

Ethical Hacking Challenge

Project Specification

9 August 2010

Hacking computer systems is illegal. Using the technology you encounter in this project on unauthorised systems could cost you many years in jail as well as your career.

Blueicon Technologies Ltd and its associates take no responsibility for any loss or injury incurred by participants during the course of this project. Blueicon Technologies Ltd reserves all rights to all intellectual property developed, other than what is already established as Free and Open without restriction. Milestones and deliverables are subject to change at any point in the project.



Hacking and malware is becoming a major problem and is threatening the stability of critical national infrastructure. Most hacks go unreported because corporations don't want their customers to think that their information may be exposed.

Weaknesses in systems are known as vulnerabilities and exploits are used to attack these weaknesses. Once a vulnerability is discovered, it can easily be patched, but by then the damage may be done. Zero Day exploits are those that are previously unused and are executed quickly without any opportunity to develop a patch.

Even though the number of available patches is increasing every year, the attack surface area (the number of vulnerabilities) continues to increase even faster. It is this exponentially increasing number of cracks and holes in systems that are the root cause of our collective inability to secure cyberspace.

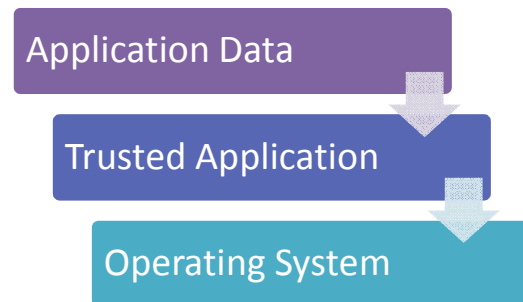
Although there are a great variety of exploits, its anatomy is fairly straightforward.

Essentially, trusted code that handles non-trusted data will malfunction in a way that allows the non-trusted code to hijack a thread of the trusted code.

Seen this way, exploits can make the critical hop from being controlled data into establishing itself as part of the application. The hacker's new powers are only restricted by the Operating System configuration for the application that it attacked.

So, an exploit on a Flash component on a web browser will still only allow the attacker to compromise the user's account and not affect other users on the machine. The Operating System will correctly recognise the exploit as the trusted application.

In order to take control of the entire machine, a second exploit may be required to compromise the Operating System directly (provided the application was running in a restricted mode). This may be easier now that the attacker has access to more system services running from the compromised user's account.



However, a single exploit may be enough if the user account belonged to a crucial person such as a bank manager. Key logging software could be used to steal passwords for attacking other more important systems. Alternatively, the machine may have network access to other vulnerable machines, enabling the hacker to escalate their rights.

Almost all the tools to defend the organisation have major issues. The only way to attain high end security is to formalise the usage paths for applications such as with

Common Criteria (CC). Not only is this expensive, but no unauthorised applications can be executed on deployments, so upgrades are difficult. The Operating Systems still pose a major risk, because they are outside the formalised process. Although Firewalls may be used to hide the Operating System, compromised machines within a LAN may have access to some of these ports.

The vulnerability threat is so pervasive and recombines to form many attack fronts. IT managers must be concerned with vulnerabilities in PDF documents enabled by the Adobe PDF Reader. Switching to *Foxit* or *XPDF* only reduces attack risk because there are fewer known vulnerabilities for these less popular PDF readers.

Users are powerless to defend themselves and can become compromised simply by browsing a legitimate web page that has an exploit for an Adobe Flash component (either through Malvertising or when the web site has been pre-hacked). Even simple graphic rendering routines such as GIF have been known to contain vulnerabilities. Once inside the LAN, new opportunities emerge for the attacker. Advanced Persistent Threats aim to acquire a permanent foothold in the organisation – lying dormant and constantly serving to undermine the integrity of the organisation.

The situation today is dire. Not only do we rely on computers for almost everything in our modern world, but the only way we can get even a modicum of security is expensive and inflexible.

“this cyber threat is one of the most serious economic and national security challenges we face as a nation.” – President Obama May 2009

Industry is not sleeping at the wheel. Instead, a whole new market called Network Access Control is rapidly emerging. NAC solutions attempt to verify that clients contain the latest patches before they are connected to the database or its services.

