.g., a phone number, and its visibility among the user's friends (thumbnails of images from [MLA12]).

21.5 Attack Graphs

21.5.1 Model

Given a network and a database of known vulnerabilities that apply to certain machines of the network, one can construct a directed graph where each node is a machine (or group of machines) and an edge denotes how a successful attack on the source machine allows to exploit a vulnerability on the destination machine. This graph, called attack graph, can be rather large and complex. Thus, it is essential to use automated tools to analyze attack graphs.

21.5.2 Tools

A tool for visualizing attack graphs is described in [NJKJ05] (Figure 21.18). The system clusters machines in order to reduce the complexity of the attack graph (e.g., machines that belong to the same subnet may be susceptible to the same attack). The Graphviz tool [EGK+04] is used to produce a layered drawing of the clustered attack graph. Similar layered drawings for attack graphs are proposed in [NJ04].

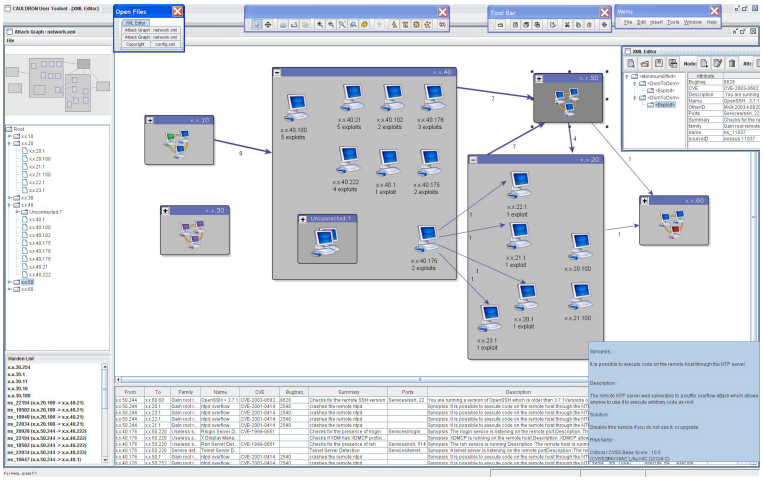
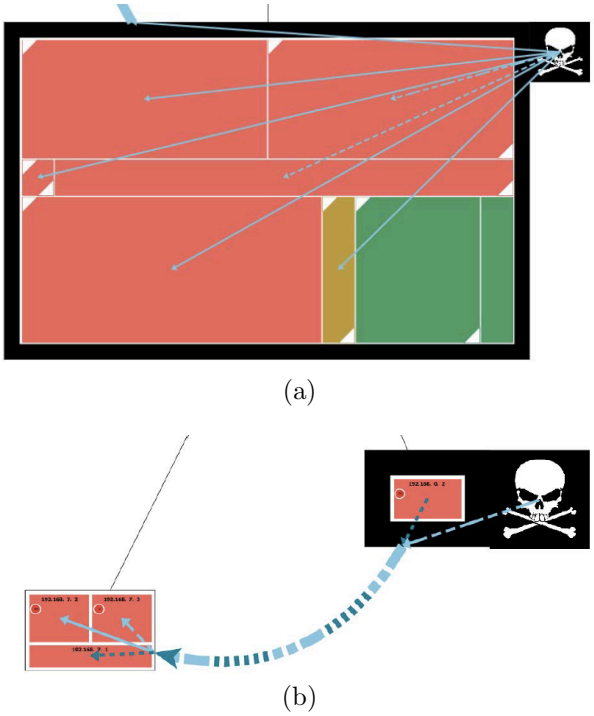


Figure 21.18 Visualization of an attack graph (thumbnail of image from [NJKJ05]).

The authors of [CIL+10] describe Navigator, another tool for visualizing attack graphs for displaying server-side, client-side, credential-based, and trust-based attacks in the network. Navigator groups machines from the same subnet based on similar vulnerabilities but also gives an "asset value" to each host that represents the importance level of this host in the network. To display this information the authors use a modification of the strip treemap algorithm by [BSW02] (see Figure 21.19). Navigator can display different types of attacks that can be brought from one entity of the network to another. In Figure 21.19, rectangles represent host groups of the same subnet and arrows show the steps that the attacker could take to progress through the network. The background colors of the entities represent the compromise levels achievable in the displayed attack scenarios. For example, red means that the root is compromised while green stands for no compromise. The color of the arrows shows the depth of the attack. The visualization tool can also show how the attack from one subnet affects the rest of the network.



