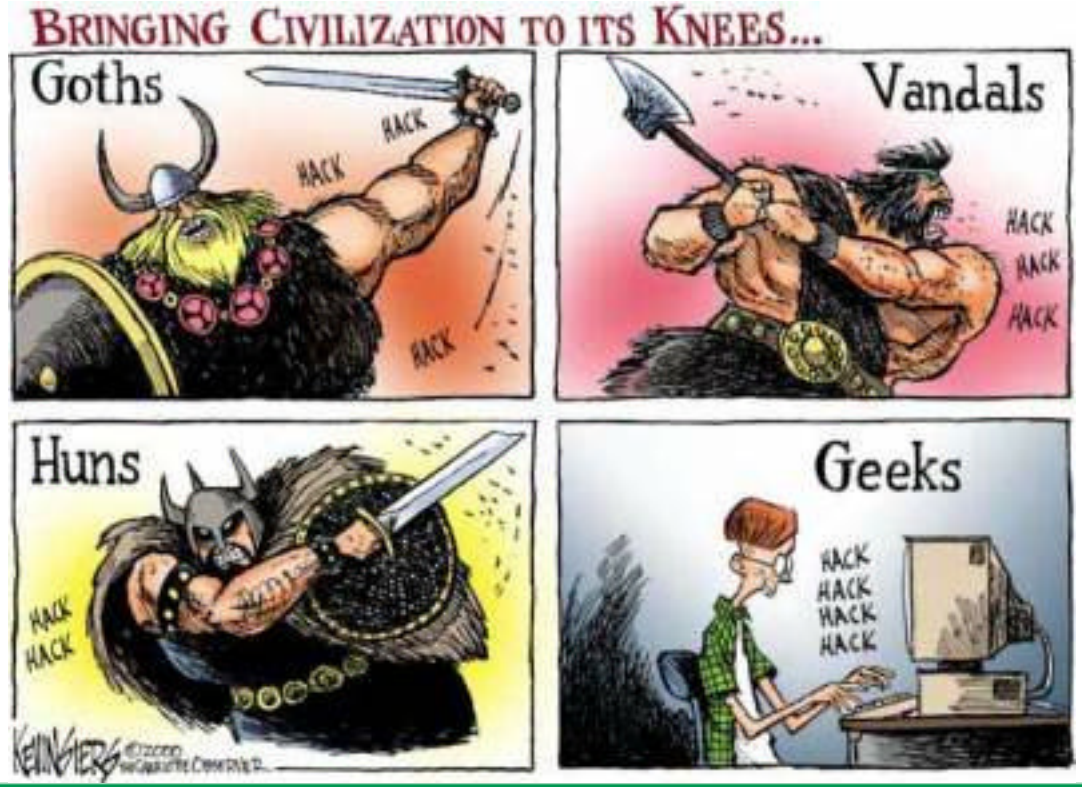
ERNW
Living Security.



How to rate the security of closed source software

Michael Thumann,
mthumann@ernw.de

#whoami

ERNW Living Security.