Unfortunately, the criminals are evolving even faster. They no longer simply try to catch machines on mass to turn into botnet zombies that commit Denial-of-Service attacks. Nowadays, they target key power-users that have special access within an organisation. They create targeted attacks for these selected individuals and take control of their machines, spy on them and use the information in sophisticated ways. It is only software that prevents the uncontrolled release of water from the flood gates of a dam to destroy a town beneath it. Many air-traffic controllers are software based and tampering with these systems can cause disasters, targeted at key individuals.

Industry response continues to be reactive with anti-virus, intrusion detection, performing real-time traffic analysis and packet logging on IP networks. Although these systems can detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts – they all fail to prevent the attack directly.

There is however an emerging branch of technologies that do operate innately and serve to prevent the Malicious Client Problem directly. These approaches, such as Google’s Native Client, isolates executing code so that it cannot attack the host. Alternatively, emulation type approaches create a virtual PC that can be isolated from the network.

The Ethical Hacking Challenge uses this sensible approach to lock execution into a virtualised environment and then to use a firewall to prevent access to the host computer. This model is formalised with the Theory of Absolute Information Security (TAIS). TAIS is new breed of security approaches that not only provides innate protection but locks information into a grid network thereby preventing it from being forwarded beyond pre-specified work areas called Zones.

A real solution

Where Google's Native Client relies on hardware support to enable security, the TAIS model can be implemented on any computation stack.

The Ethical Hacking Challenge encourages students and post graduates to apply the principles of the TAIS because the resulting deployment hides the attack surface area.

TAIS can be easily implemented using an open and free platform known as the BI3, which prevents the applications that run on it from launching an exploit on the BI3 emulator or host itself. The BI3 builds a wall that prevents the AIC from being hijacked and being used to launch further attacks. The BI3 achieves a major milestone in computer security by solving the Malicious Client Problem on a grid network.

The immediate disadvantage for any of these innate type systems is that they can only run applications written for their sandbox type environments, but, once the device can be secured, many possibilities emerge – such as secure online banking and communication.

Project Scope

Please Note:

Allow yourself time to become familiar with the material. If you get stuck, please send an email to david.hughes.021425174@gmail.com with your question. Programming is not required and assembly of the network with the Deployment Environment Specification is available as a cookbook format.

TAIS Security Approach

Most security models are concerned with providing a high level of security in relation to a specific implementation such as trusted computing. The Theory of Absolute Information Security (TAIS) is not related to any specific implementation. Instead, TAIS describes the requirements to achieve a fully secure system, but like any other security model, it will never achieve total security in practice. To use an analogy, TAIS is like the architectural drafts (design) for a skyscraper. Provided the design is followed perfectly then the building can be expected to survive an earthquake. However, if poor quality steel or cement is used then the integrity of the building cannot be guaranteed. TAIS is similar, except that the full specification will never be implemented. The result is that any implementation will suffer varying levels of degradation and security risk.

TAIS is a holistic model that addresses not just the security of the host computer but the information contained on it.

The TAIS Security approach is fairly simple.

- Secure the host computer
 - Turn off all substrate services
 - Use a secure emulator
- Prevent host computer information leakage when rendering information

- Keep information within the grid network and away from the edge
- Use a safe logical model to control information dissemination within the grid network

The level of security achieved is a function of

1. How secure the emulator is, particularly in preventing the Malicious Client Problem.
2. How well the access device can prevent information leakage.
3. How well the administrator assigns¹ access rights to people

Methodology

The Ethical Hacking Challenge (2010) will implement only a narrow range of penetration tests called External Penetration Testing (EPT). These tests will involve installing two secure BI3 emulators on top of their own secured virtualised PCs installed with Operating Systems. On a 3rd virtualised PC, conventional penetration testing tools such as Metasploit and Backtrack are used to attack or scan the system. Both BI3 emulators are connected to each other by a special protocol that prevents attacks via the protocol. Finally, a firewall prevents network access and eliminates any alternative attack route other than via the protocol. Software executing on the BI3 emulator is unable to compromise the host because it is compiled to have zero dependencies and can therefore make no external library calls.

¹ A point of difference for TAIS is that there is no risk that authorised users will disseminate information to unauthorised persons

Communication of the software executing on the BI3 emulator and the host is performed via another secured protocol that prevents risk of the client from installing or executing any new code.

The hacking challenge is structured so that participants create their own implementations based on the TAIS theory and participants attack each other's builds.

