IPSJ SIG Technical Report Vol.2018-IOT-42 No.7
2018/6/28

Vulnerability Is the Plan the Plan Is Death

Hiroki Kashiwazaki[1,a)]

Abstract: An author is a spider-like creature who has decided that, since he is intelligent, he will resist the instincts which lead his species through an exceptionally violent life-cycle which they call "the Plan" – including a drop in intelligence during "winter". Last December, a certain university announced large-scale personal information leak caused on several systems by several unauthorized accesses. Security advisory consulting companies order the university to "strengthen governance in their own institutes". This paper shows an aftermath of the personal information leak incident in the university and how administration bureau deals with a lot of the vulnerability in the university.

1. Introduction

In order to promote the activities of the organization smoothly, various tasks were computerized (to be informatization). Computerized works are provided by an information service, and the information service means an application operating on a standard OS or server software. There are no vulnerable-free OS, server software, and applications, and as soon as the vulnerability is disclosed, it must respond promptly. However, in the stack structure where the server software is on the OS and the application is on the server software, modification and change of the lower stack may affect the upper stack. In order to keep the SLA above a certain level, it may be necessary to test whether the stacks above the software is affected before updating the software. The time required for this test depends on the size of the stack at the higher level, and the test procedure is not specified, and the increase in the time required for the test shows nonlinearity. Meanwhile, the number of staff who operate and manage the information service is constant (sometimes decreases) unfortunately.

Especially in modern information systems, applications do not simply operate on server software, but applications run using various frameworks that depend on more various libraries, and cooperation with other services such as databases (Fig. 1). In this way, the time required for testing increases exponentially, and eventually system administrators will think like this: "We do not have to deal with vulnerabilities," he says. This is, of course, thought stops and delusions, but on the other hand, there is one truth for this delusion. The truth is that "how quickly we respond to vulnerabilities and countermeasures after they are published is a matter of risk assessment?". If this evaluation cannot be done quantitatively, who on the earth can blame the poor system administrator mentioned above?

Anyway, what is the necessity of vulnerability management? The primary significance is that the defense power against attacks from malicious third parties increases. But that's not all. It is also meant to suppress the speed of spread of infection when invaded inside by some method. It is also significant as circumstantial evidence to deny the doubt of infection spread. There are some cases with no obvious physical evidence that can deny doubts of spreading infection. In the cases, circumstantial evidence is important. In December 2017, a Japanese university announced large-scale personal information leak. According to open information, the leak was caused on several systems by several unauthorized accesses. The university also announced that secondary damage was not confirmed and leaked information did not include patient information of a medical hospital in the university. Also in the incident, circumstantial evidence played an important role to make things happen.

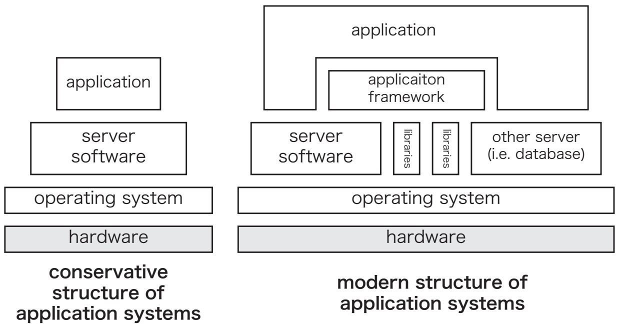

Fig. 1 A comparative diagram of conservative and modern structure of application systems

Meanwhile, there are also problems of structure of the organization. In an organization with a simple tree structure, the headquarter of the organization can issue a command in a top-down manner and subordinate organizations must follow it. However, in an organization with complicated mesh networks like university organizations, it is preferable for the author to promote autonomous behavior by showing rationality, rather than by commands or prerogatives, to govern the organization. In this paper, the author show the improvement of the vulnerability checking mechanism at a certain university and the results thereof.

[1] Cybermedia Center, Osaka University, Ibaraki, Osaka, 567–0047, Japan
[a)] reo@cmc.osaka-u.ac.jp

2. Method

There are various kinds of vulnerability checking software and services. In this section, the author introduces the representative vulnerability checkers.

2.1 Nessus

Nessus[*1] is a proprietary vulnerability scanner developed by Tenable Network Security[*2]. It is free of charge for personal use in a non-enterprise environment[*3]. Features of Nessus are shown below.