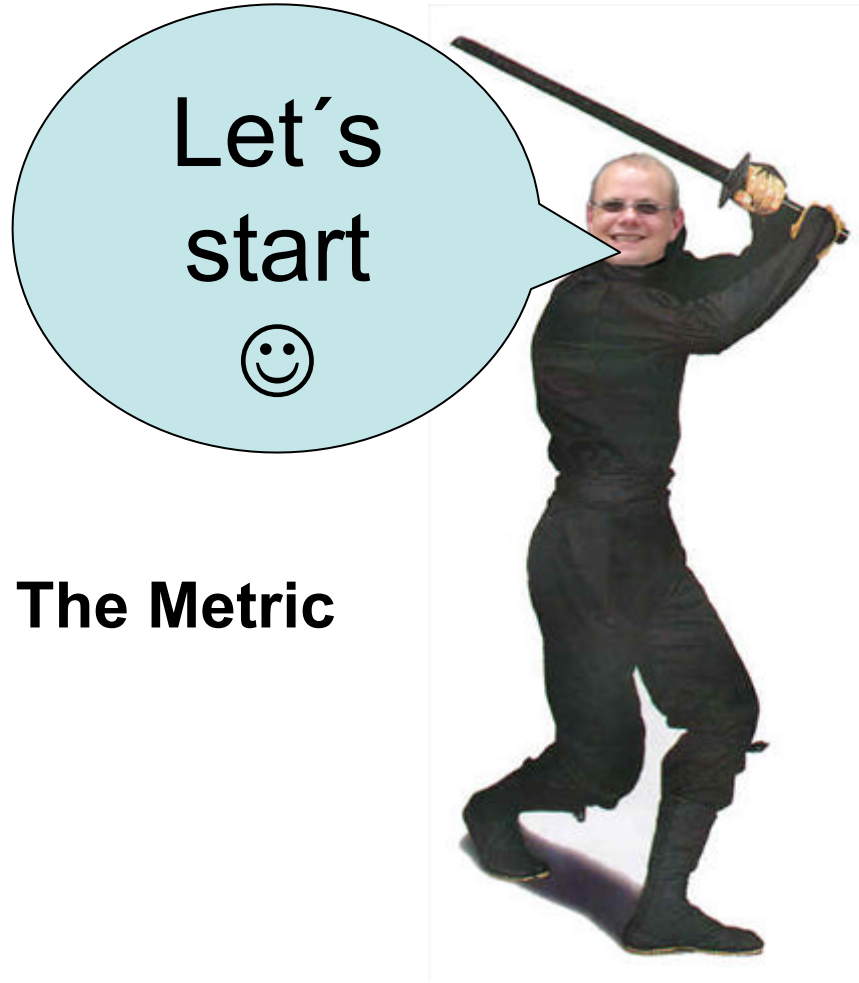
- **Head of Application Security & Chief Security Officer, ERNW GmbH**
- **Talks und Publications:**
  - “Reversing Malware for Business Purposes“, Lodon, RSA Conference 2009
  - “Application Trustworthiness“, Daycon, Dayton 2008
  - “Reversing – A structured approach“, Troopers, München 2008
  - “Hacking Second Life”, Hack-in-the-Box, Dubai 2008
  - “Reversing – A structured approach”, RSA Conference, San Francisco 2008
  - “Hacking SecondLife”, Blackhat Europe, Amsterdam 2008
  - “Hacking the Cisco NAC Framework”, Sector, Toronto 2007
  - “Hacking Cisco NAC”, Hack-in-the-Box, Kuala Lumpur, 2007
  - “NAC@ACK”, Blackhat-USA, Las Vegas, 2007
  - “NAC@ACK”, Blackhat-Europe, Amsterdam, 2007
  - “Mehr IT-Sicherheit durch PenTests”, Vieweg Verlag 2005

- **Main Tasks:**
  - Reverse Engineering
  - Security Research
  - Penentrationstests
  - Code Audits

Agenda

ERNW Living Security.

1. Introduction / the problem
2. Standard approaches
3. Alternate approach: A metric
4. What to measure
5. How to weight the results
6. How to make it portable
7. Putting all the stuff together: The Metric
8. Practical demonstration
9. Discussing the results
10. How to improve the stuff



Introduction

Introduction

ERNW Living Security.

- **Vulnerability Assessment is and gets more common in the enterprise**
- **Web Applications are assessed for security problems to lower the risk and mitigate all problems**
- **Secure coding principles help developers to make better software**
- **More security features are integrated into the development environments (Visual Studio, GCC)**

The problem

ERNW Living Security.

- **Webifying applications is going on and on, but …**
- **There are also products outside there, that are NOT webified**
- **These products are closed source software and old style programming languages are used**
- **There are no easy to use tools available to do any assessment of this type of software**
- **But there's also a need for assessments and analysis of this software**
- **The question "Can we trust this software and process our confidential data with it?" has to be answered**

You think this is a joke?

ERNW
Living Security.

ERNW
Living Security.

Standard Approaches

Standard Approaches

ERNW Living Security.