(a)

(b)

Figure 21.19 Visualization of (a) an attack graph with importance levels of the hosts, (b) multiple attacks between subnets, displayed as a hybrid edge of multiple colors (thumbnails of images from [CIL+10]).

Given that there are multiple attacks and that the type and the depth of these attacks can vary, the authors propose a way to aggregate this information into hybrid edges to avoid clutter (see Figure 21.19(b)). For example, the solid part of an edge shows server-side attacks, while client-side attacks are displayed as dashed segments.

The authors of [CIL+10] have modified the strip treemap algorithm to achieve visualizations that are aesthetically pleasing and easier to navigate. Their algorithm sets minimum width and height for the hosts within a subnet to avoid very long and thin or very short and wide host representations. Any extra space that is accumulated due to the minimum rectangle requirements is then propagated and scaled to top layers. The modification also preserves the order of the hosts when the user zooms in a host group.

21.6 Private Graph Drawing

In previous sections, graph drawing was used to visualize the data in a way that can help to observe any anomalies in the network, privacy settings, or access control to a filesystem. The authors of [GOT12] raise instead a privacy concern related to the process of drawing a graph. With the recent shift of data storage to a cloud-based storage, one can no longer assume that the input data for a graph drawing algorithm is stored locally. Hence, to draw a graph, one accesses the data remotely. This raises concerns about the privacy of the outsourced data. In [GOT12], a new model for graph drawing algorithms is proposed that fits the cloud computing paradigm and preserves data privacy and its access pattern.

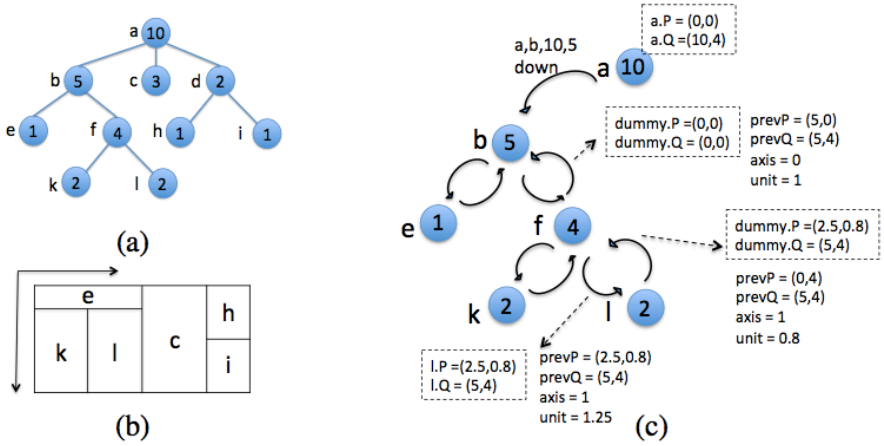


Figure 21.20 (a) Input graph and (b) its treemap drawing. (c) Execution of a privacy-preserving treemap drawing algorithm in the cloud storage model. The algorithm performs the computation required for the drawing during an Euler tour of the graph. The edge representation of an Euler tour is stored encrypted remotely. Each edge contains information about its adjacent nodes, its direction in the Euler tour of the tree (up or down); and a link to the next edge in the tour (e.g., edge a-b). During the traversal, the algorithm maintains several variables: unit, the unit length of current rectangle; prevP and prevQ, location of a previously drawn rectangle; and axis, the direction of the drawing which takes a value of x or y. When the traversal is going down the tree, each node on the way is assigned x and y coordinates of its corresponding rectangle, variables P and Q. After reaching a leaf node the algorithm follows the Euler tour up the tree. The algorithm cannot reveal when it sees the nodes it has already assigned since it would expose the height and the width of the tree. Hence, it writes dummy coordinates for all nodes it encounters when going up the tree, dummy.P and dummy.Q in the figure. (thumbnails of images from [GOT12]).