There is no idealised target implementation for participants to attack as it is recognised that no implementation will be able to achieve all the TAIS requirements. Finding a fault in an incorrectly implemented deployment is therefore meaningless.

Next year, Ethical Hacking Challenge (2011) will offer Internal Penetration Testing (IPT) that takes a close look at the BI3 emulator architecture and assumes the position of a compromised peer BI3 emulator. Penetration testing tools are less useful here and a more customised attack relating to the architecture is necessary.

Project Environment

Participants are expected to do the appropriate research from either home or Massey University – although there is internet access at Lab B. They will need to reserve times for lab access by filling in the form attached in Appendix A.

During access to Lab B, participants will need to use specific virtualisation software that may include VirtualBox, JPC, Bochs to install both Windows 7 and Linux Operating Systems. This configuration resides on a Virtual Private Network (OpenVPN) that provides firewalled access to an isolated testing network at Lab B from which penetration testing tools will launch various assaults. Some aspects of the TAIS theory are idealised so that faults in the implementation are overlooked. For example, if a fault in OpenVPN firewalling was to be found, it would not undermine the testing results. In this way, deployments can be presented easily and encapsulated within a single zip file.

Timeline

NOTE: Times may have changed from the previous documentation.

9 August 2010	Project scope, specification and deliverables made available (this document)
10 August 2010	Reserve Lab B attendance times
12 August 2010	Project officially starts
23 August 2010	Milestone 1A deadline – Penetration testing research
1 September 2010	Deployment Environment Specification made available
6 September 2010	Milestone 1B deadline – Deployment Plan
13 September 2010	BI3 emulator, Trojan, S-A and S-B images available
20 September 2010	Milestone 2 deadline – Secure Deployment installed onsite at Lab B
4 October 2010	Milestone 3 deadline – Results of self-test and attack adversaries

1 March 2011	Ethical Hacking Challenge 2011 starts
1 April 2011	Internal Penetration Testing (IPT) starts

Project Specification

Milestone 1A – Penetration testing research

1. Research a variety of freely available penetration testing tools (including Metasploit and Backtrack). List the capabilities of each and explain what these mean in its ability to compromise:
 - a. Windows 7 Home Premium
 - b. Linux (any version)
2. List software which when installed on the above Operating Systems will present at least one vulnerability and will theoretically allow the supporting user account to become compromised during an attack.

Milestone 1B – Deployment Plan

Write the Deployment Specification to implement the Vulnerability Shield (see Appendix B) for your project. You must define the following in your Deployment Specification.

- Objective of design
- Explain why the system is expected to be secure
- Describe the software stack that your deployment will operate from. You may use any PC virtualisation specified in the Deployment Environment Specification (such as JPC, QEMU, VirtualBox or Bochs).
- Include the construction sequence that will detail how you will assemble your architecture.
- Execution History will record the exact steps you actually make when you finally build your deployment in

Milestone 2. The Execution History is a much more detailed version of the Construction Sequence. Leave this section blank for now. Each time you construct your architecture, you can make notes here – these are useful pointers for others who may follow in your footsteps.

BONUS TASK (OPTIONAL)

Include any additional components on the system that will deliberately weaken it. These should be standard software components that have known vulnerabilities that you can use as a “Shortcut” to break into your own system. Ideally, use software selected from Milestone 1.A.2. Be clear in your documentation that a real deployment will not contain these doorways, but it is used here to demonstrate that the system is still secure even when it contains vulnerabilities. This protective effect is due directly to the firewall.

If you take on this Bonus Task, then include in your deployment plan a pre-test to break into the system with the firewall disabled to show its effectiveness.

If you can't find an exploit to run from any penetration testing tool, then simulate with some Java Shell software such as JavaSSH (javassh.org) that you will install on Host A and Host C. You can log in from Host C (remotely) with this and attack the configuration on Host A when the firewall is down.

Milestone 2 – Secure Deployment installed onsite at Lab B

1. Start building the Vulnerability Shield deployment. Modify the document if you realise that your abstract design and theory were wrong. If you have made changes, start the construction sequence from the beginning making sure the Execution History exactly matches your construction. Make sure you have the latest BI3 Emulator.
2. Check the Deployment Environment Specification in order to review the results of the software executing in the S-A and S-B. Check that the images are operating correctly.