- **Reverse Engineering**
- **Sandeboxes**
- **Fuzzing**

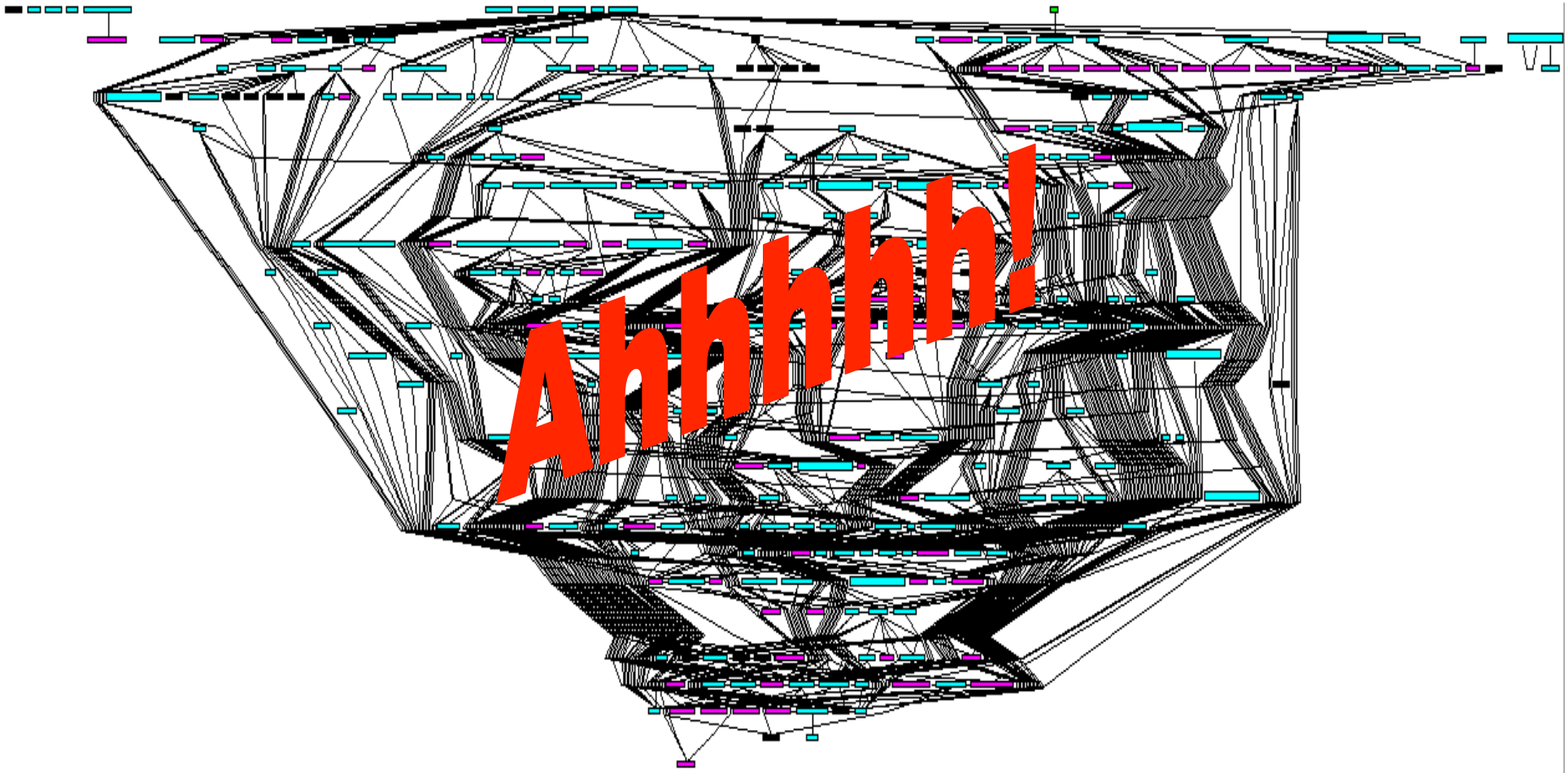
Reverse Engineering

ERNW Living Security.