- Customize reports to sort by vulnerability or host, create an executive summary or compare scan results to highlight changes.
- Broad asset coverage and profiling (Network devices (Juniper, Check Point, Cisco, Palo Alto Networks), printers, storages, Virtualization environments (VMware ESX, ESXi, vSphere, vCenter, Microsoft, Hyper-V, Citrix Xen Server), various OS (Windows, OS X, Linux, Solaris, FreeBSD, Cisco iOS, IBM iSeries), Databases (Oracle, SQL Server, MySQL, DB2, Informix/DRDA, PostgreSQL, MongoDB), Cloud (Salesforce, Amazon Web Services, Microsoft Azure and Rackspace).
- Detect viruses, malware, backdoors, hosts communicating with botnet-infected systems, known/unknown processes, web services linking to malicious content.
- Compliance auditing: FFIEC, FISMA, CyberScope, GLBA, HIPAA/ HITECH, NERC, SCAP, SOX.
- Configuration auditing: CERT, CIS, COBIT/ITIL, DISA STIGs, FDCC, ISO, NIST, NSA, PCI.

Nessus consists of two main components; nessusd, the Nessus daemon, which does the scanning, and nessus, the client, which controls scans and presents the vulnerability results to the user. Later versions of Nessus (4 and greater) utilize a web server which provides the same functionality as the client. In typical operation, Nessus begins by doing a port scan with one of its four internal portscanners (or it can optionally use AmapM[*4] or Nmap[*5]) to determine which ports are open on the target and then tries various exploits on the open ports. The vulnerability tests, available as subscriptions, are written in NASL (Nessus Attack Scripting Language), a scripting language optimized for custom network interaction.

The Nessus Project was started by Renaud Deraison in 1998 to provide to the Internet community a free remote security scanner[1]. On October 5, 2005, Tenable Network Security, the company Renaud Deraison co-founded, changed Nessus 3 to a proprietary (closed source) license. The earlier versions appear to have been removed from the official website since then. The Nessus 3 engine is still free of charge, though Tenable charges 100 USD/month per scanner for the ability to perform configuration audits for PCI, CIS, FDCC and other configuration standards, technical support, SCADA vulnerability audits, the latest network checks and patch audits, the ability to audit anti-virus configurations and the ability for Nessus to perform sensitive data searches to look for credit card, social security number and many other types of corporate data.

In July 2008, Tenable sent out a revision of the feed license which will allow home users full access to plugin feeds. A professional license is available for commercial use[*6].

2.2 OpenVAS

The Nessus 2 engine and a minority of the plugins are still GPL, leading to forked open source projects based on Nessus like OpenVAS (Open Vulnerability Assessment System, originally known as GNessUs)[*7]. OpenVAS is a software framework of several services and tools offering vulnerability scanning and vulnerability management. The framework is part of Greenbone Networks' commercial vulnerability management solution from which developments are contributed to the Open Source community since 2009. The actual security scanner is accompanied with a regularly updated feed of Network Vulnerability Tests (NVTs), over 50,000 in total. All OpenVAS products are Free Software. Most components are licensed under the GNU General Public License (GNU GPL).

The Open Vulnerability Assessment System (OpenVAS) is a framework of several services and tools. The core of this SSL-secured service-oriented architecture is the OpenVAS Scanner. The scanner very efficiently executes the actual Network Vulnerability Tests (NVTs) which are served via the OpenVAS NVT Feed or via a commercial feed service. The OpenVAS Manager is the central service that consolidates plain vulnerability scanning into a full vulnerability management solution. The Manager controls the Scanner via OTP (OpenVAS Transfer Protocol) and itself offers the XML-based, stateless OpenVAS Management Protocol (OMP). All intelligence is implemented in the Manager so that it is possible to implement various lean clients that will behave consistently e.g. with regard to filtering or sorting scan results. The Manager also controls a SQL database (sqlite-based) where all configuration and scan result data is centrally stored. Finally, Manager also handles user management includiung access control with groups and roles.

Different OMP clients are available: The Greenbone Security Assistant (GSA) is a lean web service offering a user interface for web browsers. GSA uses XSL transformation stylesheet that converts OMP responses into HTML. OpenVAS CLI contains the command line tool “omp” which allows to create batch processes to drive OpenVAS Manager. Another tool of this package is a Nagios plugin. Most of the tools listed above share functionality that is aggregated in the OpenVAS Libraries. The OpenVAS Scanner offers the communication protocol OTP (OpenVAS Transfer Protocol) which allows to control the scan execution. This protocol is subject to be eventually replaced and thus it is not recommended to develop OTP clients[*8].