Milestone 3 – Results of self-test and attack adversaries

1. Write a paragraph that makes a summary of the Vulnerability Shield.
2. Turn off the firewall on the VPN and apply any known exploits for the

target virtualised Operating System or utilise the Trojan simulation to break in if you undertook the Bonus Task of Milestone 1B. You can symbolise your victory by making a copy of the S-A image using the Trojan shell. Be careful not to attack the Host computer directly. Document your results.

3. Turn on the Firewall. Repeat the above step.
4. Install the build of another team or access it if it is already established at Lab B. Attack it first with the firewall turned off, then with the firewall turned on.
5. If you were able to break in, explain what the team did wrong during their implementation. If not, then explain why you couldn't break in.

Appendix A – Lab Attendance Schedule

Please tick or list 30 minute blocks that you would like to attend each week.

Starts	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
10:00							
10:30							
11:00							
11:30							
12:00							
12:30							
13:00							
13:30							
14:00							
14:30							
15:00							
15:30							
16:00							
16:30							

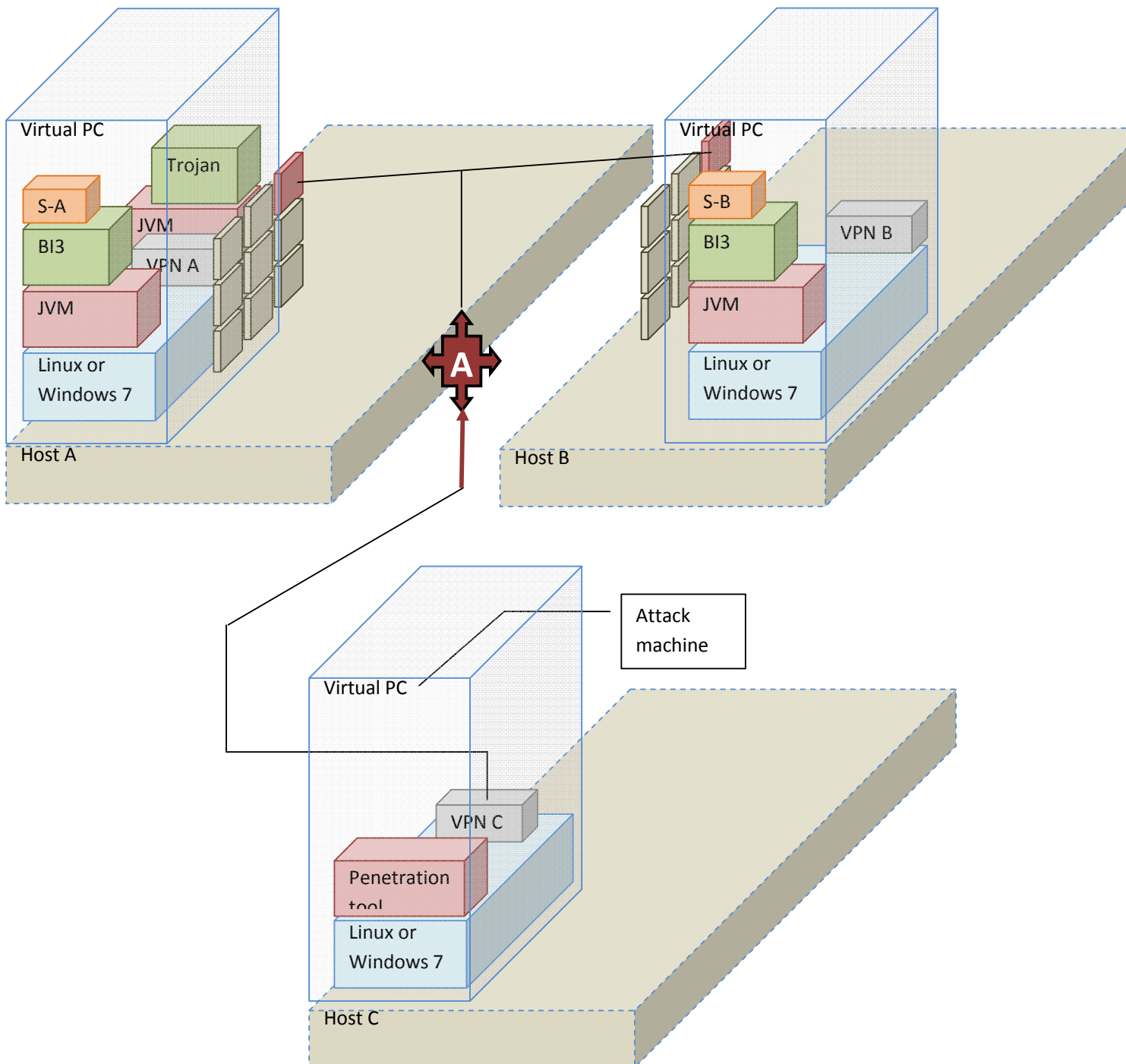
Send to david.hughes.021425174@gmail.com and await approval over the next 10 days.