21.6.1 Compressed Scanning

In the *compressed-scanning* model, data is stored encrypted on the cloud and is permuted using a pseudo-random permutation. A graph drawing algorithm is then split into rounds. During each round, the algorithm scans remotely-stored data related to the graph. For example, a data item can be a node or an edge of the graph. Once the data item is retrieved, a small computation, which could potentially modify the item, is performed in local private memory. The item is then re-encrypted and written back. A small private memory is used to store information during each round. After each round, the data is re-permuted according to a new permutation.

The number of rounds is specific to the graph drawing algorithm and does not depend on the data. The privacy of the model comes from using data encryption and accessing data in a nonrevealing manner. Note that during each round every data item is read only once, and a new permutation is created so that nobody observing the access pattern of the algorithm can deduce information about the graph, including its layout and depth. Hence, the only information that is leaked from running a graph drawing algorithm is the size of the graph, but not its topology. An algorithm satisfying this privacy property is said to be *data-oblivious*.

21.6.2 Data-Oblivious Drawing Algorithms

The authors show how to modify four classical graph drawing algorithms to fit the compressed-scanning model, including symmetric straight-line drawings and treemaps [JS91], drawings of trees, dominance drawings of planar acyclic digraphs [DTT92], and Δ-drawings of series-parallel graphs [BCD$^+$94]. These algorithms work over trees or construct a tree representation, e.g., spanning trees for acyclic digraphs.

The main technique is based on representing the graph via an Euler tour of the tree, hence every edge is stored twice and traversal of the tree involves accessing each edge only once. If the number of nodes in the graph is n, then three algorithms in [GOT12] use only a constant amount of private storage during each round. While drawing a tree with bounding rectangles uses $\log n$ private memory. For an illustration of the technique applied to treemap drawings, see the examples in Figure 21.20.

The authors show that graph drawing methods that fit the compressed-scanning model require $T(A)\mathsf{sort}(n)$ accesses to remote data storage. $T(A)$ is the number of rounds specific to the graph drawing algorithm A and does not depend on n. The factor $\mathsf{sort}(n)$ is the number of rounds that is required to securely order the data according to a new permutation after each round.

The work of [GS12] expands this line of work and shows how to simulate parallel algorithms data-obliviously using the compresses-scanning model. Given a CRCW PRAM algorithm B that runs in $T(B)$ steps using a memory of size M and $P \leq M$ processors, the authors show how to simulate B sequentially in a data-oblivious fashion in $O(T(B)M \log M)$ accesses to remote data storage. This result can be used to obliviously compute st-numberings, visibility representations, upward grid drawings, and orthogonal grid drawings of planar graphs that are stored remotely.

Acknowledgments

This chapter is an extended version of a conference paper on graph drawing methods for security visualization [TPP09]. Charalampos Papamanthou contributed to this chapter while he was at Brown University. This work was supported in part by the U.S. National Science Foundation under grant OCI–0724806. We are indebted to Massimo Rimondini for detailed comments on an earlier version of this chapter. We also thank Giuseppe Di Battista, Michael Goodrich, and Ioannis Tollis for useful suggestions.

References

[BCD+94] P. Bertolazzi, R. F. Cohen, G. Di Battista, R. Tamassia, and I. G. Tollis. How to draw a series-parallel digraph. *International Journal of Computational Geometry and Applications*, 4:385–402, 1994.

[BFN04] R. Ball, G. A. Fink, and C. North. Home-centric visualization of network traffic for security administration. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, VizSEC/DMSEC '04, pages 55–64, New York, NY, USA, 2004. ACM.

[BML97] N. Barghouti, J. Mocenigo, and W. Lee. Grappa: A GRAPh Package in JAVA. In Giuseppe Di Battista, editor, *Graph Drawing*, volume 1353 of *Lecture Notes in Computer Science*, pages 336–343. Springer Berlin Heidelberg, 1997.

[BMPP04] G. Battista, F. Mariani, M. Patrignani, and M. Pizzonia. iBGPlay: A system for visualizing the interdomain routing evolution. In Giuseppe Liotta, editor, *Graph Drawing*, volume 2912 of *Lecture Notes in Computer Science*, pages 295–306. Springer Berlin Heidelberg, 2004.