[*1] `https://www.tenable.com/products/nessus/nessus-professional`
[*2] `https://www.tenable.com/`
[*3] For enterprise use, you should call the reseller such as TOYO Corporation.
[*4] `https://www.amap.no/`
[*5] `https://nmap.org/`
[*6] `https://en.wikipedia.org/wiki/Nessus_(software)`
[*7] `http://www.openvas.org/`
[*8] urlhttp://www.openvas.org/software.html

2.3 Vuls

Vuls is Vulnerability scanner for Linux/FreeBSD, agentless, written in golang[*9]. Developers of Vuls think that system administrators have to perform security vulnerability analysis and software update on a daily basis and the basis can be a burden. To avoid downtime in production environment, it is common for system administrator to choose not to use the automatic update option provided by package manager and to perform update manually. This leads to the following problems.

- System administrator will have to constantly watch out for any new vulnerabilities in NVD(National Vulnerability Database) or similar databases.
- It might be impossible for the system administrator to monitor all the software if there are a large number of software installed in server.
- It is expensive to perform analysis to determine the servers affected by new vulnerabilities. The possibility of overlooking a server or two during analysis is there.

Vuls is a tool created to solve the problems listed above. It has the following characteristics.

- Informs users of the vulnerabilities that are related to the system.
- Informs users of the servers that are affected.
- Vulnerability detection is done automatically to prevent any oversight.
- Report is generated on regular basis using CRON or other methods. to manage vulnerability.

2.4 Tenable.io

Tenable.io is a modern vulnerability management platform[*10]. According to the datasheet, it brings clarity to security of the organization and compliance posture through a fresh, asset-based approach that accurately tracks resources of the organization and vulnerabilities, while accommodating dynamic assets like cloud and containers. it can maximize visibility and insight and effectively prioritizes the vulnerabilities, while seamlessly integrating into environment of the organization. Key benefits of the products are shown below.

- Eliminates blind spots: Tenable.io delivers the most comprehensive visibility into traditional and modern assets, such as cloud, mobile, containers and web applications.
- Focuses effort with vulnerability state tracking: Tenable.io puts the vulnerabilities you care about – those that are new, active or resurfaced – front and center so you can focus your efforts on things that really matter.
- Improves productivity through a streamlined user experience: With a modern, intuitive user interface and guided in-application messaging, Tenable.io effectively leads you through both common and complex tasks.
- Maximizes value via simplified integrations: Tenable.io includes pre-built integrations with complementary systems, like password vault, patch management and Mobile Device Management (MDM) solutions.
- Improves ROI with an elastic asset licensing model: Tenable.io offers a first-to-market assetbased licensing model that consumes just a single license unit per asset.

*9 `https://github.com/future-architect/vuls`
*10 `https://www.tenable.com/products/tenable-io`

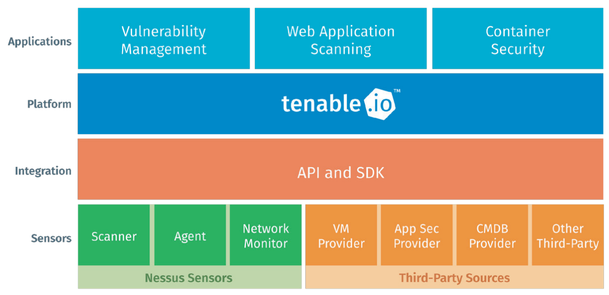

Fig. 2 Diagram of Tenable.io

The platform includes Nessus data sensors for active and agent-based scanning and passive traffic listening, as well as an API and SDK for those who want to automate the sharing of Tenable.io capabilities and vulnerability data, or build on the Tenable.io platform. Built on the Tenable.io platform are a growing number of applications that solve today's toughest security challenges, including vulnerability management, container security and web application scanning – making it easy to start with one application and upgrade to others as requirements grow. This combination of applications, data sensors and automation delivers maximum coverage and provides continuous visibility into assets and vulnerabilities – so you can take better-informed action to protect what matters most (Fig.2).

In order to establish governance in the organization of the mesh structure, the author considered that respective vulnerability checking to each asset is necessary, not uniform vulnerability check. For that purpose, it is insufficient to be able to do with the web user interface for executing the vulnerability check, and that programmable operation is also required. Also, top-down instructions in the organization of the mesh structure, such as "installing agents on the all servers", etc, may create distrust of the organization that ordered it. For this reason, passive vulnerability scanning must be performed at the beginning of the vulnerability check environment construction. The product that meets these requirements was Tenable.io.

3. Design and implemetation

At the early stage after introducing Tenable.io to the university, the operations of scanning vulnerability are executed manually with the web interface. But manual operation occurred some mistakes and its human cost is pretty large. By using Tenable.io API, the author have been making and publishing an object-oriented design of Tenable.io by Ruby[*11] correspond to Tenable.io API design[*12]. According to the class library, a large part of operations was changed to be programmable and programmable procedure could reduce the human cost.