NOTE: DO NOT ATTEND THE LAB UNTIL YOU HAVE BEEN CLEARED VIA EMAIL TO DO SO. (Your initial milestones do not require access to Lab B)

Appendix B – Vulnerability Shield Architecture

The following is the minimal construction sequence for the Vulnerability Shield architecture. The “A” represents the point of attack during External Penetration Testing (EPT). In this scenario, the attack is targeted at known vulnerabilities in the host Operating Systems. The computers on Host A and Host B

are both trusted and neither must be compromised. Host C contains the build that will be used to attack the configuration. This attack shows that the security will protect against known exploits, but it does not address attacks from within the BI3 architecture.



Construction sequence:

1. Install a clean-room implementation for both Virtual PCs running on Host A and Host B. You may select any version of Linux or Windows 7 for either of the builds. Note: Do not attempt to modify the Host Operating System.
2. Install VPN A with the ability to make an outbound connection on Port 10000 only. Ensure that it cannot receive any inbound connections. Do not connect to the Internet. Refer to Deployment Environment Specification for a step-by-step guide to configure the VPN firewall.
3. Install VPN B with the ability to receive inbound connections from anywhere to Port 10000 only. You will need to use Network Address Translation (NAT) / Port Forwarding. Again refer to the Deployment Environment Specification for a step-by-step guide to configure NAT. Ensure that no outbound connections can be made.
4. Create 3 separate virtual images for the installation. The Deployment Environment Specification explains how to do this too. Once the image is running, install the JVM on all 3 images.
5. Copy 2 instances of the BI3 Emulator with the S-A and S-B images as per the deployment architecture specified previously (Appendix B). If you took the Bonus Task, then install other software containing the selected vulnerabilities or the Trojan shell.
6. Configure the BI3 Port Connection Verification² covered in the

² The Connection Verification ensures that only valid connections are made from desired partner Nodes and on the correct port. The Connection

Deployment Environment Specification.

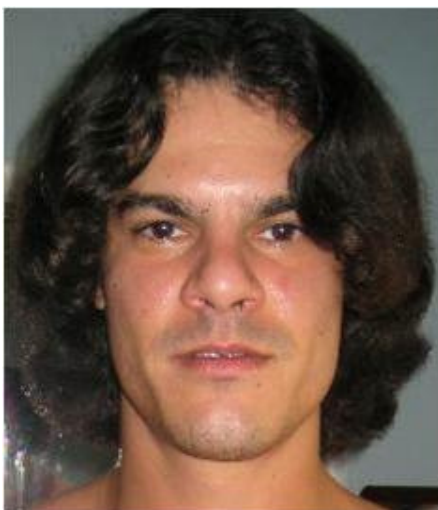
7. Install your penetration testing tools on the virtualised computer on Host C. If you undertook the Bonus Task, then install client software for the Trojan shell.

Verification shuts down the Node and issues a permanent machine compromise warning if it can't bind to the pre-designated communication port in the event that another Operating System process or application running in the host substrate has bound to the selected communication port. The Connection Verification also prevents non-authorised Nodes (identified by both Id and IP) from connecting on the open Port.

Unprecedented 25-Year Sentence Sought for TJX Hacker

By [Kevin Poulsen](#) 

March 19, 2010



Computer hacker Albert Gonzalez deserves a quarter-century behind bars for leading a gang of cyberthieves who stole tens of millions of credit and debit card numbers from a transaction processor and several giant retail chains, federal prosecutors argued in a court filing Thursday night.