- **Skilled people required (this kind of knowledge is not common in enterprises)**
- **Security assessment needs time**
- **Each binary must be reverse engineered**
- **Even used 3rd party libraries must be analyzed to rate the overall security**

This is just a Hello World App Flowchart

ERNW Living Security.



Sandboxes

ERNW Living Security.

- **Dedicated to malware analysis**
- **Maybe useful against targeted attacks and backdoors**
- **Doesn’t help to rate the security of the software**

Fuzzing

ERNW Living Security.

- **Fault injection can be very helpful to uncover vulnerabilites like buffer overflows, integer overflows and so forth**
- **Each interface must be fuzzed (protocols, user interface, file formats)**
- **Time consuming**
- **Needs also skilled people**

ERNW
Living Security.

Alternate Approach: A Metric

Why a metric

- **Good metrics can help to measure something, e.g. security**
- **Metrics are comparable, improvement can be measured**
- **Understandable results for all involved people**
- **Can be automated**
- **Timely effective**
- **But depends on what is measured and if this information is reliable and meaningful**

What to measure?

What to measure

ERNW Living Security.

- **Lets focus on windows software because it's mostly used in the enterprise**

Some ideas:

- **Compiler and linker options used**
- **Visual Studio version based on linker version**
- **Signs fort code obfuscation (anti-re, anti-debug)**
- **Import Table**
- **Code Quality metrics (McCabe and Halstaedt)**
- **Vulnerability Scan**

Compiler and Linker

ERNW Living Security.

- **New security features are available in actual versions of the development environment**
- **From Microsofts SDL: “Use actual version of development environment”**
- **Check if DEP is supported**
- **Check for ASLR**
- **Check for SafeSEH usage**
- **All that stuff can be obtained from the PE header ☺**

- **Or check for the /GS flag (stack canaries)**

Code Obfuscation

ERNW Living Security.

- **Packers and cryptors can be detected by signatures**
- **Packers and cryptors can be detected by entropy**
- **Import table to short**
- **Debugger detection**

Import Table

- **Check for banned functions (strcpy, strcat …)**
- **Network functionality within the program (look for the corresponding APIs)**
- **Registry access (look for the corresponding APIs)**
- **Create files functionality**

Code Quality

ERNW Living Security.

- **Using code complexity metrics**
- **Why? Because complexity kills ☺**
- **McCabe (counts decisions)**
- **Halstaedt (counts operators and operands)**
- **But disassembly of each single binary must be generated to calculate the metric based on it**

Vulnerability Scan

ERNW Living Security.

- **Do an analysis of the disassembly and look for vulnerabilities (like Bugscam years ago)**
- **Implement new approaches to identify the presence of vulnerabilities (Recurity Labs is working on some stuff)**
- **But disassembly of each single binary must be generated for an analysis**

Other stuff

- **Code signing**
- **Crypto**
- **More ideas? RFC ☺**

Results

ERNW Living Security.

Possible results based on the check are:

- **0 = does not improve security rating**
- **1 = improves security rating**

How to weight the results

How to weight the results

ERNW Living Security.

- **Some of the stuff we can measure has more value for rating the security than others (DEP, ASLR, SafeSEH, Linker version)**
- **Code obfuscation is also used to protect the intellectual property, so how do we have to weight this?**
- **Is network functionality a security problem? No!! But there are more risks.**
- **So we have to rate the value of the check in terms of improving security**
- **Some checks are more important than others**

How to weight the results

Assuming a security feature is present, the weight is defined as follows:

- **1 = may have some impact on security**
- **2 = can improve security**
- **3 = significantly improves security**

Reliability

We have to implement a criteria that gives us some information about the reliability of our checks to do a proper rating of the security. Reliability is defined as

- **1 = low reliability**
- **2 = medium reliability**
- **3 = high reliability**

ERNW
Living Security.

How to make it portable?

Make it portable

ERNW Living Security.

- **The approach shouldn’t be limited to windows software**
- **But the things we can check differ, depending on the target operating system, even the number of checks we can do**
- **So the checks must be replacable (check different things for different target OSs)**
- **To construct a metric the final results must be in the same range. That’s the only way to have a global rating! OS independent.**

ERNW
Living Security.