[BSW02] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *ACM Transactions on Graphics*, 21(4):833–854, 2002.

[BvO09] D. Barrera and P. C. van Oorschot. Security visualization tools and IPv6 addresses. In *Proceedings of the 6th International Workshop on Visualization for Computer Security*, VizSec '09, pages 21–26, 2009.

[Cha96] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *Proceedings of the 7th conference on Visualization*, VIS '96, pages 127–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[CIL+10] M. Chu, K. Ingols, R. Lippmann, S. Webster, and S. Boyer. Visualizing attack graphs, reachability, and trust relationships with NAVIGATOR. In *Proceedings of the 7th International Symposium on Visualization for Cyber Security*, VizSec '10, pages 22–33, New York, NY, USA, 2010. ACM.

[Con07] G. Conti. *Security Data Visualization.* No Starch Press, San Francisco, CA, USA, 2007.

[DETT99] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing.* Prentice Hall, Upper Saddle River, NJ, 1999.

[DLR11] W. Didimo, G. Liotta, and S. A. Romeo. Topology-driven force-directed algorithms. In Ulrik Brandes and Sabine Cornelsen, editors, *Graph Drawing*, volume 6502 of *Lecture Notes in Computer Science*, pages 165–176. Springer Berlin Heidelberg, 2011.

[DSN12] G. Di Battista, C. Squarcella, and W. Nagele. How to visualize the K-Root name server. *Journal of Graph Algorithms and Applications*, 2012. In print.

[DTT92] G. Di Battista, R. Tamassia, and I. G. Tollis. Area requirement and symmetry display of planar upward drawings. *Discrete & Computational Geometry*, 7:381–401, 1992.

[Ead84] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[EGK+04] J. Ellson, E.R. Gansner, E. Koutsofios, S.C. North, and G. Woodhull. Graphviz and dynagraph – static and dynamic graph drawing tools. In Michael Jünger and Petra Mutzel, editors, *Graph Drawing Software*, Mathematics and Visualization, pages 127–148. Springer Berlin Heidelberg, 2004.

[FMK+08] F. Fischer, F. Mansmann, D. A. Keim, S. Pietzko, and M. Waldvogel. Large-scale network monitoring for visual analysis of attacks. In *Proceedings of the 5th International Workshop on Visualization for Computer Security*, VizSec, pages 111–118, Berlin, Heidelberg, 2008. Springer-Verlag.

[FR91] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, November 1991.

[GB98] L. Girardin and D. Brodbeck. A visual approach for monitoring logs. In *Proceedings of the 12th Conference on Systems Administration*, LISA '98, pages 299–308, Berkeley, CA, USA, 1998. USENIX Association.

[GOT12] M. Goodrich, O. Ohrimenko, and R. Tamassia. Graph drawing in the cloud: Privately visualizing relational data using small working storage. In *Graph Drawing*, 2012. To appear.

[GS12] M. Goodrich and J. Simons. More graph drawing in the cloud: Data-oblivious st-numbering, visibility representations, and orthogonal drawing of biconnected planar graphs. In *Graph Drawing*, 2012. Poster.

[GT11] M. Goodrich and R. Tamassia. *Introduction to Computer Security*. Addison-Wesley, 2011.

[HPPT08] A. Heitzmann, B. Palazzi, C. Papamanthou, and R. Tamassia. Effective visualization of file system access-control. In *Proceedings of the 5th International Workshop on Visualization for Computer Security*, VizSec '08, pages 18–25, Berlin, Heidelberg, 2008. Springer-Verlag.

[JS91] B. Johnson and B. Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *IEEE Visualization*, pages 284–291, 1991.

[Mar08] R. Marty. *Applied Security Visualization*. Addison-Wesley, 2008.

[MFG+06] J. Montemayor, A. Freeman, J. Gersh, T. Llanso, and D. Patrone. Information visualization for rule-based resource access control. In *Proceedings of the 2nd Symposium on Usable Privacy and Security*, SOUPS '06, 2006.

[MFK+09] Florian Mansmann, Fabian Fischer, Daniel A. Keim, Stephan Pietzko, and Marcel Waldvogel. Interactive analysis of netflows for misuse detection in large IP networks. In *DFN-Forum Kommunikationstechnologien*, pages 115–124, 2009.