“[T]he sentences would be the longest ever imposed in an identity theft case and among the longest imposed for a financial crime, which is appropriate because Gonzalez was at the center of the largest and most costly series of identity thefts in the nation’s history,” wrote Boston-based Assistant U.S. Attorney Stephen Heymann. “He knowingly victimized a group of people whose population exceeded that of many major cities and some states.”

The government also disputed a defense claim that [Gonzalez suffers from Asperger’s disorder](#), a mild form of autism that was grounds for a [slightly reduced sentence](#) in a previous hacking prosecution.

Gonzalez, 28, is set for sentencing next week on three indictments covering virtually every headline-making bank-card theft in recent years, including intrusions at TJX, DSW Shoe Warehouse, Office Max, Hannaford Brothers, 7-Eleven, and Heartland Payment Systems, which alone exposed magstripe data on 130 million credit and debit cards. He performed the intrusions while an informant for the Secret Service.

The hacker's plea agreements contemplate a total prison term of between 17 and 25 years.

In December, Gonzalez's lawyer, Martin Weinberg, argued for the low end of the sentencing range, pointing out that [Gonzalez cooperated with the government](#) against his U.S. co-conspirators and two Eastern European hackers known as "Grigg" and "Annex." Weinberg also argued that Gonzalez was driven by a psychological obsession with computers, submitting a report by a defense-paid psychiatrist that found the hacker's behavior consistent with Asperger's disorder.

Over defense objections, a federal judge allowed a government-paid psychiatrist to also examine the hacker, and that expert came to a different conclusion, noting that Gonzalez appears to have no problems forging social and romantic relationships.

"I found considerable evidence of Mr. Gonzalez's substance abuse and probable antisocial personality disorder," wrote Dr. Mark Mills, in a [report](#) (.pdf) also filed Thursday. "I found no evidence of Asperger's disorder or internet addiction."

Heymann added that Gonzalez's leadership role also belies the Asperger's claim. "Those with Asperger's are almost by definition not leaders," he wrote. "Instead they are followers, often perceived as peripheral, isolated and strange."

[Gonzalez Accomplice Gets Probation for Selling Browser Exploit](#)

By [Kim Zetter](#) 

March 23, 2010 |



A computer security professional who sold Internet Explorer exploit code to credit card hacker Albert Gonzalez was sentenced Tuesday in Boston to three years probation and a \$10,000 fine.

Jeremy Jethro, 29, was paid \$60,000 by Gonzalez for a zero-day exploit against Microsoft's browser, "the purpose and

function of which was to ... enable the conspirators to unlawfully gain access to, and redirect, individual's computers," according to court records.

Gonzalez led a team of hackers who gained unauthorized access to company networks and stole more than 90 million credit and debit card numbers, though it's not clear what role, if any, the \$60,000 zero-day played in the attacks. Jethro's attorney, Stacey Richman, told Threat Level the exploit was a dud.

"The exploit never worked," she said. "None of them worked. There was a question of potentially two [exploits] and neither of them worked."

Jethro pleaded guilty to a misdemeanor conspiracy charge for providing the malware. Under Tuesday's sentence, Jethro will be confined at home, under electronic monitoring, for the first six months of his three-year-long probation.

Richman said Jethro did not know Gonzalez's intended use for the exploit. She also said the judge took into consideration her client's life change in 2006 when he turned to Christianity and "renounced any aspect of any wrongful behavior."

She said Jethro, who is currently working in the computer industry “had spent the years since then entirely in a very proper manner.”

He’s the third person to be sentenced for conspiring with Gonzalez in criminal activity. Last December, Stephen Watt, a former coder for Morgan Stanley, was [sentenced to two years in prison](#) for providing a sniffer to Gonzalez that helped him siphon card data from TJX’s corporate network. Watt was also ordered to pay restitution to TJX in the amount of \$171.5 million.

Earlier this month, Humza Zaman, a former network security manager at Barclays Bank, was [sentenced to 46 months in prison and fined \\$75,000](#) for serving as a money courier for Gonzalez. He was charged with laundering between \$600,000 and \$800,000 for Gonzalez.

Gonzalez is scheduled to be sentenced this week in Boston for his role in the hacks of TJX, Dave & Busters, Hannaford Brothers, 7-Eleven and Heartland Payment Systems. He faces a sentence of between 17 and 25 years. Prosecutors are asking for the latter.