Putting all together: The Metric

Lets define the checks

ERNW Living Security.

Definition of check (weight, reliability, result)

- **DEP check (3,3,r)**
- **ASLR check (3,3,r)**
- **SafeSEH check (2,3,r)**
- **Linker check (3,3,r)**
- **/GS check (3,2,r)**
- **Not packed check (1,2,r)**
- **No Banned functions check (3,2,r)**
- **No networking (1,2,r)**

Lets define the checks

ERNW Living Security.

Definition of check (weight, reliability, result)

- **No registry (1,2,r)**
- **No file creation (1,2,r)**
- **McCabe (1,2,r)**
- **Halstaedt (1,2,r)**
- **Vulnerability check (3,2,r)**

Some math ☺

ERNW Living Security.

The final result for a check must be calculated:

- **Value (V) = (weight + reliability) * result**
- **Possible results = {0,1,2,3,4,5,6}**

Example:

- **DEP weight: DEP significantly improves security**
- **DEP reliability: high (can be detected reliable in the PE Header)**
- **DEP check (weight, reliability, result)**
- **(3,3,1) = 6**
- **(3,3,0) = 0**

Next step

ERNW Living Security.

Assuming a positive result we have to calculate the value for each check:

- **DEP check (3,3,1) = 6**
- **ASLR check (3,3,1) = 6**
- **SafeSEH check (2,3,1) = 5**
- **Linker check (3,3,1) = 6**

Next step

- **Summarize these values: 6 + 6 + 5 + 6 = 23 => 100%**
- **Calculate the result of the real check: 6 + 6 +0 +6 = 18 => ?? %**
- **18 * 100 / 23 = 78,26% = 78,26**

- **We call the result TTI = Thumann's Trustworthiness Index**

Why TTI? Because I'm getting older

ERNW Living Security.

Working with the results

ERNW Living Security.

Result	Rating	Description
< 34	Red	Not trustworthy
>= 34 and < 67	Yellow	Can proccess public and internal data
>= 67	Green	Can process confidential data

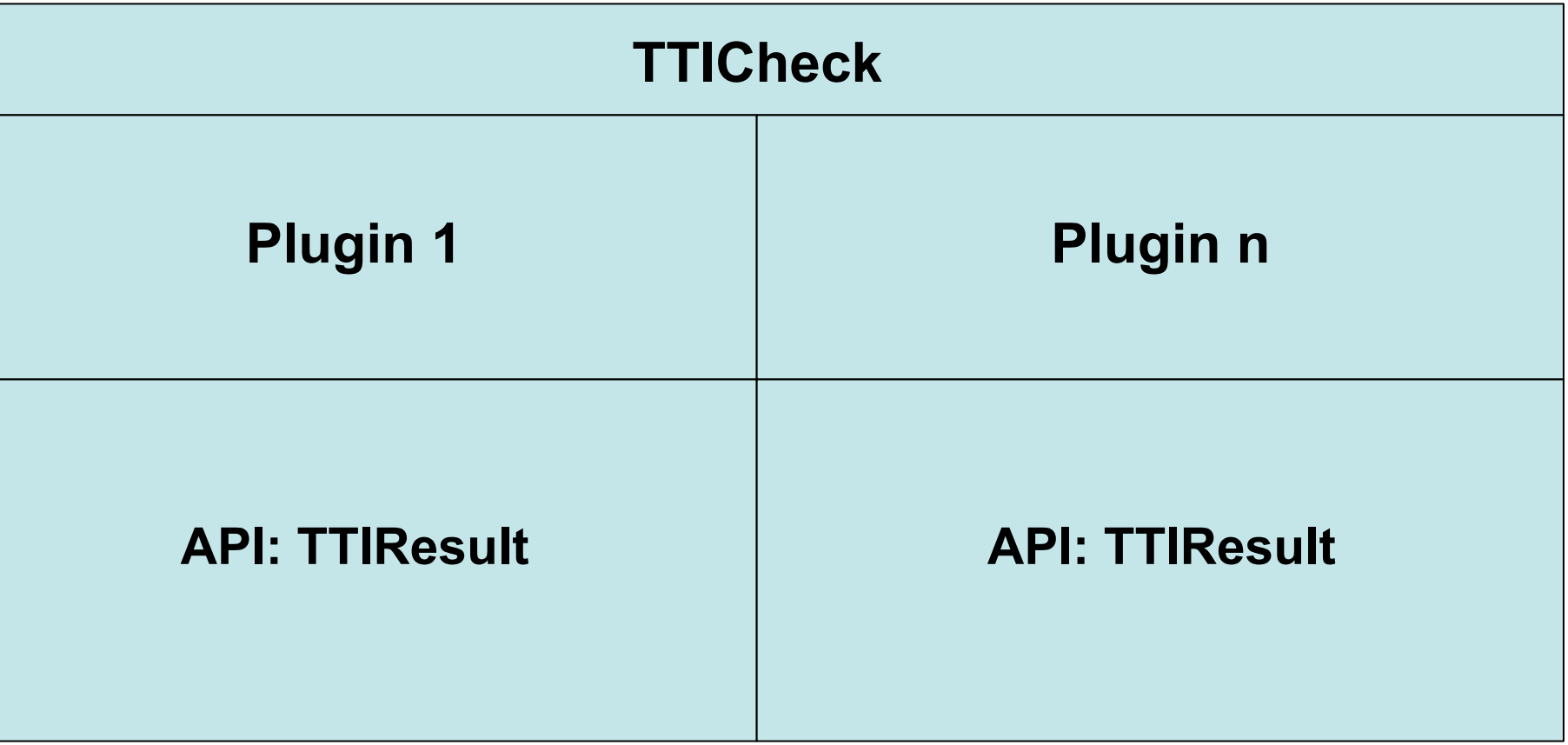
ERNW
Living Security.