The number of resource types of Tenable.io API is 25. Resource types consist of agent-config, agent-exclusions, agent-groups, agents, assets, audit-log, bulk-operations, editor, exclusions, exports, file, filters, folders, groups, permissions, plugins,

*11 `https://github.com/reokashiwa/tenable`
*12 `https://cloud.tenable.com/api`

policies, scanner-groups, scanners, scans, server, session, target-groups, users, and workbenches. All of the resource types can be separated into two types of API. The first is API with GET method and another is with POST method. API with GET method requires several parameters of HTTP. Both of GET and POST method require `X-ApiKeys` that is combination of `accessKey` and `secretKey`. The parameters can be described in a single hash. The hash is expanded to the sequence of the combination between a key and its value binding with "=". Tenable.io API with GET method can be described as a method that requires two arguments, a path and the hash. A Ruby implementation of a common method with GET method is shown below.

```
class TenableIO
  def initialize(conf)
    accessKey = conf[:accessKey]
    secretKey = conf[:secretKey]
    @x_apikeys = sprintf("accessKey=%s; secretKey=%s",
                         accessKey, secretKey)
    @uri = URI.parse(conf[:uri])
  end

  def get(path, parameter_hash)
    @uri.path = path

    if parameter_hash
      query = String.new
      parameter_hash.each{|key,value|
        query = query + sprintf("%s=%s&", key, value)
      }
      query.gsub!(/&$/,"")
      @uri.query = query
    end

    https = Net::HTTP.new(@uri.host, @uri.port)
    https.use_ssl = true
    response = https.get(@uri.request_uri,
                         {'X-ApiKeys' => @x_apikeys})
  end
```

API with POST method require POST body. The POST body also can be described in a single hash. POST body is sent converted to JSON format[*13]. A Ruby implementation of a common method with POST method is shown below.

```
def post(path, post_body_hash)
  @uri.path = path

  https = Net::HTTP.new(@uri.host, @uri.port)
  https.use_ssl = true
  request = Net::HTTP::Post.new(@uri.request_uri,
           {'X-ApiKeys' => @x_apikeys,
            'Content-Type' => 'application/json'})
  request.body = post_body_hash.to_json
  response = https.request(request)
end
```

There are no needs to implement all the resource types of Tenable.io because i.e. scanner can not be added adaptively. The resource types and methods necessary for flexible scan of each asset are as follows.

- `editor`
  - list: Returns the template (scan type) list.
- `folders`
  - list: Returns the current user's scan folders.
- `scanners`
  - `list:` Returns the scanner list.
- `scans`
  - `copy:` Copies the given scan.
  - `create:` Creates a scan.
    * `uuid:` The uuid for the editor template to use.
    * `settings.name:` The name of the scan.
    * `settings.enabled:` If true, the schedule for the scan is enabled.
    * `settings.text_targets:` The list of targets to scan (required).
  - `export_request:` Export the given scan. Once requested, the file can be downloaded using the export download method upon receiving a "ready" status from the export status method.
  - `launch:` Launches a scan.
  - `list:` Returns the scan list.

By implementations of these classes and their methods, a sample code to execute "basic scan" to IPv4 address "192.0.2.1" is shown below. `config` is a configuration hash given by a local configuration file.

```
editor = Editor.new(config)
scan_list = editor.list({'type' => 'scan'})
templates = scan_list['templates']
templates.each{|template|
  if template['name'] == 'basic'
    basic_scan_uuid = template['uuid']
  end
}

scanners = Scanners.new(config).list({})['scanners']
scanners.each{|scanner|
  if scanner['name'] == 'scanner'
    scanner_id = scanner['id'].to_s
  end
}

scans = Scans.new(config)
post_body_hash =
  {"uuid" => basic_scan_uuid,
   "settings" => {
   "name" => `uuidgen`.chomp,
   "scanner_id" => scanner_id,
   "enabled" => "true",
   "launch" => "ON_DEMAND",
   "text_targets" => "192.0.2.1",
   "emails" => "reo@cmc.osaka-u.ac.jp",
     }
   }
response = scans.create(post_body_hash).body
```

4. conclusion

The author designed and implemented to check the vulnerabilities in the organization of mesh structure. Ruby implementation is now in-progress. Python implementation will start soon. Automated checking is future works.

References

[1] : *Nessus Network Auditing, Second Edition*, Syngress (2008).

*13 `https://www.json.org/`