[MLA12] A. Mazzia, K. LeFevre, and E. Adar. The PViz comprehension tool for social network privacy settings. In *Proceedings of the 8th Symposium on Usable Privacy and Security*, SOUPS '12, pages 13:1–13:12, New York, NY, USA, 2012. ACM.

[MMB05] C. Muelder, K. Ma, and T. Bartoletti. A visualization methodology for characterization of network scans. In *Proceedings of the IEEE Workshops on Visualization for Computer Security*, VIZSEC '05, pages 4–, Washington, DC, USA, 2005. IEEE Computer Society.

[MMK07] F. Mansmann, L. Meier, and D. Keim. Graph-based monitoring of host behavior for network security. In *Proceedings of the 4th International Workshop on Visualization for Computer Security*, VizSec '07, 2007.

[NJ04] S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, VizSEC/DMSEC '04, pages 109–118, New York, NY, USA, 2004. ACM.

[NJKJ05] S. Noel, M. Jacobs, P. Kalapa, and S. Jajodia. Multiple coordinated views for network attack graphs. In *Proceedings of the IEEE Workshops on Visualization for Computer Security*, VIZSEC '05, pages 12–, Washington, DC, USA, 2005. IEEE Computer Society.

[Noa04] A. Noack. An energy model for visual graph clustering. In Giuseppe Liotta, editor, *Graph Drawing*, volume 2912 of *Lecture Notes in Computer Science*, pages 425–436. Springer Berlin Heidelberg, 2004.

[OKB06] J. Oberheide, M. Karir, and D. Blazakis. VAST: visualizing autonomous system topology. In *Proceedings of the 3rd International Workshop on Visualization for Computer Security*, VizSec '06, pages 71–80, New York, NY, USA, 2006. ACM.

[TN00] J. Tölle and O. Niggemann. Supporting intrusion detection by graph clustering and graph drawing. In *Proceedings of 3rd International Workshop on Recent Advances in Intrusion Detection*, RAID '00, Toulouse, France, 2000.

[Tol] J. Toledo. EtherApe a live graphical network monitor tool. http://etherape.sourceforge.net/.

[TPP09] R. Tamassia, B. Palazzi, and C. Papamanthou. Graph drawing for security visualization. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing*, volume 5417 of *Lecture Notes in Computer Science*, pages 2–13. Springer Berlin Heidelberg, 2009.

[TRNC06] S. T. Teoh, S. Ranjan, A. Nucci, and C. Chuah. BGP eye: a new visualization tool for real-time detection and analysis of BGP anomalies. In *Proceedings of the 3rd International Workshop on Visualization for Computer Security*, VizSec '06, pages 81–90, New York, NY, USA, 2006. ACM.

[WL02] W. H. Winsborough and N. Li. Towards practical automated trust negotiation. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks*, POLICY '02, pages 92–103. IEEE Computer Society Press, June 2002.

[XMB+06] I. Xydas, G. Miaoulis, P.-F. Bonnefoi, D. Plemenos, and D. Ghazanfarpour. 3D graph visualization prototype system for intrusion detection: A surveillance aid to security analysts. In *Proceedings of the 9th International Conference on Computer Graphics and Artificial Intelligence*, 3IA, pages 153–165, 2006.

[Yee06] G. Yee. Visualization for privacy compliance. In *Proceedings of the 3rd International Workshop on Visualization for Computer Security*, VizSec '06, pages 117–122, New York, NY, USA, 2006. ACM.

[YSTW05] D. Yao, M. Shin, R. Tamassia, and W. H. Winsborough. Visualization of automated trust negotiation. In *Proceedings of the IEEE Workshops on Visualization for Computer Security*, VIZSEC '05, pages 8–, Washington, DC, USA, 2005. IEEE Computer Society.

[YYT+04] X. Yin, W. Yurcik, M. Treaster, Y. Li, and K. Lakkaraju. VisFlowConnect: netflow visualizations of link relationships for security situational awareness. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, VizSEC/DMSEC '04, pages 26–34, New York, NY, USA, 2004. ACM.