Demo Time

```
root@hal#
```


root@hal#

Tool architecture



Tool architecture: Config File

ERNW Living Security.

Plugin,Weight,Reliability

dep.dll,3,3

aslr.dll,3,3

safeseh.dll,2,3

linker.dll,3,3

gs.dll,3,2

Tool architecture: API

ERNW Living Security.

```
// TTI Plugin Template by Michael Thumann

#include "stdafx.h"
#include "stdio.h"
#include "windows.h"

#define DLL extern "C" __declspec(dllexport)

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    return TRUE;
}

DLL DWORD __cdecl TTIResult(char *path, int result )
{
    // Here comes the magic stuff
    return 0;
}
```

0 = check failed
1 = check ok

Tool progress

ERNW Living Security.

Plugin	State
DEP check	implemented
ASLR check	implemented
SafeSEH check	implemented
Linker check	implemented
/GS check	Not implemented
Not packed check	Not implemented
No Banned functions	Not implemented
No networking check	Not implemented
No registry access check	Not implemented
No files creation check	Not implemented

Tool progress

ERNW Living Security.

Plugin	State
McCabe check	Not implemented
Halstaedt check	Not implemented
Vulnerability check	Not implemented
Code signed check	Not implemented
State of the art crypto check	Not implemented

Discussing the results

Pitfalls

ERNW Living Security.

- What about implemented backdoors or covered channels?
- What about vulnerabilities that can't be mitigated?
- What about cleartext network communication?
- What about compliance requirements like encrypted storage of data?
- Can we replace a detailed security assessment with this approach?
- Are we able to do a detailed security assessment of each application that is used in our network?

Advantages

ERNW Living Security.

- **We can do an easy rating of the security based on some principles for developing secure software**
- **No detailed assessment required**
- **No legal problems when reverse engineering software**
- **We know, if the application was developed with security in mind**
- **Thinking about security always improves security**
- **And to be honest: Who of you is doing any security assessments of closed source software ☺?**

ERNW
Living Security.

Improving the results

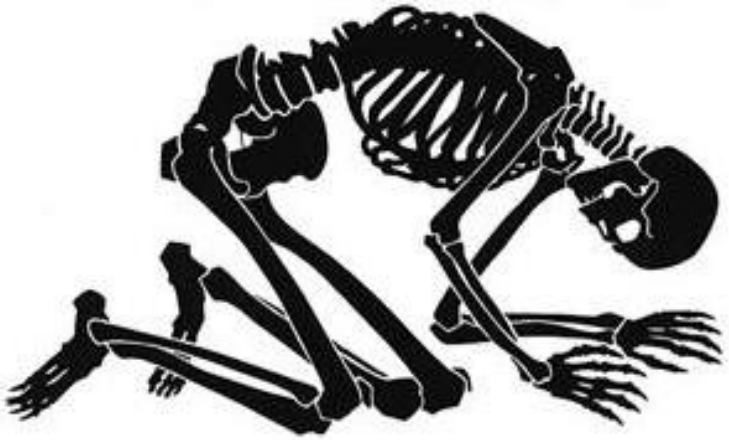
Work in progress

ERNW Living Security.

- **If we do more reliable checks that significantly improve security, we get better results**
- **More checks also means that it is less important, if one check fails, maybe because some functional requirement had a higher priority**
- **The concept has to prove its value in more assessments (a concept based on this approach is actually in the implementation phase at a major well known e-commerce site)**
- **Tool must be improved (UI, reports, more plugins …)**

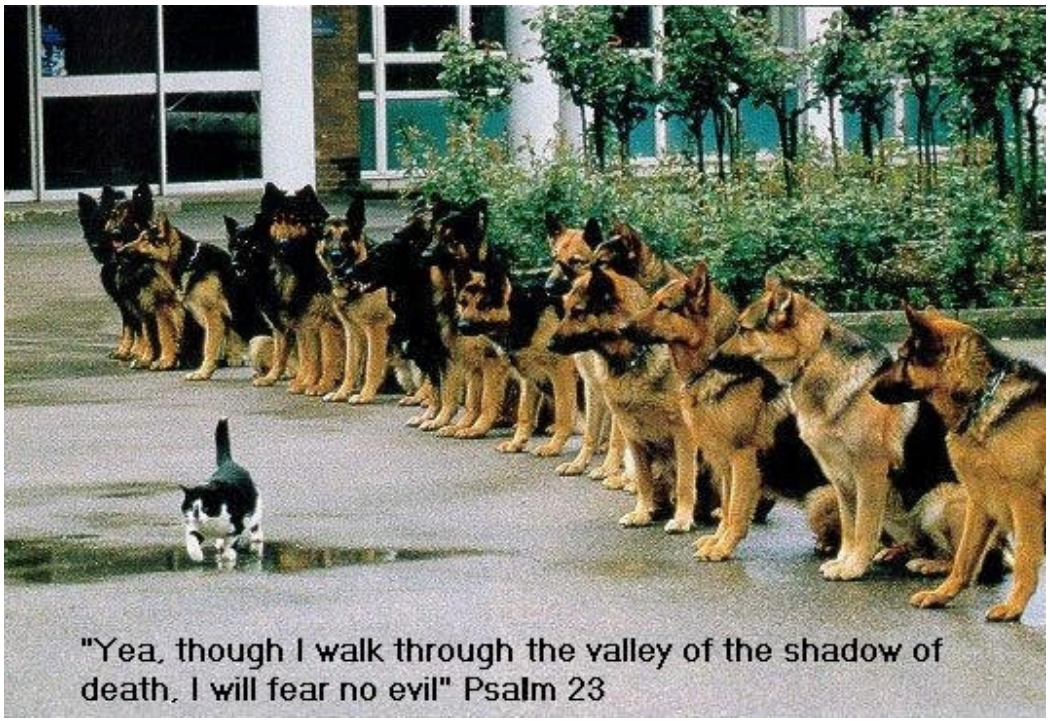
ERNW
Living Security.

Questions?
And Answers…



ERNW
Living Security.

Thank you for your patience



Contact me @

Michael Thumann
ERNW GmbH
Germany
mthumann@ernw.de
www.ernw.de